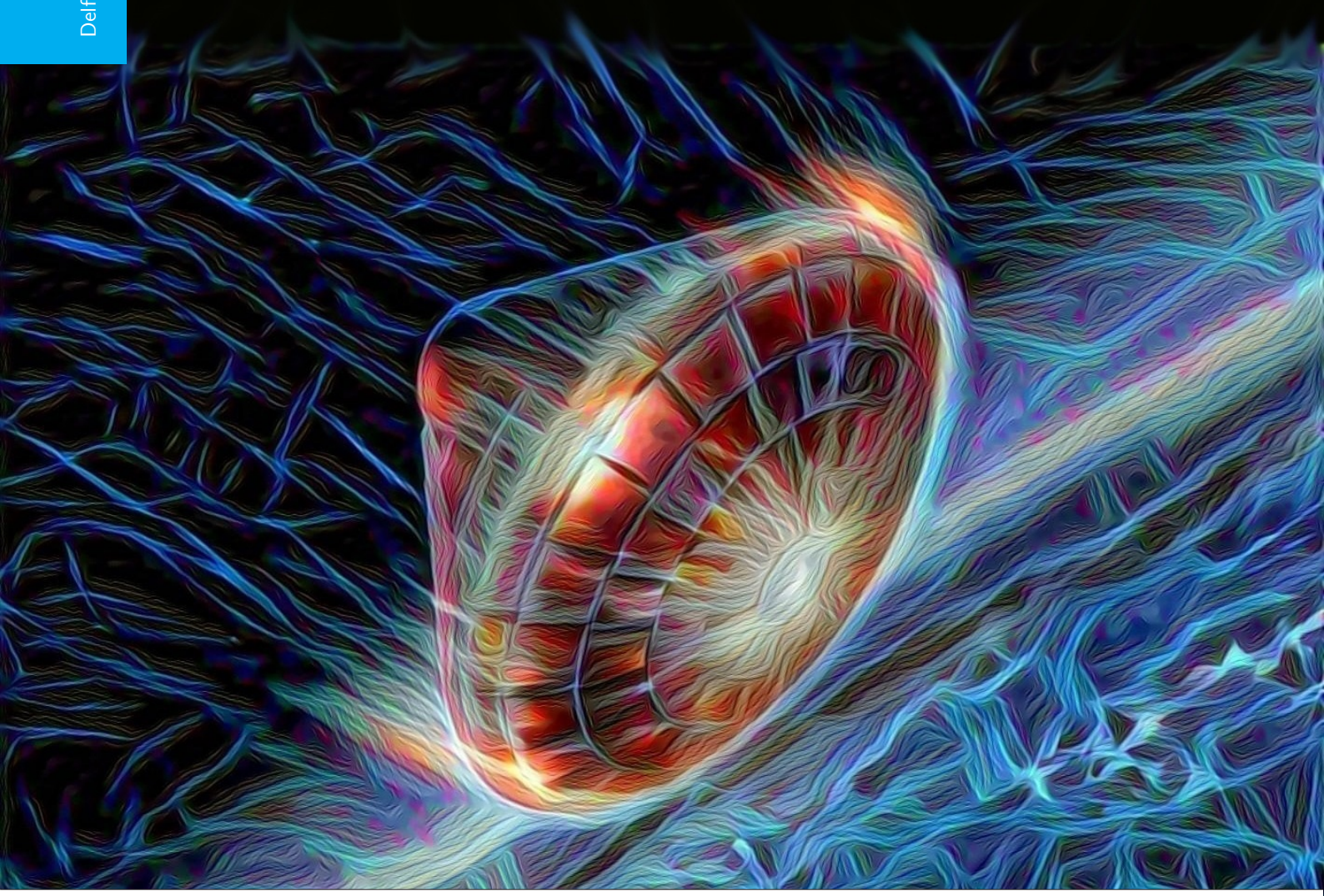
Aerocapture Guidance

Numerical and Artificial Intelligence Solutions

E. M. Zucchelli

Delft University of Technology



TU Delft Delft University of Technology

Challenge the future

Cover image credit: ESA - D. Ducros (2002); Atmospheric Re-entry Demonstrator - artist's impression. Modified with Google Deep Dream Generator

Aerocapture Guidance

Numerical and Artificial Intelligence Solutions

by

E.M. Zucchelli

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday November 25, 2016 at 2:00 PM.

Student number:	4332067	
Supervisor:	Dr. ir. E. Mooij,	TU Delft, Astrodynamics and Space Missions
Thesis committee:	Prof. dr. ir. P. N. A. M. Visser,	TU Delft, Astrodynamics and Space Missions
	Dr. ir. E. van Kampen,	TU Delft, Control and Simulation
	Ir. J. Geul,	TU Delft, Astrodynamics and Space Missions

This thesis is confidential and cannot be made public until November 24, 2016.

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

"We may have knowledge of the past but cannot control it;
we may control the future but have no knowledge of it."

Claude E. Shannon

ABSTRACT

The aerocapture maneuver has the potential of literally opening new worlds to space exploration. Despite this, it has never been attempted before.

The guidance logic is a key element for the success of an aerocapture. It has to be able to guide the vehicle in a variety of adverse and highly perturbed conditions. Moreover, it should possibly do so in an optimal way. During this research, a lunar return aerocapture on Earth is used as a reference mission. The guidance problem is strictly related to trajectory analysis, and thus this is carried out first. It is shown that, for the case in object, the ideal trajectory that minimizes the ΔV also minimizes all the constraints, as well as the total heat load. Because of this, the aerocapture guidance for Earth can be reduced from a multi-objective constrained problem to a single objective optimization. The objective is minimization of the ΔV for the worst case conditions.

The optimal aerocapture numerical predictor-corrector (NPC) developed by Lu et al. [2015] is a good starting point for the analysis of the guidance. Its main drawbacks are the fact that it makes use of a large number of trajectory integrations at each call, on average between 10 and 20, and that it requires an extensive tuning effort. During this research, two major modifications have been proposed: a different numerical method, and a different trajectory planning in the NPC. With the first, the number of iterations is reduced to 2, without any significant loss in performance. With the second, the required tuning becomes much less demanding. Specifically, this is possible because of the inclusion of a simplified model of the rotational kinematics of the spacecraft. It is concluded that this newly developed guidance can perform as well as the original one, but with less tuning and with only two iterations per guidance call. Specifically, for initial flight-path angles between -5.6° -5.1°, the guidance can ensure a safe orbit insertion with a maximum ΔV of $80\,\mathrm{m\,s^{-1}}$, a maximum load factor of 5, and a maximum integrated heat load of $650\,\mathrm{MJ\,m^{-2}}$. Such a result is valid also for highly perturbed environments, including turbulent atmospheres. These results are used as benchmark to compare the performance of the artificial intelligence guidance.

The kind of artificial intelligence that will be used is an artificial neural network (ANN). An ANN is a computational approach that is capable of reproducing any function, and can do so with very little computational requirements. One way of training a neural network consists of supervised learning. A large amount of optimized trajectories is generated, and then the network is taught how to follow them. With such a method, for the same conditions as before, the neural guidance can meet the requirements with a worst case scenario ΔV of $210\,\mathrm{m\,s^{-1}}$. Despite less performing than the optimal NPCs, this result is very interesting, since the ANN does so with a fraction of the computational requirements. It is also believed that the training design can be improved, and thus the performance as well.

With reinforcement learning (RL), it is possible to let the networks learn by themselves, directly interacting with the simulation environment. The goal is that of minimizing the objective function at the in a variety of perturbed situations. In this research, Super Symmetric Sampling (SyS) Policy Gradient with Parameters Exploration (PGPE) is used. With such a method, the ANN guides the vehicle along a trajectory; it then approximates a gradient of the objective with respect to the network's parameters by sampling, and it updates itself. At every iteration, a trajectory with different conditions and perturbations is simulated. Sup SyS PGPE seemed a very promising design, but also very challenging. Eventually, its performance was not very performing, because of a particular feature of the design. For the networks so trained in this research, aerocapture can be ensured for a maximum ΔV of $600\,\mathrm{m\,s^{-1}}$.

PREFACE

Suppose you have never written an MSc thesis, and want to learn how to write a proper one. One way of doing that could be listening to your supervisor, who gives you guidance, tells you when you are doing something wrong, and how to improve your work. This is a similar case to supervised learning.

However, the adventurous learner may prefer not to listen to anyone, and write his thesis his way. At the end, he would be graded. Therefore, the first time the student would most likely fail. Nonetheless, being adventurous, the learner would not give up, and keep on trying. He would also remember, and learn to recognize, what patterns lead to better results. After many attempts, years, sweat and tears, the student would have gained enough experience to know how to write the best possible thesis. Without diving into philosophical debates on whether such a student shall be called adventurous or stubborn, this second approach is more similar to reinforcement learning.

These are two of the methods that will be used to solve the problem of the optimal aerocapture guidance. At this point, a little of creativity is needed to imagine how those examples may be compared to such a problem, but hopefully everything will be clear after having read this report.

I will not attempt to be the judge of whether I was the diligent learner, or the adventurous, stubborn one. I guess only my supervisor Dr. Erwin Mooij can tell. At this point, I want to express my gratitude to him, for the opportunity he gave me with this research, and the advices he shared with me. I am particularly thankful for being patient during some of the interesting, but also never ending, discussions we have had.

I am also grateful to Ir. Jacco Geul for having encouraged me to enter the world of reinforcement learning, and to Dr. Erik-Jan van Kampen for having met with me a couple of times. To both of you, and to Prof. Pieter N.A.M. Visser, I say thanks for agreeing to be part of my Thesis Committee.

Thanks also to all of my friends here, back home and around the world. I am sorry if I missed the Bouwpub a couple of times, and I know that writing a thesis is not a good excuse. Another considerable portion of my gratitude goes to my friend Mihir, who offered me more than once his expertise in using LaTeX, and to Bronius, who has been a great companion during the writing of this thesis. I firmly believe that these months in which you have been in Leiden have improved our friendship.

Eventually, I would now like to thank my parents, who always fully supported me in every way, and my brother, who has been many times a great life advisor and a role model.

Enrico M. Zucchelli
Delft, November 2016

CONTENTS

LIST OF ACRONYMS

ANN	artificial neural network
APC	analytic predictor-corrector
BFGS	Broyden-Fletcher-Goldfarb-Shanno
c.o.m.	center of mass
CNES	Centre National d'Études Spatiales
CNSR	Comet Nucleus Sample Return
DCM	direction cosine matrix
DNN	deep neural network
DOF	degrees of freedom
EC	energy controller
EDL	entry, descent and landing
EDL-SA	Entry, Descent, and Landing Systems Analysis
ELM	extreme learning machine
ESA	European Space Agency
FFNN	feed forward neural network
FIM	Fisher Information Matrix
FTC	feedback trajectory control
GNC	guidance, navigation and control
GRAM	Global Reference Atmospheric Model
GUACA	Global Upper Air Climate Atlas
HYPAS	hybrid predictor-corrector aerocapture scheme
ISPT	In-Space Propulsion Technologies
ISS	International Space Station
LaRC	Langley Research Center
LAURA	Langley Aerothermodynamic Upwind Relaxation Algorithm
LM	Levenberg-Marquardt
MAP	Middle Atmosphere Program
MDP	Markovian decision process
MGS	Mars Global Surveyor
MPCV	Multi-Purpose Crew Vehicle
MSL	Mars Science Laboratory
MSR	Mars Sample Return
NASA	National Aeronautics and Space Administration
NPC	numerical predictor-corrector
PGPE	Policy Gradient with Parameters Exploration
PICA	phenolic impregnated carbon ablator
PID	proportional-integral-derivative
POST	Program to Optimize Simulated Trajectories
RAAN	Right Ascension of the Ascending Node
RCS	reaction control system
ReLU	rectified linear units
RK	Runge-Kutta
RL	reinforcement learning
SGD	stochastic gradient descent
SNOPT	Sparse Nonlinear OPTimizer
SQP	sequential quadratic programming
SyS	Symmetric Sampling
TD	time delayed
TPC	terminal point controller
TPS	thermal protection system
TRL	technology readiness level

LIST OF SYMBOLS

Term	Description	Units
	Latin letters	
$\mathbf{a}$	Acceleration vector	$\mathrm{m\,s^{-2}}$
a	Semi-major axis of orbit	m
a	Speed of sound	$\mathrm{m\,s^{-1}}$
$\mathbf{C}$	Direction cosine matrix	-
$\mathbf{C}_A$	Aerodynamic force coefficients vector	-
D	Drag force	N
E	Specific energy	$\mathrm{m^2\,s^{-2}\,kg^{-1}}$
e	Orbital eccentricity	-
$\mathbf{F}_{ij}$	Force applied by body i acting on body j	N
f	Flattening parameter	-
$\mathbf{F}_A$	Aerodynamic force vector	N
g	Gravity central acceleration	$\mathrm{m\,s^{-2}}$
g_0	Earth's gravity central acceleration at sea level	$\mathrm{m\,s^{-2}}$
h	Altitude of the vehicle from reference surface of celestial body	m
H	Hamiltonian functional	-
i	Orbital inclination	rad
$J_{m,n}$	Higher order gravitational field terms	-
L	Lift force	N
m	Mass of a body	kg
M	Mach number	-
n	Load factor	$\mathrm{m\,s^{-2}}$
o_i	i^{th} f the neural network	-
p	Semi-latus rectum	m
p	Gas pressure	Pa
$\dot{q}$	Heat flux	$\mathrm{W\,m^{-2}}$
$\bar{q}$	Dynamic pressure	Pa
$\mathbf{q}$	Quaternion vector	-
R	Reward function	-
r	Distance from origin	m
R	Specific gas constant	$\mathrm{J\,K^{-1}\,kg^{-1}}$
R_n	Radius of vehicle nose	m
R_E	Earth equatorial radius	-
$\mathbf{r}$	Position vector	m
S_{ref}	Vehicle reference surface	$\mathrm{m^2}$
t	Time	s
T	Gas temperature	K
U	Gravitational potential	$\mathrm{m^2\,s^{-2}}$
$\mathbf{V}$	Velocity vector	$\mathrm{m\,s^{-1}}$

Term	Description	Units
V	Speed	$\mathrm{m\,s^{-1}}$
V	Value function	-
$\mathbf{x}$	State vector	-
	Greek letters	
α	Angle of attack	rad
χ	Velocity heading angle	rad
δ	Longitude	rad
η	Update factor	-
γ	Flight path angle	rad
λ	Co-state function	-
μ	Gravitational parameter of celestial body	$\mathrm{m^3\,s^{-2}}$
Ω	Right ascension of ascending node	rad
ω	Argument of periapsis	rad
ω_{cb}	Angular velocity of celestial body	$\mathrm{rad\,s^{-1}}$
$\boldsymbol{\omega}$	Angular velocity vector	$\mathrm{rad\,s^{-1}}$
ρ	Density	$\mathrm{kg\,m^{-3}}$
σ	Bank angle	rad
τ	Latitude	rad
θ	True anomaly	rad
θ	Latitude angle in spherical coordinates	rad
$\boldsymbol{\theta}$	FFNN parameters vector	-
	Frame indices	
$\mathcal{A}$	Aerodynamic frame	
$\mathcal{I}$	Inertial planetocentric frame	
$\mathcal{R}$	Rotating planetocentric frame	
$\mathcal{V}$	Vertical frame	
	Subscripts	
0	Initial state/value	
a	At apoapsis of orbit	-
c	Pertaining to circular orbit	
cmd	Commanded control variable	
δ	Latitudinal component	
E	Pertaining to Earth	
e	Entry state/value	
ex	Exit state/value	
f	Final state/value	
p	At periapsis of orbit	
r	Radial component	
w	Pertaining to wall	
x	x component	
y	y component	
z	z component	
∞	At infinity	

1

INTRODUCTION

The aerocapture maneuver is a technology that allows orbital capture with only one pass inside the atmosphere of a planet, almost null propellant use and in the time of around half of the desired final orbital period. Alternative to the traditional propulsive capture and to aerobraking, its benefits in terms of payload-mass increase are believed to highly reduce the cost of many missions, and even enable otherwise impossible ones. The maneuver was first proposed by Cruz [1979], and in its original description it consisted of three main phases: the exoatmospheric approach, the atmospheric pass and the periapsis raise.

The exoatmospheric trajectory may rely on open-loop guidance, navigation and control (GNC), and plays a fundamental role in the success of the maneuver, since it strongly affects the initial conditions of the following phase. During the atmospheric pass, the vehicle enters the atmosphere and the required ΔV to have a certain apoapsis altitude is given to the vehicle by atmospheric drag. The latter phase requires autonomous and closed-loop GNC, because the plasma surrounding the vehicle would make impossible any external communications, and because the large instability and sensitivity of the trajectory make an unguided flight virtually impossible. The vehicle is then guided in a way such that, after exiting the atmosphere, the spacecraft is injected into an orbit lying on the same plane and having the same apoapsis altitude as the target one. During the atmospheric pass, the usual way of controlling the trajectory consists, for a capsule-like vehicle, of bank-angle modulation. When the apoapsis is reached, a thrust burn is applied to raise the periapsis to the desired one. At this point, the final orbit is almost obtained, and hopefully only negligible corrections are needed. A more detailed breakdown of the maneuver is shown in Fig. 1.1.

The difficulties in achieving aerocapture come from two main factors. First, the high speed of the trajectory, which, together with the non-linearity of the dynamics, makes the system extremely unstable. Secondly, after a certain point, the atmosphere becomes too rarefied, and the vehicle has no more means of controlling and correcting the trajectory.

An aerocapture has never been attempted before, despite the many benefits it could give in terms of mass reduction [Percy et al., 2005] and the fact that it has been studied for the past four decades. It also has many similarities with skipping entry, a maneuver the Apollo guidance had been designed for, and aerobraking, which has instead been successful[1] a handful of time. IAerocapture has been proposed for many missions, and it is an important part in almost all of the architectures that were considered by National Aeronautics and Space Administration (NASA) for their Entry, Descent, and Landing Systems Analysis (EDL-SA). The latter is a study commissioned in 2008 to identify the roadmap of entry, descent and landing (EDL) technology needed to successfully land large payload, including manned vehicles, on Mars [Ciancolo et al., 2011].

[1] With a rather low success rate, though.

A determining aspect of aerocapture consists of how the vehicle is guided during the atmospheric flight: we enter now the domain of GNC. Guidance is defined by the Oxford Dictionaries as "*advice or information aimed at resolving a problem or difficulty, especially as given by someone in authority*". This, although helpful, is not very precise. Fortunately, the same Oxford Dictionaries provide an engineering definition:

> *The directing of the motion or position of something, especially an aircraft, spacecraft, or missile.*

The goal of the guidance system is thus that of giving the right direction to the motion. In non propelled EDL, this translates to give steering commands to the aerodynamic angles. To achieve its goal, a guidance logic has to be complemented by a system telling it where it is, the navigation, and another one actuating its directing, the control. All together, they make the GNC.

The commands given by the guidance shall be such that, if actuated by the controller, they would lead the vehicle to the desired final orbit, while satisfying all the constraints, both at the end and during the entirety of the trajectory. Moreover, this should preferably be done in an optimal way. All these requirements imply that, in addition to knowing where it is, the guidance system should also know what its environment is, and how its commands will affect the future states. It is important that the guidance acts in prospect of not only the immediate future, but of the long term as well.

An additional complication comes from the fact that the commands should be provided at a rather high frequency (usually not smaller than 1 Hz). This is required because of the fast and non-linear dynamics governing the trajectory.

Two common methods to guide a spacecraft during aerocapture are analytic predictor-corrector (APC) and NPC. As the name suggests, predictor-correctors, both analytical and numerical, rely on a trajectory that is "predicted" at a certain moment; at each successive guidance call, the prediction is then "corrected" based on the new state. One of the many alternatives to these methods consist of having a stored trajectory, that is then tracked Wingrove [1963]. An example of aerocapture guidance making use of this method is the terminal point controller (TPC). Eventually, another very popular concept for aerocapture is the energy controller (EC), which makes analytical assumptions, and aims to achieve a certain reference energy at the exit of the atmosphere.

In all traditional guidance methods, the knowledge about how the commands of the guidance will affect the future states is explicit, in the sense that it comes from some modeling of the environment. However, it is possible to try and solve the problem in a few different ways, that require no explicit modeling of the environment, at least within the guidance logic. Artificial intelligence is one such solution. Two branches of this technology are applied in this research: supervised learning and RL. The first consists of learning from labeled input-output pairs. This means that the commands (outputs) first need to be found in a different way, and then taught to the learning element. The second makes use of learning by trial and error. The learning element explores the environment, receives rewards, and exploits what it learned. It does all this with the goal of obtaining the largest amount of rewards. In this case, there is no need of labeled data: the model of the environment, together with an appropriate reward function, is sufficient. Many keywords have been just been mentioned, and they will become more clear further in the report. At this point of the discussion, the reader does not need a technical definition of those keywords, and can rely on their traditional meaning.

The use of the two just described methods may have many advantages over traditional guidance solutions. In fact, artificial intelligence solutions have the capability to provide the optimal command in real-time, whereas both APCs and NPCs generally do not aim to achieve an optimal trajectory. This is because predictor-correctors need to solve the command at a very high frequency, and an optimization problem cannot usually be solved at such a frequency. Moreover, also a guidance

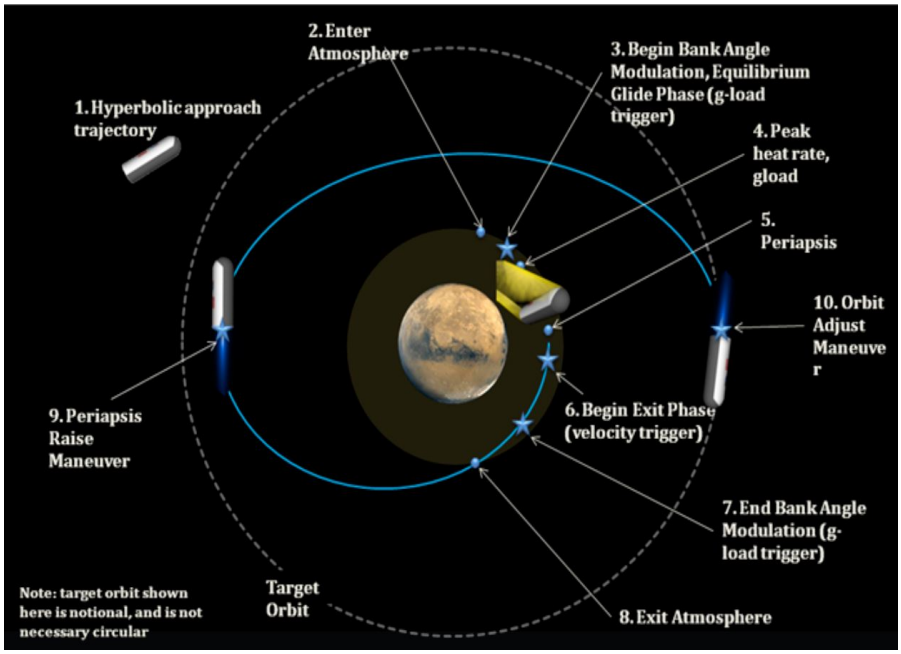


Figure 1.1: Aerocapture phases [Zumwalt et al., 2010].

that tracks a stored trajectory, even if such trajectory were optimal, would not provide an optimal command, due to the linearization occurring in tracking. Lu et al. [2015] exploit an analytical, simplified solution of optimal control theory to have an NPC with a structure such that the commands are closer to be optimal. The method requires very large margins to be sufficiently reliable, and therefore the optimality is reduced.

The idea behind the use of supervised learning consists of the fact that in theory a learning agent could be taught how to approximate the optimal solution at any point in space, and do so very fast. Such an optimal trajectory is usually obtained in deterministic environments, and thus the so obtained trajectory may not be optimal in a stochastic environment. RL can instead provide a solution that is optimal even when considering all the possible future perturbations. In addition, artificial intelligence methods could compute a command in a fraction of the time needed by the NPC. This final reasoning is what motivated the use of artificial intelligence solutions to the problem of aerocapture guidance.

In the remaining of this introductory chapter the research question of this thesis is formulated; after that, the outline for the subsequent chapters is given.

1.1 RESEARCH QUESTION

The entire research is focused on the guidance logic for aerocapture, an atmospheric maneuver that, although being highly convenient, and possibly a game-changing technology in space exploration, has never been attempted before. The lack of a reliable and performing enough guidance system is a reason why an attempt never occurred. From the previous elaboration, using artificial intelligence seems a good candidate for the problem. Hence, a question arises, specifically:

> *Can an artificial intelligence guide a spacecraft to achieve aerocapture in an optimal and robust way?*

While the adjective robust is clear, and means "*able to withstand or overcome adverse condi-*

tions"[2], optimality is a more ambiguous concept, and very relative. A clear, and quantifiable way of defining *optimal and robust* is given in Section 2.7.

A subquestion that would help answering the main question is:

What are the most useful parameters to evaluate optimality in aerocapture?

It has also been said that RL is a good solution for highly perturbed environments. Hence, for the research to be consistent, one should also answer the following subquestion:

How does the state-of-the-art guidance solution perform in a highly perturbed atmosphere?

While answering these questions, some modifications to the current state-of-the-art are also proposed. The main question is very vast, and therefore this research will be limited to using only two kinds of artificial intelligence solutions among the large variety of existing ones.

1.2 REPORT OUTLINE

The subquestions will be the topic of Chapter 4, whereas the main question will be answered in Chapter 7.

Chapters 2 and 3 can be considered as introductory. The former deals with previous missions in which aerocapture was planned, or that made use of maneuvers similar to it. It also includes detailed motivations for studying aerocapture, as well as an overview of the maneuver, the mission requirements, and the problem statement. The latter treats the dynamics of hypersonic entry, and the corresponding environment, forces, and perturbations.

The goal of Chapter 4 is indeed that of answering the two subquestions. This will be done by analyzing the requirements, constraints, and optimal trajectories of aerocapture. A few theoretical implications that are, at the best of the author's knowledge, novel, will also be derived, specifically in the Subsections 4.3.2 and 4.3.3. Also, the guidance by Lu et al. [2015] will be reproduced, modified in different ways, and tested in highly perturbed environments. The test setup will be described in Chapter 6. Chapter 4 will then include some partial conclusions concerning aerocapture and traditional NPC guidance methods.

Chapter 5 gives a synthesis of the foundations of artificial intelligence. The chapter involves machine learning, deep learning, and reinforcement learning. It is complemented with a few examples of previous researches that made use of artificial intelligence for Solar System exploration. To make the subsequent chapter lighter, verifications of training algorithms is included here.

Chapter 6 describes not only the experimental setup used for this research, but any form of software tools that were developed during this research. It also reports in detail any verification or validation that required more than a manual or visual check.

Chapter 7 describes the results of this research. The amount of data produced, saved, and stored as "results" goes beyond 50,000 trajectories, guided by tens of different neural guidances through six different environments. It goes without saying that not all of the results will be reported, and many will not even be mentioned. However, this reduction is not very difficult to make, considering that many sets of trajectories turned out being unfortunate at best.

Finally, Chapter 8 concludes this research, and includes recommendations to whoever considers diving into the world of aerocapture and artificial intelligence for Solar System exploration.

[2] As defined by the Oxford Dictionaries.

To avoid any ambiguity in the remaining of the report, when referring to target or final orbit, it is meant the orbit that would be obtained after periapsis raise and corrections. The exit orbit, or exit orbital leg, would be the exoatmospheric part of the flight before periapsis raise. Descending and ascending legs are definitions limited to the atmospheric part of the flight instead.

2

MISSION OVERVIEW

As stated in Chapter 1, an aerocapture maneuver has never been attempted before. However, many missions making use of it were, and currently are, being studied. Also, similar missions, namely skipping entry and aerobraking, have succeeded in the past. In this chapter, previous skipping and aerobraking missions are reviewed. After that, missions for which aerocapture was planned are presented, to give an idea of which are the main possibilities for the use of an aerocapture. Before all this, however, detailed motivations for why aerocapture may be beneficial to Solar System exploration are given in the first section.

The goal of this chapter is that of putting the aerocapture maneuver into context. First, an insight of what would be the most interesting reference mission to use, as well as the reference vehicle, according to real cases in which aerocapture has been proposed. Reference vehicles and missions are then set in Section 2.6. Then, some characteristics of the aerocapture trajectories are shown.

Eventually, the mission requirements are drawn, together with the optimal aerocapture guidance problem statement. Any design choice concerning the guidance that will be made throughout this report will be done with those requirements and that problem statement in mind.

2.1 AEROCAPTURE FOR SOLAR SYSTEM EXPLORATION

The aerocapture maneuver is among one of the main current goals of NASA. Many efforts are currently being done by both the Flaghsip Technology Demonstrations (FTD) Team and the In-Space Propulsion Technologies (ISPT) in the U.S.A. In Europe, the Aerofast project recently put big efforts in aerocapture research.

The reason for studying and focusing on aerocapture is twofold: first, it has the potential to greatly reduce the mass, and hence the cost, of many missions to any of the Solar System bodies that have an atmosphere; second, it might enable missions otherwise impossible with the current state of technology, such as a mission to Neptune.

Aerocapture may be greatly beneficial simply because of Tsiolkovsky's famous rocket equation, which implies that the required propellant-mass fraction for an entirely propulsive capture increases exponentially with the ΔV. Therefore, any propulsive maneuver exponentially increases the mass at launch, which is an important factor of the mission cost. The mass fraction of the aeroshell when attempting an aerocapture maneuver instead increases quasi-linearly with the ΔV, as displayed in Figure 2.1 (despite the rule is empirical). Hall et al. [2005] point out the fact that the aeroshell mass also depends on the entry velocity: as an example, an elliptical capture on Jupiter only requires a ΔV of $1.4\,\mathrm{km\,s^{-1}}$. In such case, because of the very high entry velocity, the mass fraction of the aeroshell needed for an aerocapture would be very large.

Benefits of aerocapture and their impact in Solar System exploration have been studied by Hall et al. [2005]. In their work, they showed how such a technology would, in most cases, improve con-

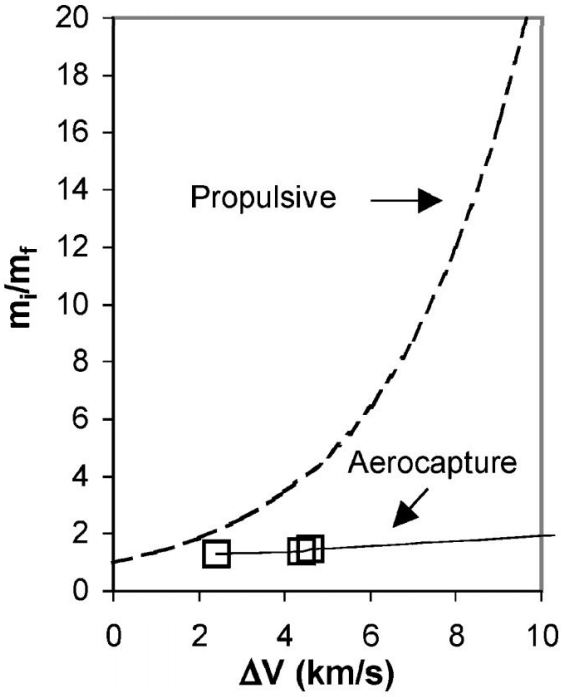


Figure 2.1: Comparison of aerocapture to propulsive orbit insertion mass [Hall et al., 2005].

sistently the available payload mass of a scientific mission with respect to many other alternatives. In addition, they showed that, given the current state of technology, the only viable options for some missions (such as a circular orbit around Neptune) would be that of using aerocapture.

In their work, they also showed that aerocapture gives the best performance when one wants to insert into a circular orbit rather than into an elliptical one. For this reason, aerocapture for elliptical orbit insertion will never be considered in this research.

As an example, the payload for a mission to Mars would be increased by a factor of 1.15 with respect to aerobraking and by a factor of 1.63 with respect to a 370 s specific impulse chemical propulsion capture. In this case, the benefit of aerocapture is not very relevant if compared to aerobraking. However, it should be considered that, in situations where the duration of the maneuver is relevant, as, for example, a manned mission, aerobraking becomes very disadvantageous, if not impractical. Thus, for that kind of mission, the choice of an aerocapture would allow for great benefits also with respect to aerobraking.

Other situations give much larger improvements: for a mission to Venus, the capability is increased by 79 % with respect to the best alternative option; for a mission to Uranus, the capability is increased by 218 %, and for a mission to Titan the increase is 280 %.

Of large interest is also the fact that aerocapture has been chosen by NASA as a reference approach for landing high mass (between 20 and 80 t) cargo on Mars, despite being considered impractical for the landing of manned spacecraft [Drake et al., 2010]. Moreover, when the EDL-SA study was developed, the most efficient strategy for high-mass payload landing on Mars included an initial aerocapture; such a strategy was architecture 2 [Ciancolo et al., 2011].

2.2 SIMILAR FLOWN MISSIONS: SKIPPING ENTRY AND AEROBRAKING

Skipping entry is similar to aerocapture in the sense that the first skip usually turns a hyperbolic orbit into an elliptical one. Similarity with aerobraking consists instead only the fact that the atmosphere is used to brake, while still remaining in orbit.

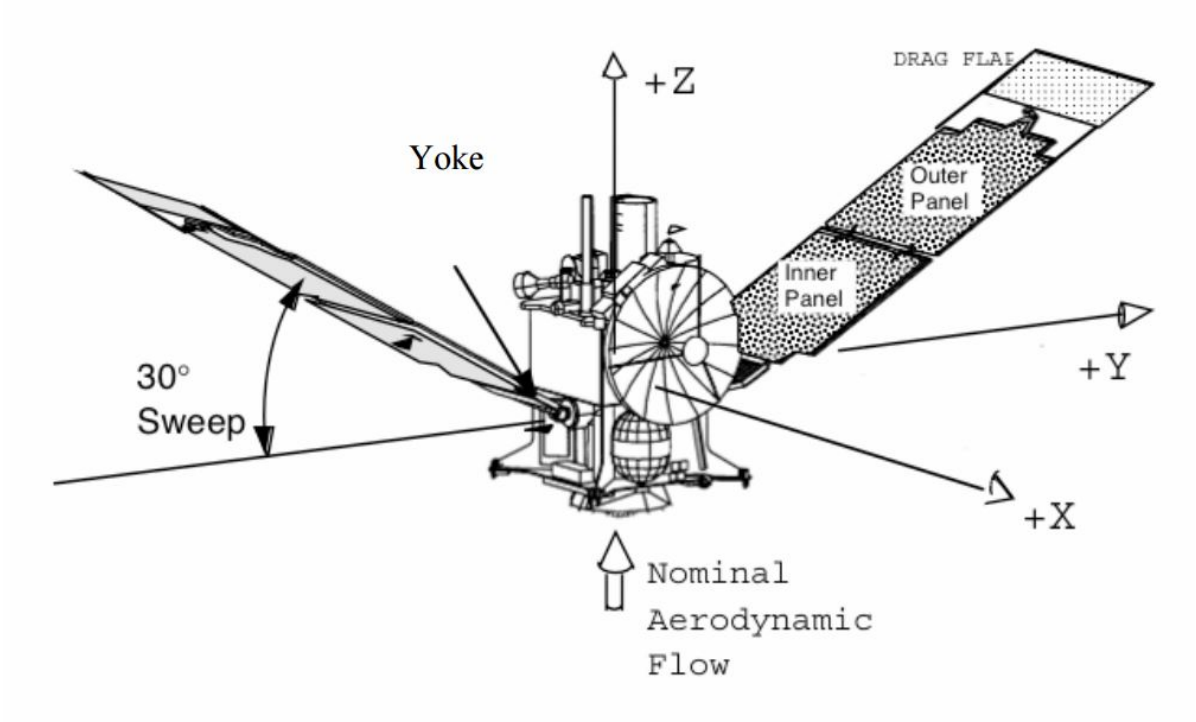


Figure 2.2: Mars Global Surveyor (MGS) in aerobraking configuration [Spencer and Tolson, 2007].

Skipping entry was performed three times only, by the Russian Zond-6, Zond-7 and Zond-8, in, respectively, 1968, 1969 and 1970. An earlier mission, Zond-5, also attempted a skip entry, while carrying a biological payload, that included two survivor tortoises. The guidance of Apollo was also capable of skip entry, but never performed it, because the risks were considered too high for a manned mission. An error could have either led to a crash or to a too long skip, which would have caused the death of the crew due to lack of oxygen. A large part of the heritage for aerocapture is, nevertheless, considered to come from Apollo, including the first guidance system that was ever developed Gamble et al. [1988], which is an adaptation of the Apollo skip entry guidance.

Aerobraking has been achieved for the first time in 1991 during the Japanese Hiten mission [Uesugi, 1996]. The spacecraft was in a very elliptical orbit around the Earth, and during its first two passes inside the atmosphere it braked of $1.7\,\mathrm{m\,s^{-1}}$ and $2.8\,\mathrm{m\,s^{-1}}$, reducing its apogee of a total of 22 500 km, with respect to its initial value.

During the Magellan Mission to Venus, for the first time aerobraking was successful on an extraterrestrial planet. The Magellan spacecraft reached the planet on August 10, 1990, and was propulsively captured in a highly eccentric orbit. Its goals included obtaining near-global radar images of Venus' surface, topographic map and gravity field data. On May 25, 1993, after a mission extension was approved to obtain better gravity field data, the spacecraft started dipping into Venus' upper atmosphere to decrease the energy of the orbit; when aerobraking was complete (August 3, 1993), around 70 days later and after 700 atmospheric passes [Tolson et al., 2013], the orbital period was reduced from the initial 3.26 hours to 1.5 hours, reducing the velocity of the orbit by $1200\,\mathrm{m\,s^{-1}}$ [Lyons, 2000]. The attempt was successful, in spite of the fact that the spacecraft was not designed for such a maneuver. Magellan was then able to circularize its orbit with an almost propellant-free maneuver. Although the vehicle was not designed with aerobraking in mind, the aerobraking software proved to be very efficient for autonomous operations.

The first aerobraking on Mars was achieved by the MGS, which was propulsively captured on September 12, 1997. After that, it started an aerobraking maneuver to lower its apoapsis. During that phase, the period of the orbit was reduced from the initial 45 hours after capture to 1.86 hours, and the orbit was slowed down by about 1200 $\mathrm{m\,s^{-1}}$ [Lyons, 2000]. Figure 2.2 shows the configuration of the spacecraft during aerobraking. The panels were used with as additional decelerating surface; a sweep angle was planned to reduce the structural stress.

The maneuver should have lasted four months only, but because of a failure in the latch of one of the solar panels it had to be rescheduled; the final orbit was achieved only on February 1999. Additional problems faced during the MGS mission include the fact that Mars is often subject to dust storms. Although those occur at lower altitudes (around 50 km), they can affect the atmospheric density at even higher altitudes, where aerobraking occurs. According to Tolson et al. [2013], differences in density from orbit to orbit were up to 40 % of one standard deviation during the mission. This meant that the operations team had to be ready to raise the periapsis of the orbit with very short notice in case a dust storm occurred.

The first failure, and last attempt to date, of aerobraking, is that of the Mars Climate Orbiter. The goal was investigating the Martian atmosphere and its interactions with the surface. The vehicle reached Mars on September 23, 1999, when a propulsive capture was scheduled to insert the spacecraft into an elliptical orbit whose periapsis was meant to be inside the atmosphere. Aerobraking would have circularized the orbit afterwards. The signal was lost when Mars occultation began, 49 s earlier than predicted, and was then never reacquired. A ΔV was computed to have a periapsis altitude of 110 km, higher than the minimum 80 km altitude which were necessary for the spacecraft to survive. After subsequent investigation, it was found that the vehicle was inserted into an orbit with a periapsis altitude of 57 km, which caused the disintegration of the spacecraft. This error occurred because of the use of non-SI units on the input data for the computation of the ΔV [NASA, 1999].

2.3 AEROCAPTURE IN PROPOSED MISSIONS

Aerocapture has been proposed for a few different missions. Those usually consisted of sample-return missions. However, it is very likely that a first aerocapture maneuver will be a demonstration on Earth, that would enable to increase the TRL of the maneuver significantly [Munk and Moon, 2008; Percy et al., 2005]. Within the reasons why aerocapture has never been attempted before is the fact that a demonstration has not been accomplished yet, due to the high costs that it would imply. Therefore, the risks of an aerocapture are still unacceptable, and the maneuver has been so far preferably avoided. In addition to sample-return missions, aerocapture has also been considered for possible missions to Neptune [Lockwood, 2004], because it is currently the only means by which that planet could be orbited.

2.3.1 MARS SAMPLE RETURN MISSION

A Mars Sample Return (MSR) mission has been studied for many years. It is considered, among the unmanned missions, to be the single mission with the highest scientific return. The possibility of studying a sample of Mars on Earth would lead to much more scientific data than any mission before, because the analysis would not be constrained to the instruments carried on board of a lander. Being Mars the most Earth-like planet in our Solar System, analysis of samples from there would give a large amount of information about our planet. It might also clarify the question concerning the existence of life, now or in the past, on the Red Planet. For these reasons, the MSR has been continuously assigned a high priority by the U.S. National Research Council (NRC); it has, however, never been planned yet. The main problems consist of the fact that it requires many new technologies and that, for the mission to have the most possible scientific return, the samples have to be properly selected. The studies for this mission have always been carried out by international teams, because of the high costs that it would have. One of those teams was formed by NASA and the Centre

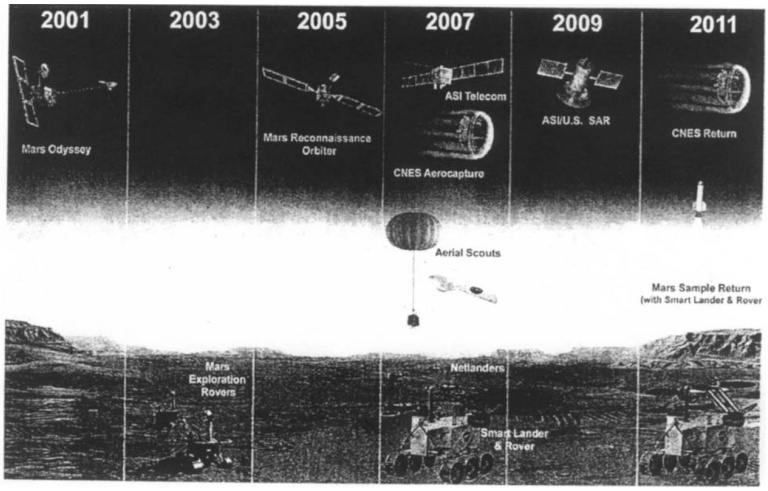


Figure 2.3: Mars Sample Return mission queue [Cazaux et al., 2004].

National d'Études Spatiales (CNES) in France. Cazaux et al. [2004] stated the main technological developments and challenges that such a mission would require; among those was also aerocapture. A trade-off was done among many capture methods for orbiting Mars, and aerocapture turned out to be the most efficient way, because of the large weight of the orbiter, which had to carry on-board also the propellant to later leave Mars' orbit.

The original planning included the launch of three vehicles (one lander, one rover, and one orbiter) with two launchers, in 2003 and in 2005 [O'Neil and Cazaux, 2000]. The samples would have returned to Earth in 2008. The various phases of the mission can be consulted in Figure 2.3. The first launch would have been operated by NASA, and have sent to the Red Planet a Lander carrying a Rover and the Mars Ascent Vehicle, which would have come to destination in December 2003. In August 2005, an Ariane 5 launcher would have lift off with a similar payload to the first one, together with an Orbiter supplied by CNES and including a NASA-JPL rendezvous and capture payload and NASA-Langley Research Center (LaRC) Earth entry vehicle. When arriving to Mars, the Orbiter would have achieved to first aerocapture in history, crossing the Martian atmosphere to be captured without the use of a propulsive system. After aerocapture, the spacecraft would have had an elliptical orbit with apoapsis altitude of 1500 km. There it would have used rockets to raise the periapsis to 200 km. The orbit would have then been circularized and the Orbiter would have started the search for the mission sample containers, that would have been put on orbit by the respective Mars Ascent Vehicles. After the recovery of both samples, the Orbiter would have then departed from the Red Planet and entered an orbit that would have led it to Earth. It is interesting to notice that despite the use of aerocapture saved the orbiter around $2\,\mathrm{km\,s^{-1}}$ of ΔV, a total ΔV of $3.5\,\mathrm{km\,s^{-1}}$ for periapsis raise, circularization, maneuvering and departure would still be needed. Then, direct reentry would occur on Earth, with an initial velocity of $11.5\,\mathrm{km\,s^{-1}}$.

A more recent planning [iMars Working Group, 2008] for an international Mars Architecture Sample Return Mission (iMARS), to which both European Space Agency (ESA) and NASA took part, concluded instead that an aerocapture would not be necessary, because the Mars Science Laboratory (MSL) heritage would be sufficient in that case. It would therefore be preferred to use a less fuel efficient but safer option than aerocapture.

2.3.2 COMET-NUCLEUS SAMPLE RETURN MISSION ROSETTA

The original goal of the Rosetta mission, as it was planned back in the 1990s, was that of returning about 10 kg of samples from the comet nucleus: the mission was indeed initially called Comet Nu-

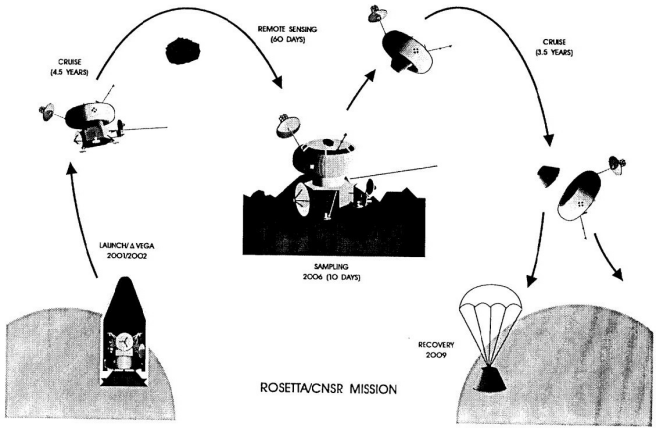


Figure 2.4: Artist's impression of the Rosetta/CNSR mission [Schwehm, 1989].

cleus Sample Return (CNSR). In view of this sample return, aerocapture had been chosen as a possible option for the capture of the vehicle in Earth orbit. Such a mission would have provided large scientific return because the possibility of analyzing comet samples would have been determining for the knowledge of the Solar System primordial nebula [Serrano-Martinez and Hechler, 1989]. In this case the samples would have rendezvoused with the International Space Station (ISS) after the aerocapture, and later been brought to Earth's surface for scientific analysis. The sample return was however discarded, because of the difficulty in keeping the samples at the required temperature for a long time in the ISS.

The mission phases are given in a schematic way in Figure 2.4. Those would have been the following: launch with a Titan/Centaur, Earth gravity-assist, transfer to comet, rendezvous with the comet at aphelion, at around 5 AU, descent and landing on comet, sampling, departure, and return to Earth with approach with V_∞ of around $10\,\mathrm{km\,s^{-1}}$. Afterwards, aerocapture would have begun on the upper layers of Earth's atmosphere. The aerocapture was planned to have an initial flight path angle of $-10.5°$ and initial velocity of about $15\,\mathrm{km\,s^{-1}}$. According to Serrano-Martinez and Hechler [1989], a guidance and navigation system was developed that allowed the atmospheric phase to have accelerations below 20 g with a capsule-like vehicle.

2.4 EXISTING GUIDANCE ALGORITHMS

The four most common kinds of aerocapture guidance algorithms that can be found in literature are: APC, NPC, TPC and EC. In 2002, a joint CNES-NASA team ran a simulation campaign for a comparison of those algorithms, the results of which were published in Rousseau et al. [2002]. The goal was to chose the best candidate for the Mars Premier mission. The four algorithms were therefore tested using a simulator, and evaluated according to four main criteria: accuracy, robustness, loads control and complexity. In all of those algorithms, the controlled variable is the bank angle.

The APC divides the aerocapture in two phases: the entry phase, during which the spacecraft simply tries to lose as much energy as possible without violating any of the loads constraints, and the exit phase, during which the vehicle is guided to obtain an orbit that is the closest possible to the required one. This algorithm makes use of analytical predictions for the exit phase; specifically, those predictions rely on the assumption of constant altitude rate [Cerimele and Gamble, 1985]. Modified and newer version of this framework include the work by Masciarelli et al. [2000] and Hamel and Lafontaine [2006]. The differences are small, and often consist of better defining the transition conditions, and making the algorithm more flexible. The NPC also divides the aerocapture in two phases [Powell and Braun, 1993]. However, the prediction is done using numerical

methods and simplified equations of motion, and during the exit phase, the constant bank angle needed to have the right apoapsis altitude is computed at each guidance call. Different modifications of this algorithm exist, the main one being the use of a single phase at constant bank angle, as one of the various modes in Lafleur [2011]. The EC uses analytical simplifications as the APC does. In this case, the analytical assumption is used to compute the final energy state at exit. It then uses the error between predicted and target energy state, together with its derivative, to compute an energy gain and, consequently, the commanded angle [Rousseau, 2001]. At last, the TPC uses a stored trajectory, and then analytically computes proper gains to track it. The gains are influence coefficients obtained with calculus of variations, and are computed so that the vehicle shall reach the predefined terminal state at the final time [Ro and Queen, 1998].

It turned out that all the algorithms had the capability of achieving a successful aerocapture; among those, the TPC had the best general results, despite the bad performance in the scenario including dust storm and the lack of a density scale height estimator.

A more recent guidance schemes is the hybrid predictor-corrector aerocapture scheme (HYPAS) [Masciarelli and Queen, 2003], a semi-analytic predictor corrector that flies a constant altitude rate exit phase, and its evolution as a drag-tracking scheme [Casoliva et al., 2008]. However, not too many efforts have been done in the direction of improving performance. By improving performance, it is usually meant trying to minimize the need of propellant for orbit circularization. In this sense, the PredGuid+A [Lafleur, 2011] can be considered to be one of the first guidance algorithms focusing on performance as well. Another algorithm that aims to optimality is the one developed by Lu et al. [2015].

Strictly related to this research, is the work done by Gelly and Vernis [2009], who developed a neural guidance for aerocapture using RL. Their guidance was compared to a feedback trajectory control (FTC) guidance scheme, and gave better results in accuracy, loads constraints and bank angle consumption. More insight into their work will be provided in Subsection 5.3.1. At the best of the author's knowledge, supervised learning has not been yet applied to aerocapture. Hence, in that same subsection, the work by Sanchez-Sanchez et al. [2016] is shown, which consists of supervised learning for optimal terminal landing.

2.5 AEROCAPTURE MANEUVER OVERVIEW

In this section, a qualitative overview of the aerocapture maneuver is given.

During aerocapture, orbital energy of a vehicle generally goes from positive to negative[1]. This happens with a single atmospheric pass. Once the vehicle exits the atmosphere[2], it enters an elliptical orbit with periapsis inside the atmosphere. Hence, once the apoapsis of the orbit is reached, a relatively small thrust burn is given. This way, the periapsis is raised above the edge of the atmosphere, to the desired value. If necessary, a second burn is applied half an orbital period later, to correct any possible errors in the apoapsis. Errors in inclination are also corrected, usually during the first of the two burns. In general, the atmospheric part of the flight is divided in two phases: a descending leg, during which the loads on the vehicle are largest, and an ascending leg, which usually lasts much longer. Two interesting cases of aerocapture occur when the bank-angle is saturated, and are described in the next subsections.

2.5.1 SATURATED TRAJECTORIES

Figures 2.5 to 2.8 show some of the most interesting information about two aerocapture trajectories on Earth, both with initial relative velocity of $13\,\mathrm{km\,s^{-1}}$, both targeting an apoapsis altitude of

[1] This is not entirely true for a Lunar return case, which begins with slightly negative orbital energy instead. The word aerocapture can still be used for this case though, since the vehicle is captured into the orbit of the Earth.

[2] The vehicle does not exit the atmosphere. For altitudes larger than a certain value, usually between 100 km and 120 km on Earth, the effect of the atmosphere on the trajectory becomes negligible.

200 km. One trajectory is flown with full lift-up, and the other one with full lift-down. The initial flight-path angles were found using the software described in Section 6.5 and are, respectively, -8.7156° and -5.7695°. Both trajectories are flown by the Apollo Command Module, whose data is provided in Table 3.1 of Subsection 3.5.4.

Figure 2.5 describes the altitude - time profile. The full lift-down trajectories last six times longer than the full lift-up trajectories. Both trajectories fly the ascending leg at an approximately constant altitude rate. The assumption of constant altitude rate is an important assumption in the previously mentioned APCs guidance concepts. In both cases, the ascending leg is the one that lasts most of the time. The lift-down trajectory reaches a minimum altitude of 65 km, while the lift-up trajectory goes deeper into the atmosphere, to altitudes of 48 km.

Combining this figure with the previously described one, it can be deduced that most of the energy for the lift-up trajectory is depleted in about 30 s. This has major implications for the simulation of the attitude dynamics, since large attitude maneuvers have durations of that order of magnitude.

Figure 2.6 shows the altitude - velocity profile. In addition to what was said before, it is seen that the minimum altitude in the lift-up trajectory is reached at a velocity already as low as 9.5 km, whereas it is reached with a velocity of 11 km for a lift-down trajectory. This means that most of the energy is depleted during the descending leg for a lift-up trajectory, and that the opposite occurs in a lift-down trajectory. This plot is interesting also because it shows that the final velocity of the lift-down trajectory is much larger (by about $800\,\mathrm{m\,s^{-1}}$) than the final velocity of the lift-up trajectory, despite both aiming to the same apoapsis altitude. More about this will be said in Section 4.1. Combining this

Figure 2.7 shows the trajectories in the energy - dynamic pressure plane. Such a plane is very interesting because the dynamic pressure is strictly related to the load factor. In addition, it shows the aerocapture corridor: any trajectory starting with the same initial velocity as that of the boundary trajectories, and aiming to the same apoapsis altitude, cannot have a single point in this plane outside of the area enclosed by the lift-down and the lift-up trajectories. In Section 4.3.3 it will be shown how in such a plane there is, for every point, a range of allowed flight-path angles: this describes the volume of all possible states for which an aerocapture can occur, for fixed initial velocities and apoapsis altitude. The maximum dynamic pressure for a lift-down trajectory is 6.5 times smaller than that for a lift-up trajectory. In addition, it is seen that the final energy of the lift-down trajectory is $5.5\,\mathrm{MJ\,kg^{-1}}$ than its lift-up counterparts.

Eventually, Figure 2.8 shows the two trajectories in the flight-path angle - velocity plane. In particular, it shows how the flight-path angle of the full lift-down trajectory is almost zero during the entire trajectory, and specifically at the end of it. The lift-up trajectory has instead an ever increasing flight-path angle.

This preliminary discussion showed already two benefits of the lift-down trajectory, namely the much smaller dynamic pressure peak and the larger final velocity. Intuitively, one may think that a larger final velocity will lead to a smaller final ΔV needed to raise the periapsis. Such fact will be proven in Section 4.1. In the next subsection some motivations will be given to why, in reality, it is not advised to fly a full lift-down trajectory.

2.5.2 STABILITY AND SENSITIVITY OF AEROCAPTURE TRAJECTORIES

Before designing a guidance, it is important to realize how stable and sensitive to the initial conditions the aerocapture trajectory is.

In a re-entry problem two of the most important parameters are the ballistic coefficient and the lift-to-drag ratio. Gurley [1993] analyses how these two parameters affect an aerocapture maneuver on Mars, with target apoapsis of 37 400 km, and how they affect the altitude, velocity, and acceleration history. In their work, when talking about the lift-to-drag ratio, the lift is meant as its component in the vertical plane, $L\cos\sigma$. In his analysis, the simplified equations of motion for spherical

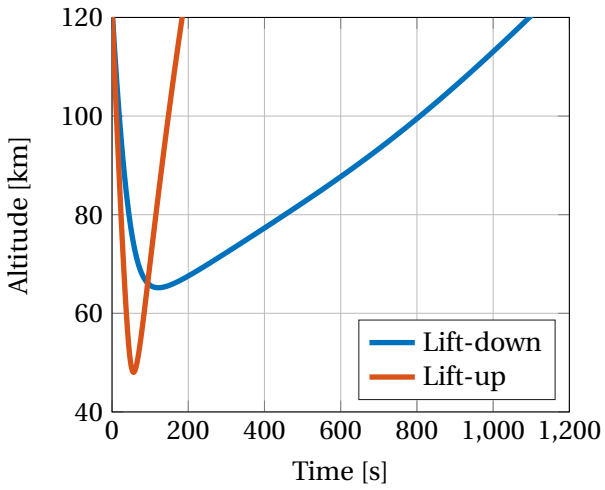


Figure 2.5: Full lift-up and full-lift down trajectories for $h^{\star}_{apo} = 200\,\text{km}$ in the $V - t$ plane.

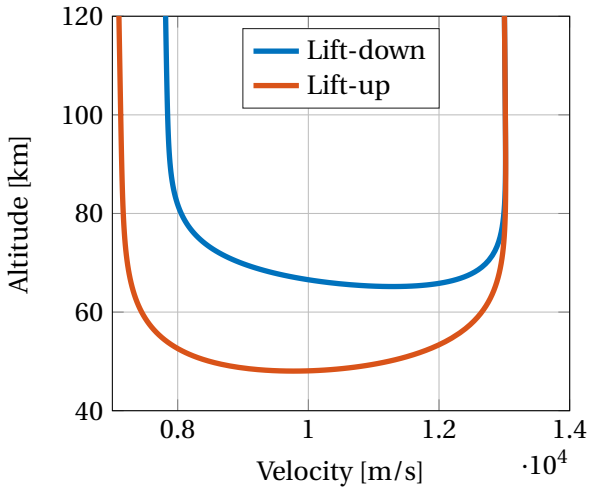


Figure 2.6: Full lift-up and full-lift down trajectories for $h^{\star}_{apo} = 200\,\text{km}$ in the $V - h$ plane.

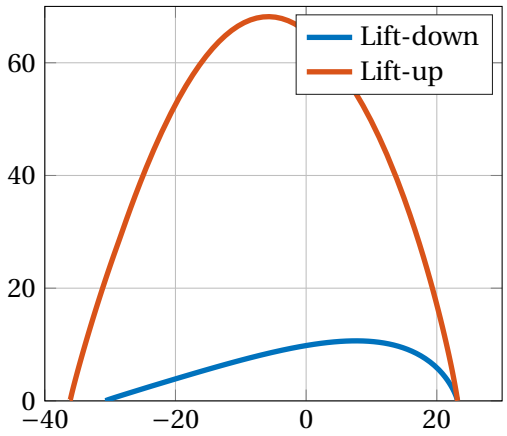


Figure 2.7: Full lift-up and full-lift down trajectories for $h^{\star}_{apo} = 200\,\text{km}$ in the $E - \bar{q}$ plane.

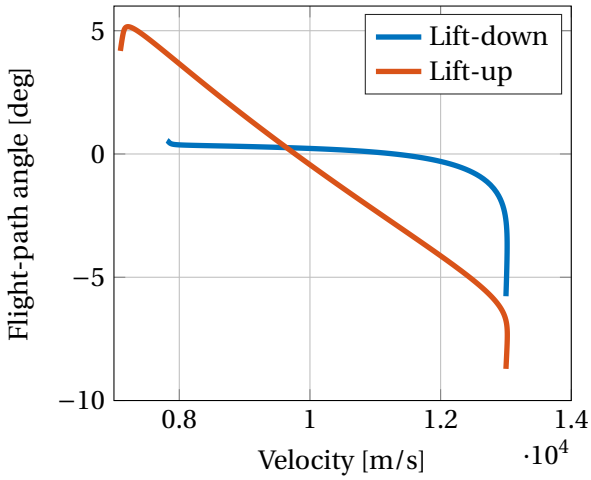


Figure 2.8: Full lift-up and full-lift down trajectories for $h^{\star}_{apo} = 200\,\text{km}$ in the $\gamma - V$ plane.

non-rotating planet, with exponential isothermal atmosphere, and planar flight are used.

They show that despite a large negative lift-to-drag is very convenient for having a small load factor peak (as well as for reducing the final required ΔV, as shown in Section 4.3), it also makes the maneuver very unstable. In fact for a lift-to-drag ratio equal to -1.5, it is required to have an accuracy of around 1 mm at the aimpoint (the point that is aimed before aerocapture) to have an error in the apoapsis of not more than $\pm 200\,\text{km}$. When having a positive lift-to-drag ratio of 0.5, an accuracy of 36 m is instead sufficient for the same error in apoapsis. This is, of course, in case of absence of deterministic conditions.

Hence, for same entry velocities, a full lift-down trajectory is much more sensitive to initial conditions than a full lift-up one.

2.6 REFERENCE MISSION

So far, the main cases in which aerocapture has been proposed are:

1. Earth aerocapture at Mars return conditions (as in the MSR mission)

2. Earth aerocapture at comet return conditions (as in the CNSR mission)

3. Mars aerocapture at Earth return conditions (as in the MSR mission)

However, the most likely first attempt of aerocapture would be a technology demonstrator. The "easiest" way to do so would be having a lunar return trajectory. These are the same conditions used in Lu et al. [2015]. Hence, the reference mission conditions will be taken from there, to facilitate any comparison.

In their work, the Orion Multi-Purpose Crew Vehicle (MPCV) was used. Nonetheless, in this research, the reference vehicle chosen is the Apollo Command Module. The aerodynamic data, reference surface, as well as reference mass are found in NASA [1965]. The main difference between the Orion MPCV and the Apollo Command Module is the nominal lift-to-drag ratio, about 15 % larger for the latter.

More details about initial conditions, perturbations, and dispersions are given later, in Table 4.1 of Section 4.8. At the moment, it is sufficient to know that the simulated trajectory will be a lunar return.

2.7 MISSION REQUIREMENTS AND PROBLEM STATEMENT

Now that a sufficient context has been given, the mission requirements and problem statement can be given. On a high level, every space mission is designed to minimize the cost and maximize the chances of success. Since, in most cases, it is impossible to have the certainty of success, a success rate of 3 standard deviations will be considered acceptable. This means that, on a Monte Carlo run of 1,000 trajectories, no more than three failures can occur.

The Monte Carlo runs will be simulated with a wide range of initial flight-path angles, spanning about 1.5°. It is now assumed that a good approach planning can lead to an accuracy in entry flight-path angle of 0.25°. Hence, it is possible to redefine robustness, and require to have a range of possible angle entries[3] of 0.5°, in which only one failure occurs during one Monte Carlo run[4].

The guidance will be considered successful if it gives commands such that, given ideal control, navigation and actuators, the trajectory meets all the requirements.

Hence, it is better to start with the requirements for the trajectory that will be actually flown. which are given in the frame below. Most of the requirements can better be understood after having read Chapter 3.

TRJ-1	The trajectory shall lead to an exit orbit such that the periapsis raise would be smaller than ΔV_{max}.
TRJ-2	The trajectory shall have a maximum heat rate peak smaller than $\dot{q}_{max}$, as well as a maximum load factor smaller than n_{max}.
TRJ-3	The trajectory shall have a total heat-load smaller than Q_{max}.

At this stage of the research, it is early to quantify ΔV_{max} and Q_{max}, since these are design parameters, which may be dependent on the capability of the guidance itself. Hence, at the moment, it will still be attempted to minimize these two parameters. ΔV_{max} can be given an upper bound. A propulsive capture from lunar return conditions would require a ΔV of approximately $3.000\,\mathrm{m\,s^{-1}}$.

[3] At lunar entry conditions, an accuracy of 0.5° in entry angle equals to an accuracy of about $100\,\mathrm{m\,s^{-1}}$ in vertical velocity.

[4] In one of the previously mentioned Monte Carlo runs, 333 trajectories occur within a range of 0.5°. Statistically speaking, at this point, a sample of about 333 trajectories within such a range of entry angles may be a little small.

Since an aerocapture would require a heat-shield, and is, at the moment, less reliable than a propelled capture, the upper bound for ΔV will be set at[5] $1.000\,\mathrm{m\,s^{-1}}$.

Concerning TRJ-2, upper bounds can also be given. An ablator such as the phenolic impregnated carbon ablator (PICA) can withstand heat fluxes as large as $18\,\mathrm{MW\,m^{-2}}$ [Willcockson, 1999]. This value will be used as constraint on $\dot{q}_{max}$. Reasonable values for n_{max} range between 5 and 25, depending on the mission. Aerocapture had been planned in the past for sample return missions. For these cases, the load factor constraints would be the same as that of the spacecraft[6]. Aerocapture has also been considered for manned missions though, in which the constraint on the load factor may be required to be as small as 5 [Lyne, 1994][7]. For this reason, the maximum load factor will also be set as an objective function to be minimized. The smaller the maximum load factor, the wider would be the applicability of aerocapture.

After having more in detail analyzed the mission, all the limit values will be set. At this point, the mission is designed in such a way that all the trajectory requirements will have to be met during flight. Hence, given the considerations of the beginning of this subsection, the requirement for the guidance logic can be specified:

GD-1 The guidance shall lead to a trajectory that respects all requirements TRJ-1 to TRJ-3, for a wide range of initial conditions and perturbations. Specifically, it shall do so for a range of initial flight-path angles of at least 0.5°.

The guidance is a major component in a trajectory design: hence, the limit values for ΔV_{max} and Q_{max} will depend on the performance of the guidance. Specifically, they will be equal to[8] the maximum values encountered in such a range of entry conditions, using a certain guidance logic. The range will be chosen such that the maxima are minimized. Thus, the problem of the guidance can be reformulated in terms of a multi-objective, constrained, optimization in stochastic environments:

Optimal Aerocapture Guidance Problem Statement

The optimal aerocapture guidance logic aims to guide the vehicle along trajectories in a way such that the maximum expected values of the quantities mentioned in the requirements TRJ-1 to TRJ-3 are minimized. This has to be be valid for a variety of perturbations and initial conditions; specifically, for a reasonably large range of entry angles, and for at least 99.7 % of the cases.

As previously mentioned, such a range of entry angles is required to be at least 0.5° wide.

The parameters will the be minimized: nonetheless, if a range in which one of the following conditions cannot be ensured, the aerocapture will be declared unfeasible, or, at the very least, inconvenient with respect to other solutions. These minimum conditions are, for the mission and vehicle in object:

[5] This value is an inintial guess. Properly estimating a more accurate value of ΔV above which aerocapture is not convenient may require a very demanding and multidisciplinary analysis.

[6] The payload would have been a rock, whose limits to load factors are probably much larger than that of the vehicle

[7] In that work, Lyne [1994] specifies that such a requirement is needed after long duration flights, such as a mission to Mars. Otherwise, the load factor limit can as well be larger.

[8] In reality, safety factor shall be used, but these will not be considered here.

1. $\Delta V_{max} < 1000\,\mathrm{m\,s^{-1}}$;
2. $\dot{q}_{max} < 18\,\mathrm{MW\,m^{-2}}$;
3. $n < 25$,

The problem statement puts into context and makes quantifiable the adjectives "*optimal and robust*" used in the main research question of Chapter 1. *Optimal* is then defined as a minimization of the trajectory parameters of requirements TRJ-1 to TRJ-3. For the guidance to be *robust*, the parameters have to be minimized for the worst case scenario[9].

To give an example, a hypothetical guidance that always leads to the same ΔV of $400\,\mathrm{m\,s^{-1}}$ will be preferred to another hypothetical guidance that leads to an average ΔV of $100\,\mathrm{m\,s^{-1}}$, but with a 1 % of situations for which the final ΔV is $500\,\mathrm{m\,s^{-1}}$.

The problem statement will have to be kept in mind both when solving the guidance problem, both when using numerical solutions or artificial intelligence solutions.

An additional requirement of the guidance consists of being able to give the commands in real-time. There is no attempt at quantifying such a requirement during this research. Nevertheless, methods that give equal or similar performance, but are clearly less computationally demanding, will be preferred.

As a last remark, the guidance logic is strictly entangled with the navigation system, from which it obtains its inputs, and with the attitude controller, which actuates the commands given by the guidance. To properly analyze the guidance, it will be necessary to have a test setup such these three subsystems can be, as much as possible, disentangled. Such a discussion will be resumed in Chapter 6.

2.8 SUMMARY

Aerocapture can be greatly beneficial in Solar System exploration, and has therefore been proposed in a few missions. In most cases, these concerned sample returns. It has also been considered a relevant option for many architectures of the NASA EDL-SA. Nonetheless, it has never been attempted. Moreover, aerobraking, a less risky maneuver on paper, has a very low success rate, because of the Mars Climate Orbiter crash.

Eventually, it is believed that the most likely first aerocapture will probably be flown as a technology demonstrator on Earth, at lunar return conditions, by a capsule-like vehicle. The reference vehicle for this research will be the Apollo Command Module.

To answer the main research question stated in Chapter 1, this work will aim to solve the problem of the optimal aerocapture guidance. This will be done by using both numerical and artificial intelligence solutions.

[9] Technically, on 333 trajectories, one failure is acceptable. Hence the minimization would be for the second worst case scenario, among the scenarios simulated.

3

FLIGHT DYNAMICS

As stated in the first chapter, an aerocapture deals with both atmospheric entry and orbital flight. Both situations have to be modeled, for two reasons: first, a comprehensive and precise model is necessary for the simulator to be accurate; second, a simpler model is needed for the NPC guidance logic to be able to make the proper decisions in real time. Requirements are thus different for the two objectives. The aim of this chapter will be that of describing the forces acting on the vehicle during an aerocapture, and its consequent motion, in the most accurate way. Nonetheless, some assumptions need to be made and justified for the simulator and the guidance logic. Also, different state variables and reference frames may be used for the two goals.

The simulator that will be used treats the vehicle as a point mass, neglecting its rotational dynamics. The bank angle, which is the controlled angle, is treated as a parameter. However, such parameter will be subject to physical constraints in its maximum velocity and acceleration. More about this will be explained in Subsection 6.2.6, and is not relevant for the moment.

The first step to take is defining the various reference frames in which forces are defined, and the relations between those very frames; this is done in Sections 3.1 and 3.2. In Section 3.3, it is decided by what variables the motion would be described; for different problems, different variables may be used. Section 3.4 discusses the fundamentals of dynamics; since the motion described is that of a point mass, these are rather simple. In Section 3.5 the environment and corresponding forces are described; the main forces acting on the vehicle are described, and models to obtain them are discussed. A stress is given to the atmospheric model and density perturbations. In addition, related to the environment, even though not with the dynamics, is aeroheating. In Section 3.7 the reasons why some simplifications can be done in the environment model are given. Eventually, Sections 3.8 and 3.9 describe the motion of the vehicle in specific sets of coordinates and reference frames, respectively for the orbital and atmospheric part of the flight. In both cases, the motion is derived from the forces described in 3.5.

3.1 REFERENCE FRAMES

Position, velocity and acceleration are all vectors, and are therefore defined by their direction and magnitude. The direction is strictly related to the reference frame the force is described in. For this reason, a discussion concerning reference frames is necessary.

The reference frames of interest for this research are listed below. All frames are right-handed: thus, only two axes need to be described for the frame to be uniquely defined. The description of the reference frames is based on [Mooij, 1997].

Inertial planetocentric frame. This reference frame is, as the name says, inertial[1], and centered at the center of mass (c.o.m.) of the central body. It is defined by the subscript $\mathcal{I}$. The $Z_{\mathcal{I}}$-axis points towards the North pole[2], and the $X_{\mathcal{I}}$-axis points towards the point in which the intersection between the reference meridian[3] and the equator is at the reference time $t = t_0$. The inertial frame is defined in a slightly different way on Earth, but this is of no use for this research.

This reference frame is of fundamental importance because the target orbit is defined with respect to it.

Rotating planetocentric frame. The rotating planetocentric reference frame (subscript $\mathcal{R}$) rotates together with the central body, and is fixed to its c.o.m.. The $Z_{\mathcal{R}}$-axis is the same as the $Z_{\mathcal{I}}$-axis, whereas the $X_{\mathcal{R}}$-axis points towards the intersection between the zero longitude meridian and the equator, and therefore coincides with the $X_{\mathcal{I}}$-axis once per revolution. This reference frame is of interest because the atmosphere rotates together with the planet.

Vertical frame. The vertical reference frame (subscript $\mathcal{V}$) defines, with its $X_{\mathcal{V}}$ and $Y_{\mathcal{V}}$-axes, the local horizon, and it is centered in the vehicle. The $Z_{\mathcal{V}}$-axis points towards the c.o.m. of the central body; the $X_{\mathcal{V}}$-axis is parallel to the meridian, perpendicular to the $Z_{\mathcal{V}}$-axis, which points northward[4]. This frame serves as a link between the aerodynamic and the rotating frames.

Aerodynamic reference frame. The aerodynamic reference frame (subscript $\mathcal{A}$) is centered in the vehicle; the $X_{\mathcal{A}}$-axis points in the direction of the velocity of the vehicle relative to the air (and, consequently, to the ground, because no wind is modeled in this research), and the $Z_{\mathcal{A}}$-axis points in the direction opposite to the lift force. The importance of this reference frame consists of the fact that the three axes are all in the opposite direction of the three components of the aerodynamic force.

3.2 FRAME TRANSFORMATIONS

Many sets of variables, or attitude parameters, can be used to rotate from one frame to another. Some of the sets most commonly used in aerospace applications are: direction cosine matrix (DCM), Euler angles, quaternions, Gibbs (or Rodrigues) parameters, and modified Gibbs parameters. The interested reader may refer to either Junkins and Turner [1986], Shuster [1993], or Wie [2008] for more information about attitude parameters. Diebel [2006] also provides a very complete survey of transformations between the different representations.

The goal of this section is to describe only the sets that would be used in this research, which are Euler angles[5]. Euler angles are chosen because they are used in literature to express the orientation between the frames used in this research[6]. For the same reason, they are used also in the equations of motion expressed in spherical, relative coordinates (see Section 3.9). Transformations between Euler angles make use of unit-axis rotations, which are special cases of DCMs: therefore, these are also briefly discussed in the next subsection.

[1] The definition of the adjective "inertial" differs between mechanical and relativistic mechanics. Here, it will be used in its classical meaning. Also, what will be called here inertial is pseudo-inertial instead; but for the purpose of this research, there is no difference between the two.

[2] Because of the relatively short durations of the simulations, any misalignment between the North pole and the $Z_{\mathcal{I}}$-axis that might be caused by precession and nutation is neglected.

[3] For the purpose of this research, it does not matter what the the reference meridian, nor the time t_0, are, as long as the choice is self-consistent throughout the report.

[4] This definition is valid for a spherical central body; if the shape is different, errors will be introduced.

[5] Here, for simplicity, Tait-Bryan angles are included when mentioning Euler angles. Also, rotations with Euler angles will be intrinsic for the entirety of the report: this means that the rotations after the first one are about the axes of the rotating coordinate systems, and not around the original, or initial, orientations.

[6] It might as well be possible to express the same relationships in terms of quaternions, or any other set of parameters.

3.2.1 EULER ANGLES

Any frame orientation with respect to another can be described by at least three coordinates. If those directions coincide to one of the axes of the rotated reference frames, the angles of each of the three rotations are called Euler angles. When using such a minimal set, it is impossible to ensure that in every possible orientation all coordinates are both unique and defined Junkins and Turner [1986].

The problem of having an undefined angle can be solved by using a different convention when needed. For those same orientations though, the kinematics become singular. This problem does not need to be solved, since those orientations happen in conditions that are very far from nominal, and are of no interest to this research[7]. There are 12 different sets of Euler angles [Wie, 2008], which differ from each other depending on the order of the axes around which the rotations occur. Rotations usually follow the sequence X-Y-Z or Z-Y-X, but also, in some cases, the Z-X-Z.

It is possible to obtain a DCM $\mathbf{C}_{\mathcal{B},\mathcal{A}}$ using the Euler angles α_1, $alpha_2$ and $alpha_3$ around, respectively, the X-, Y-, and Z-axis [Wie, 2008]:

$$\mathbf{C}_{\mathcal{B},\mathcal{A}} = \mathbf{C_1}(\alpha_1)\mathbf{C_2}(\alpha_2)\mathbf{C_3}(\alpha_3), \tag{3.1}$$

where

$$\begin{aligned}
\mathbf{C_1}(\alpha_1) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha_1 & \sin\alpha_1 \\ 0 & -\sin\alpha_1 & \cos\alpha_1 \end{bmatrix}, \\
\mathbf{C_2}(\alpha_2) &= \begin{bmatrix} \cos\alpha_2 & 0 & -\sin\alpha_2 \\ 0 & 1 & 0 \\ \sin\alpha_2 & 0 & \cos\alpha_2 \end{bmatrix}, \\
\mathbf{C_3}(\alpha_3) &= \begin{bmatrix} \cos\alpha_1 & \sin\alpha_1 & 0 \\ -\sin\alpha_1 & \cos\alpha_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.
\end{aligned} \tag{3.2}$$

are unit-axis rotation matrices around, respectively, the X-axis, Y-axis and Z-axis.

It is now possible to transform the vector $\mathbf{r}_{\mathcal{A}}$ (expressed in terms of the coordinates of frame $\mathcal{A}$) into the vector $\mathbf{r}_{\mathcal{B}}$, expressed in coordinates of frame $\mathcal{B}$:

$$\mathbf{r}_{\mathcal{B}} = \mathbf{C}_{\mathcal{B},\mathcal{A}}\mathbf{r}_{\mathcal{A}} \tag{3.3}$$

Kinematic differential equations can be obtained for Euler angles as well, but will not be reported; in case the sequence is $Z-Y-X$ a singularity occurs for the second angle tending to $\pm 90°$. These equations will not be directly applied in this thesis; nevertheless, the spherical equations of motion, which are reported in Section 3.9, are a consequence of, among other things, these kinematic relations. As an example, this is a reason why a singularity occurs, in that set of equations, when the flight-path angle goes to $\pm 90°$.

The rotations between the previously described frames are listed and their respective Euler angles are described in the next subsection.

3.2.2 FRAME TRANSFORMATIONS

Below the frame transformations that will be used in this research are listed. These transformations are specific examples that show how to obtain the corresponding DCM, provided that one knows

[7] This is not entirely true, since one such condition is flight exactly above one of the poles. Nevertheless, since it is almost impossible for a point mass to be exactly on a specified axis, the vehicle will only be in proximity of the poles axis. This occurrance may cause some numerical issues, but no singularities.

the required Euler angles. The rotation between any of the frames mentioned in Section 3.1 can be computed by use of the here reported transformations, or a combination of them. Unless otherwise stated, the source of this subsection is Mooij [2014]. For every rotation, it will be pointed out whether, and for which orientation, singularities in the kinematics equations occur. These are important to understand when the use of the equations of motion in terms of spherical, relative coordinates (described in Section 3.9) may lead to problems.

$\mathcal{R}$ to $\mathcal{I}$ By definition of the two reference frames involved in this transformation, the rotation is done using the following matrix:

$$\mathbf{C}_{\mathcal{I},\mathcal{R}} = \mathbf{C_3}(-\omega_{cb}t), \tag{3.4}$$

where ω_{cb} is the angular speed of the given planet around its Z-axis. This rotation never causes any singularity. It is often necessary to transform the velocity $\mathbf{V}$ from the rotating to the inertial frame. In such case:

$$\mathbf{V}_{\mathcal{I}} = \frac{d\mathbf{r}_{\mathcal{I}}}{dt} = \frac{d\mathbf{C}_{\mathcal{I},\mathcal{R}}}{dt}\mathbf{r}_{\mathcal{R}} + \mathbf{C}_{\mathcal{I},\mathcal{R}}\frac{d\mathbf{r}_{\mathcal{R}}}{dt} = \boldsymbol{\omega}^{\mathcal{I}}_{\mathcal{I},\mathcal{R}} \times \mathbf{r}_{\mathcal{R}} + \mathbf{C}_{\mathcal{I},\mathcal{R}}\mathbf{V}_{\mathcal{R}}, \tag{3.5}$$

where $\mathbf{r}$ is the position vector, and $\boldsymbol{\omega}^{\mathcal{I}}_{\mathcal{I},\mathcal{R}}$ is the angular velocity vector of the rotating frame with respect to the inertial one, expressed in the coordinates of the inertial frame.

$\mathcal{V}$ to $\mathcal{R}$ Given the planetocentric longitude τ and latitude δ, the transformation matrix is given as:

$$\mathbf{C}_{\mathcal{R},\mathcal{V}} = \mathbf{C_3}(-\tau)\mathbf{C_2}(\pi/2+\delta). \tag{3.6}$$

With the first rotation around the $Z_{\mathcal{R}}$-axis, the X-axis points in the direction of the meridian of longitude τ; with the second rotation, around the $Y_{\mathcal{V}}$-axis, the Z-axis becomes perpendicular to the surface of the (spherical) central body at the coordinates τ, δ. The $X_{\mathcal{V}}$ and $Y_{\mathcal{V}}$-axis define the plane tangent to the central body. This rotation causes singular kinematics for $\delta = \pm\pi/2$. For those values of δ, τ is undefined.

$\mathcal{A}$ to $\mathcal{V}$ This rotation is obtained by use of the following equation:

$$\mathbf{C}_{\mathcal{V},\mathcal{A}} = \mathbf{C_3}(-\chi)\mathbf{C_2}(-\gamma)\mathbf{C_1}(\sigma), \tag{3.7}$$

where γ is the flight-path angle, χ is the heading and σ is the bank angle. All the angles used here are those that define the rotation between those two reference frames; therefore, the transformation is straightforward. The heading angle[8] is undefined when $\gamma = \pm\pi/2$; also, as previously mentioned, the kinematics equations become singular for those same conditions.

3.3 STATE VARIABLES FOR TRANSLATIONAL MOTION

As for the attitude parameters, in some cases the description of the motion can be more convenient to be done in one or another set of state variables. Consequently also transformations between different sets of state variables are needed.

In this section, the sets of state variables that will be used are listed, followed by an explanation of what they are used for, why those were chosen, and how to transform from one set to another. The full state vector usually includes (at least) twelve variables: six variables define position and velocity, and six more define attitude and rotational velocity of the vehicle. In this research, only the first six are used. However, two additional state variables are added, namely the bank angle and bank angle rate. These are, technically, two components of the rotational state; however, they are treated in a particular way, as will be shown in Subsection 6.2.6. Mass could also have been treated

[8] For such an orientation, a problem also occurs with the bank angle, which loses its capability of controlling the vertical component of the lift. Because of this, it will also be considered as undefined for the remainder of the research.

as a state variable. However, in this research, the change in mass is only due to heat-shield ablation and reaction control system (RCS) consumption, and is assumed to be negligible.

The motion is supposedly the same, no matter what variables are used. Hence, before starting the discussion, it is necessary to understand why different state variables are used. Below is a schematic list that explains which, and why, different variables sets were used.

1. **Cartesian state variables**: used to simulate the trajectory, they are chosen because the equations of motion using these variables make the software very flexible.
2. **Spherical state variables**: used in the prediction of the NPC, they are chosen because they allow the equations of motion to be very fast, with only six lines of code, and only six transcendental functions. In most literature they are used for this same purpose. In addition, they are used for analyzing the results, as they are intuitive to the engineers.
3. **Orbital state variables**: used to define the target orbit, and also as inputs for the artificial intelligence guidance, as they are intuitive too.

Some of this sets also has some issues, which will be discussed.

3.3.1 CARTESIAN STATE VARIABLES

Cartesian state variables are the components of the position and velocity vectors with respect to the three orthogonal axes of the reference frame in use. The state vector is made of the three position components along the three axes of the reference frame, and the total derivatives of those with respect to time:

$$\mathbf{x} = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} \end{bmatrix}. \tag{3.8}$$

Cartesian coordinates are the only singularity-free set of variables listed here.

3.3.2 SPHERICAL STATE VARIABLES

These state variables are the distance from c.o.m. of the central body r, longitude τ and latitude δ for what concerns the position; for the velocity, V is the magnitude, γ is the flight-path angle and χ is the heading. Figure 3.1 also defines those variables. It should be stated that all the angles used in this set are Euler angles, and therefore, with their use, come all the drawbacks listed in Subsection 3.2.2.

An easy transformation between Cartesian and spherical variables follows:

$$r = \sqrt{x^2 + y^2 + z^2}, \tag{3.9}$$

$$\tau = \text{atan2}(y, x), \tag{3.10}$$

$$\delta = \text{atan2}(z\sqrt{x^2 + y^2}). \tag{3.11}$$

The Cartesian velocity components should then be rotated into the vertical frame using the rotations shown in Subsection 3.2.2. At that point, the three remaining variables can be computed in (almost) the same way as the first one. In fact:

$$V = \sqrt{\dot{x}_V^2 + \dot{y}_V^2 + \dot{z}_V^2}, \tag{3.12}$$

$$\chi = \text{atan2}(\dot{y}_V, \dot{x}_V), \tag{3.13}$$

$$\gamma = -\text{atan2}(\dot{z}_V, \sqrt{\dot{x}_V^2 + \dot{y}_V^2}). \tag{3.14}$$

No singularities explicitly appear in these transformations, since the atan2 function is always defined.

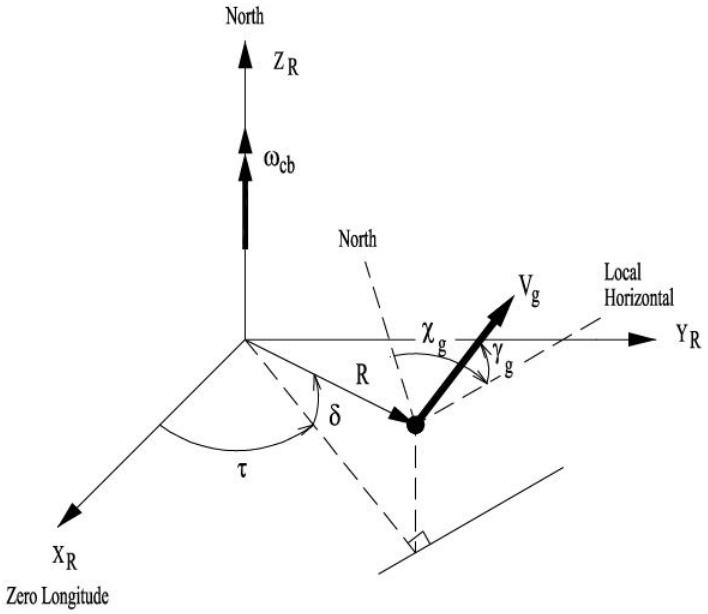


Figure 3.1: Definition of the spherical state variables [Mooij, 1997].

The inverse transformation is:

$$x = r\cos\tau\sin\delta, \tag{3.15}$$

$$y = r\sin\tau\sin\delta, \tag{3.16}$$

$$z = r\cos\delta. \tag{3.17}$$

The transformation for the velocity components is similar, using the opposite sign for the flight-path angle and the frame rotation.

3.3.3 ORBITAL PARAMETERS

Five of these parameters are constant along an unperturbed orbit; this is indeed the reason why the orbital parameters are used to define the target orbit. The sixth describes the position along that orbit. The five constant elements can be used to describe shape, size and orientation of the orbit. Those are (some of which are shown in Figure 3.2):

- **semi-major axis** a: its geometric definition differs between open and closed orbits; a universal definition is, instead, $a = -\mu/2E$, where E is the specific orbital energy (see Section 3.8);
- **eccentricity** e: it defines the shape of the orbit: $e = 0$ is for circular orbits, $0 < e < 1$ for elliptic orbits, $e = 1$ for parabolic orbits and $e > 1$ for hyperbolic orbits;
- **inclination** i: it is the angle between the angular momentum of the orbit and the $Z_{\mathcal{R}}$-axis. It is defined for $0° \leq i \leq 180°$, being $i = 0°$ for an equatorial prograde orbit, $i = 90°$ for a polar orbit and $i = 180°$ for an equatorial retrograde orbit.
- **Right Ascension of the Ascending Node (RAAN)** Ω: it is the angle between the line of nodes and the $X_{\mathcal{I}}$-axis, and is defined for $0° \leq \Omega < 360°$.
- **argument of periapsis** ω: it is the angle between the line of nodes and the eccentricity vector (which is the vector that points from the focus of the ellipse in the direction of the periapsis of the orbit) and is defined for $0° \leq \omega < 360°$.

The remaining time varying variable, which specifies the position of the body on the orbit described by the afore-mentioned elements, can be described in many ways. The chosen one or this report is the true anomaly, defined as:

- **true anomaly** θ: it is the angle between the eccentricity vector and the position vector;

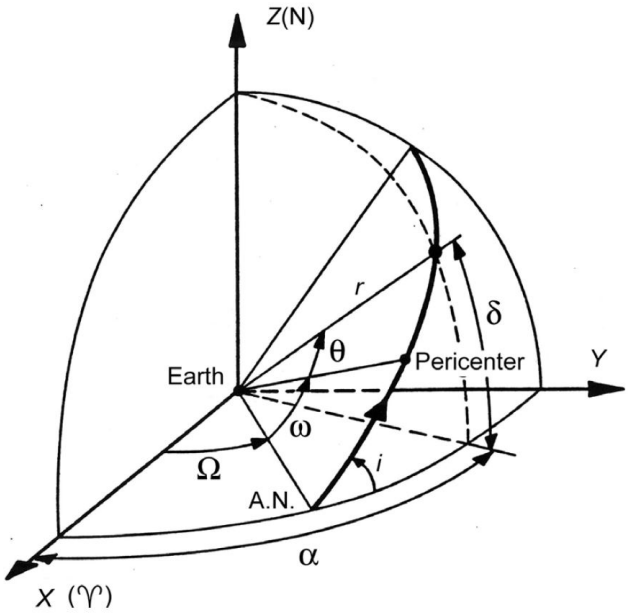


Figure 3.2: Definition of i, Ω, ω, and θ [Wakker, 2015].

Unfortunately these variables suffer from singularities as well. From its definition, it is clear that a singularity for the semi-major axis occurs during aerocapture, since such a maneuver brings the vehicle's specific orbital energy from positive to negative. This is easily solved using the orbital energy itself[9]. Additional problems occur when using these variables to propagate the motion, but this is not the case in this research.

The transformation from Cartesian, inertial planetocentric state variables, to orbital parameters (and its inverse one) is a rather simple, but relatively long, algorithm, which can be found on any standard astrodynamics book (e.g.: Wakker [2015], Wertz [2009] or Vallado [2001]), and will not be given here. Wakker [2015] also provides an algorithm to obtain orbital elements from spherical coordinates.

3.4 FUNDAMENTALS OF DYNAMICS

Newtonian, or classical, mechanics are sufficient to describe the motion of the vehicle for this problem. Newton's Laws were first enunciated in his *Principia* in 1687. Those are:

- A body, when viewed in an inertial reference frame, keeps its constant velocity (equal or different to zero) unless the sum of all the external forces acting on it is different from zero.
- A body acceleration is directly proportional to the force applied to it and inversely proportional to its mass.

[9] Two other, less relevant, problems occur with this set. One occurs when $e = 0$, which causes the argument of the periapsis to be undefined. This is not a problem for this research, because during a successful aerocapture $e = 0$ never holds, and because the argument of periapsis is never used anyways. The other occurs when $i = 0$, which causes the RAAN to be undefined, but is never used either.

- To each action corresponds a reaction opposite in direction and equal in magnitude.

Newton's laws refer to the translational motion of a rigid body. The vehicle whose motion is described can be very well approximated as such. In an inertial frame, Newton's laws for the translational motion of a rigid body can be synthesized by the following equation, in case i and j form an isolated system:

$$\mathbf{F}_{ij,\mathcal{I}} = -\mathbf{F}_{ji,\mathcal{I}} = m_j \frac{d\mathbf{V}_{j,\mathcal{I}}}{dt} = -m_i \frac{d\mathbf{V}_{i,\mathcal{I}}}{dt}, \tag{3.18}$$

where $\mathbf{V}_{i,\mathcal{I}}$ and $\mathbf{V}_{j,\mathcal{I}}$ are the inertial velocities of the c.o.m.s of the two considered bodies. In case one does not have an isolated system, the velocity of the c.o.m. of body i varies as follows:

$$\mathbf{F}_{i,\mathcal{I}} = m_i \frac{d\mathbf{V}_{i,\mathcal{I}}}{dt}, \tag{3.19}$$

where $\mathbf{F}_{i,\mathcal{I}}$ is the sum of all the external forces applied to i.

So-called pseudo-forces appear when describing the motion in a non-inertial frame. This is the case when using the spherical relative equations of motion, reported in Section 3.9.2. Pseudo-accelerations in this research are caused by the use of the $\mathcal{R}$ frame, whose origin moves inertially, but whose orientation is not inertial, and changes at a constant rate. In such a case, the second of Newton's Laws is modified as follows:

$$m_i \frac{d\mathbf{V}_{i,\mathcal{R}}}{dt} = \mathbf{F}_{i,\mathcal{I}} - \boldsymbol{\omega}^{\mathcal{I}}_{\mathcal{I},\mathcal{R}} \times \left(\boldsymbol{\omega}^{\mathcal{I}}_{\mathcal{I},\mathcal{R}} \times \mathbf{r}_{i,\mathcal{R}}\right) - 2\boldsymbol{\omega}^{\mathcal{I}}_{\mathcal{I},\mathcal{R}} \times \mathbf{V}_{i,\mathcal{R}}, \tag{3.20}$$

where ($\boldsymbol{\omega}^{\mathcal{I}}_{\mathcal{I},\mathcal{R}}$ is the vector expressing the rotation of $\mathcal{R}$ with respect to $\mathcal{I}$, in $\mathcal{I}$ coordinates.

3.5 ENVIRONMENT AND FORCE MODELS

The main forces acting on the vehicle during re-entry are gravity and aerodynamics. It is therefore necessary to understand the laws of both of them. In this section, the models to evaluate these forces are discussed. However, in the case of aerodynamics, the force requires the modelling of the atmosphere (which will be treated in a dedicated section 3.6), of different flow regimes, of the vehicle, as well as the shape of the Earth (which will be discussed first). Moreover, strictly related to aerodynamics, is also aeroheating, treated in Subsection 3.5.5.

Eventually, since nor a guidance logic nor a computer simulator can be exhaustive, due to both limits of the models and computational power, Section 3.7 discusses also which forces will be neglected, as well as which models should or not be used, and why.

3.5.1 PLANET SHAPE

In general, no planet is exactly spherical, and the difference between polar and equatorial radius may be of even tens of kilometres. This difference is expressed by the flattening parameter f [Mooij, 1997]:

$$f = 1 - \frac{R_p}{R_e}, \tag{3.21}$$

where R_p is the polar radius of the celestial body, and R_e is the equatorial radius. When a planet is assumed oblate, one has the following approximation[10] for the altitude [Mooij, 2014]:

$$h \approx r - R_p\left(1 - e\sin^2\delta\right). \tag{3.22}$$

[10] The approximation comes from the first two terms of the Taylor expansion of an ellipse in spherical coordinates. In the equation, δ should be the geographic latitude, which is different from the geodetic latitude. Defining such a difference is not of interest for this research. It is sufficient to know that the error caused by using the geocentric latitude is also small.

All atmospheric models are defined with respect to the altitude above the reference ellispoid. For Earth, using a spherical model instead of an ellipsoid on, can cause errors in altitude up to 20 km. This can, in turn, cause a difference in density of up to 5 times. Also, one should remind that a strict consequence of the previous equation is that, in non vertical flight, $\dot{r} \neq \dot{h}$.

It should be noted that the, during this research, the shape of the planet is not considered when defining the apoapsis altitude. The latter will be defined to be the difference between the apoapsis radius and the equatorial radius of the planet.

3.5.2 GRAVITY

Newton published the law of universal gravitation in his Principia, in 1687. With that equation, for the first time, it was understood that the laws governing celestial mechanics were the same that act on terrestrial environment:

$$\mathbf{F}_{ij} = -G\frac{m_i\, m_j}{r_{ij}^3}\mathbf{r}_{ij}, \tag{3.23}$$

where G is the gravitational constant. In astrodynamics, the gravitational parameter μ is preferably used:

$$\mu = G\, m_j. \tag{3.24}$$

Because gravity is conservative, it is possible to find the potential of the field generated by one of the two bodies that are attracting each other:

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

where r is the norm of r_{ij}. These laws are only valid if the two bodies i and j are either point masses or perfect spheres. For this reason, if higher accuracy is required, gravity is usually modeled using Legendre polynomials [Wakker, 2015].

As it will be shown in Section 3.7, in aerocapture it is not necessary to include non-central gravity components other than that due to the second degree zonal harmonic, J_2. Its effect is given below, in Cartesian components, with respect to the inertial reference frame [Wakker, 2015]:

$$\begin{aligned} g_x &= -\frac{3}{2}\mu J_2 \frac{R^2}{r^5} x(1-5\frac{z^2}{r^2}),\\ g_y &= -\frac{3}{2}\mu J_2 \frac{R^2}{r^5} y(1-5\frac{z^2}{r^2}),\\ g_z &= -\frac{3}{2}\mu J_2 \frac{R^2}{r^5} z(1-5\frac{z^2}{r^2}). \end{aligned} \tag{3.26}$$

and in the vertical reference frame, in spherical components (needed for the equations in spherical relative coordinates, reported in Section 3.9):

$$\begin{aligned} g_\delta &= -\frac{3}{2}\mu J_2 \frac{R_e^2}{r^4}\sin 2\delta,\\ g_\tau &= 0,\\ g_r &= \frac{3}{2}\mu J_2 \frac{R_e^2}{r^4}\left(3\sin^2\delta - 1\right). \end{aligned} \tag{3.27}$$

It is worth mentioning that since the J_2 effect only depends on the latitude, and not on the longitude, as seen in the last set of equations, the gravity field is still conservative. The potential energy has a different form from that of a central gravity field [Wakker, 2015]:

$$U = -\frac{\mu}{r} + \frac{1}{2}\mu J_2 \frac{R_e^2}{r^3}\left(3\sin^2\delta - 1\right). \tag{3.28}$$

This property will be very important for the verification in Section 6.2.3.

3.5.3 AERODYNAMICS

The aerodynamic force is caused by the interaction between a vehicle and a fluid in relative motion with respect to one another. The total force and moment acting on the vehicle are the result of the integral of friction and pressure on the totality of the surface of the vehicle. Friction and pressure distributions depend on many variables, of which the most relevant are: shape, size and attitude of the body, dynamic pressure, Mach number, Knudsen number and Reynolds number. The shape and size of the body are assumed constant during flight. The Knudsen number describes how much rarefied the gas is, and has a strong influence in the aerodynamic coefficients, even though that occurs at very high altitudes, where the aerodynamic force is very small. In Section 3.7 it will be explained why the dependency on the Knudsen number can be neglected. Reynolds number is always very large during aerocapture, and due to its asymptotic influence on the coefficients, it can be considered as a constant[11]. Therefore, during a flight, the aerodynamic coefficients are functions of only attitude and Mach number. It will be seen in the following subsection that for a capsule-like vehicle in trim conditions, the angle of attack is also function of the Mach number.

The Mach number is the ratio between the velocity of the vehicle and the local speed of sound, which is a function of the chemical properties of the gas and its temperature; thus:

$$M = \frac{V}{a} = \frac{V}{\sqrt{\gamma R T}}. \tag{3.29}$$

In general, there is an linear relation between the force and the dynamic pressure; thus, the aerodynamic acceleration is usually expressed as [Mooij, 1997]:

$$\mathbf{F}_{\mathbf{A},\mathcal{A}} = \begin{bmatrix} -D \\ -S \\ -L \end{bmatrix} = -\frac{\bar{q} S_{ref}}{m}, \begin{bmatrix} C_D \\ C_S \\ C_L \end{bmatrix}, \tag{3.30}$$

where $\bar{q}$ is the dynamic pressure, S_{ref} is a reference surface, constant for the body, m is the mass of the vehicle, and C_D, C_L and C_S are the aerodynamic coefficients, respectively for drag, lift and sideslip. The latter coefficients are functions of all the variables mentioned above.

The aerodynamic force is always assumed to be acting on the c.o.m. of the vehicle; since this is an incorrect assumption, an aerodynamic moment also needs to be added. However, due to the assumptions that will be stated in Subsection 3.5.4, there is no need to include these moments in this research.

Being linearly dependent on the dynamic pressure, the aerodynamic force is a linear function of the density as well. Different ways to estimate the density, as well as to compute possible perturbations, are reported in Section 3.6.

As a final remark, the entirety of aerocapture occurs in the hypersonic regime, which is defined as flight at Mach numbers higher than 5. According to Anderson [2000], at very high Mach number (higher than 15) there is almost no more relation between Mach number and aerodynamic coefficients: the latter have indeed an asymptotic behavior for the former going to infinity. This property will be used in Section 3.7, when making assumptions about considerations on the atmospheric temperature profile.

3.5.4 VEHICLE MODEL

For a capsule-like vehicle no control surfaces are available. Therefore, except for the bank angle, the attitude of the vehicle is the result of the balance between aerodynamic moments with respect to the c.o.m.. The equilibrium angle of sideslip is zero because of symmetry. Concerning the angle of attack, the equilibrium is obtained with the angle for which the aerodynamic pitching moment

[11] This is not true in proximity of the stagnation point though.

Table 3.1: Trim angle of attack and corresponding aerodynamic coefficients for the Apollo Command Module [Robinson et al., 2009].

Mach no.	Angle of attack [°]	C_L	C_D	BC [kg m^{-1}]
0.4	-12.86	0.24465	0.85300	537.39
0.7	-15.62	0.26325	0.98542	465.16
0.9	-18.3	0.32074	1.10652	414.26
1.1	-25.13	0.49373	1.16970	391.86
1.2	-24.87	0.47853	1.15600	396.50
1.35	-25.99	0.56282	1.27880	358.44
1.65	-26.78	0.55002	1.26570	362.14
2.0	-26.86	0.53247	1.27210	360.34
2.4	-26.38	0.50740	1.24120	369.32
3.0	–25.86	0.47883	1.21670	376.74
4.0	-23.88	0.44147	1.21480	377.32
10.0	-23.21	0.42856	1.22460	374.30
≤ 29.5	-19.94	0.38773	1.28910	355.61

coefficient is zero, and stable. Such coefficient is a function of Mach number as well: thus, the trim angle of attack is a function of Mach number only.

Robinson et al. [2009] show the aerodynamic data of Apollo in trim conditions in their paper: their values are used, which are reported in Table 3.1. Their values have been obtained by interpolation of the data in NASA [1965].

Perturbations of ±10 % in the drag and lift coefficients will be included during the simulations. These are mainly cause by offsets in the c.o.m., which, in turn. causes a different trim angle-of-attack

Additional relevant data are the reference surface, set equal to 12.017 m, and the mean mass, 5000 kg. The latter is varied during the simulations by ±1.5 %.

3.5.5 AEROHEATING

At hypersonic regime, aeroheating becomes very important and causes a strong constraint for the vehicle. Aeroheating constraints include heat-flux, total heat load, and derivative of the heat-flux. Heat-flux determines the equilibrium temperature of the outer surface of the vehicle, which is usually subject to a limit; the total heat load is instead usually important when an ablative thermal protection system (TPS) is being used. In theory, the derivative of the heat-flux is also important because, the larger it is, the less the vehicle is in thermal equilibrium, and thus larger gradients and thermal stresses occur. Nonetheless, this last element will not be considered here.

Anderson [2000] offers many methods to compute these parameters, most of which are very precise and complicated. Specific computational methods applied to re-entry can be found in Prakash and Zhong [2009] or in Gnoffo [2003]. However, since these values consist of constraints only, one can use simpler, and less precise, methods, and apply safety factors to those.

At supercircular velocities the main components of the heat-flux come from air through convection and radiation; in this research, it is assumed that all the heat is then radiated outwards by the surface of the heat shield. Any heat ablating the heat shield, or conducted inwards, will not be considered.

Sutton and Graves [1971] give an easy relation to compute the heat-flux due to convective heat for the stagnation point of an axisymmetrical blunt body in a hypersonic flow in chemical equilib-

rium:

$$\dot{q}_{conv} = K\sqrt{\frac{\rho}{R_n}}V^3 \qquad \left[\mathrm{W\,m^{-2}}\right], \tag{3.31}$$

where R_n is the nose radius of curvature, and K is a constant depending on the gas mixture. This relation assumes laminar flow. The constant is found to be equal to 1.7623×10^{-4} on Earth [Samareh, 2009].

A more conservative approach would be that of using an empirical method, proposed by Detra and Hidalgo [1961]. The methods were compared by Carandente et al. [2013], and it turned out that the latter is more conservative than the former. The Detra-Hidalgo relation is:

$$\dot{q}_{conv} = 5.15\times10^{-5}\sqrt{\frac{\rho}{R_n}}V^{3.15}, \qquad \left[\mathrm{W\,m^{-2}}\right] \tag{3.32}$$

This relation is set for values for Earth's atmosphere, and will be used in the analysis of the results, being more conservative. The Sutton-Graves relation will instead be used in the validation.

Both previously mentioned equations are valid when considering a cold wall. In re-entry problems, however, the wall heats up relatively fast, and thus the wall temperature should be included:

$$\dot{q}_{conv,hw} = \dot{q}_{conv}\left(1-\frac{T_w}{T_{w,ad}}\right), \tag{3.33}$$

where T_w is the wall temperature (the computation of which is discussed a few paragraphs below) and $T_{w,ad}$ is the adiabatic wall temperature:

$$T_{w,ad} \approx \frac{H_{tot}}{c_p}, \tag{3.34}$$

$$H_{tot} \approx \frac{V^2}{2}, \tag{3.35}$$

where c_p is the specific heat of the gas at constant pressure.

Whereas Equation (3.35) becomes more accurate for larger velocities, Equation (3.34) becomes more and more inaccurate for larger enthalpies.

The problem here is that c_p varies with temperature, and therefore this approximation gives values that are highly off at hypersonic velocities: as an example, at $12\,\mathrm{km\,s^{-1}}$ one obtains an adiabatic wall temperature of about 72 000 K, which is very far from reality, because of ionization and many other reactions occurring. A correction for this is proposed, based on the interpolation of the work by Menart and Henderson [2008], which provides the total enthalpy for ionized air at 1 kPa, in chemical equilibrium. Figure 3.3 shows the plot from which such relation is interpolated.

The value of 1 kPa has been chosen because it is the closest to the maximum dynamic pressure encountered by the validation trajectory simulated in Subsection 6.2.7 (which is about 3 kPa). By so doing, one obtains, for that same validation case, a range of adiabatic wall temperatures between 10 000 K and 6000 K. The hot wall correction has then an impact for about 25 % of the total convective heat rate, being the shield equilibrium temperature around 2500 K.

At superorbital velocities also radiative heating from the gas becomes relevant. Tauber and Sutton [1991] proposed an easy analytical expression, which requires the interpolation of only one small table, for the radiative heating at stagnation point of a blunt body, assuming thermochemical equilibrium. Also in this case, heat strictly depends on the gas mixture, and thus, on the planet. This kind of heat becomes almost zero for velocities lower than $9000\,\mathrm{m\,s^{-1}}$ on Earth. The expression has the form (in $\mathrm{W\,cm^{-2}}$):

$$\dot{q}_{rad} = C\,R_n^a\,\rho^b\,f(V), \tag{3.36}$$

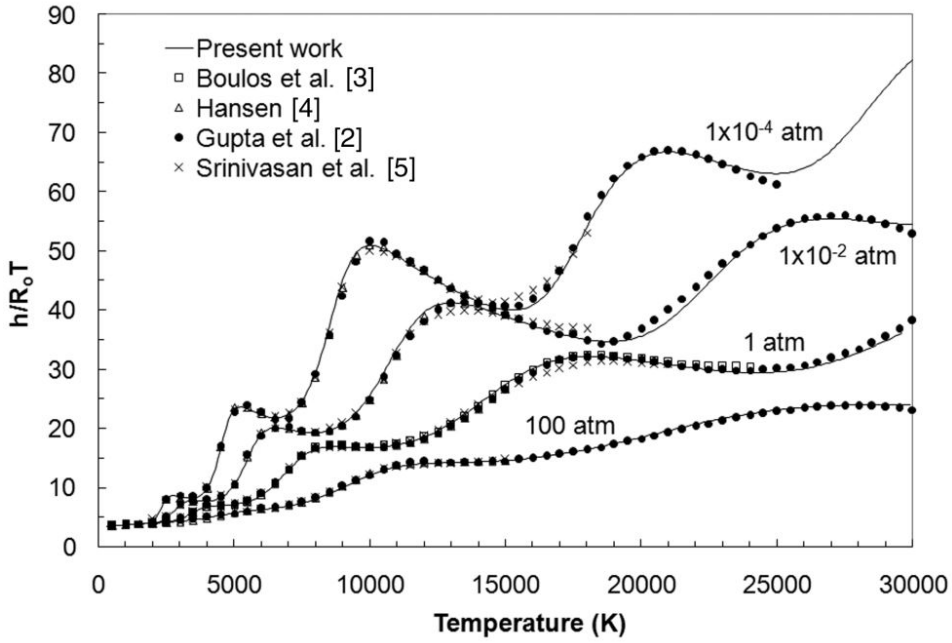


Figure 3.3: Air temperature at chemical equilibrium as a function of nondimensional total enthalpy, at different pressures. From Menart and Henderson [2008].

where C is a constant depending on the atmosphere gas mixture, and $f(V)$ is the interpolated function, and depends on the atmospheric gas mixture. Its tabulated values can be found on Tauber and Sutton [1991] for both Mars and Earth. Values of a and b depend on the planet, but may as well be a function of nose radius or velocity. For air, on Earth:

$$\begin{aligned} C &= 4.736 \times 10^4 \\ a &= 1.072 \times 10^6 \, V^{-1.88} \, \rho^{-0.325} \\ \text{if} \quad 1\,\text{m} \leq R_n \leq 2\,\text{m}, &\qquad a \leq 0.6 \\ \text{if} \quad 2\,\text{m} < R_n \leq 3\,\text{m}, &\qquad a \leq 0.5 \\ b &= 1.22 \end{aligned} \tag{3.37}$$

Equation (3.36) gives an error between -17% and +15% for the situations in which it was compared to more accurate numerical methods.

In case one assumes that the wall is in thermal equilibrium, which is reasonable in case of reusable heat-shields, the wall temperature can be found by equating the sum of radiative and convective heat to the radiative heat emitted by the wall itself:

$$\dot{q}_{conv}\left(1 - \frac{T_{w,eq}}{T_{w,ad}}\right) + \dot{q}_{rad} = \epsilon\sigma T_{w,eq}^4, \tag{3.38}$$

where σ is the Stephan-Boltzmann[12] constant and ϵ is the wall emissivity[13]. When the heat-shield is ablative, a large portion of the heat is dissipated in the ablation process. Therefore, the wall temperature would be smaller. However, this correction will not be considered in this research.

[12]The Stephan-Boltzmann constant is, at the best of the current knowledge, found to be equal to 5.670367×10^{-8} W/m^{2}K^{-4}

[13]The wall emissivity defines how closely the radiation of a body resembles that of a black body. The higher the value, the more the body radiates the heat outwards. It cannot be larger than 1. The emissivity of a body is equal to its absorptance, and in reality is a function of the wavelength. For wavelengths corresponding to the average temperature of a Lunar entry, the Apollo Command Module has $\epsilon \approx 0.8$ [Robinson et al., 2009]. In theory, also the incoming radiative

In conclusion, by summing both convective and radiative heat, making use of, respectively, the Detra-Hidalgo relation (Eq. 3.32) (corrected with hot wall correction, using interpolated data for adiabatic wall temperature), and the Tauber-Sutton (Eq. 3.36) relations, each with an appropriate safety factor, it would be possible to obtain a good and conservative estimate of the stagnation-point heating.

As a last remark for this subsection, it should be kept in mind that for a capsule-like vehicle flying at an angle of attack, as is usually the case, the point with maximum heat is usually not the stagnation point, but the *hot corner*, which is the part of the edge of the heat-shield that is first impinged by the flow. Also, the nose radius is often best substituted by the effective nose radius, which is a function of the angle of attack, and differs between convective and radiative heat-flux.

3.6 ATMOSPHERE MODELS

The aerodynamic force is a linear function of the density, and density also determines the heat-flux that a vehicle would encounter during entry. The speed of sound has instead an effect on the aerodynamic coefficients. For this research, the most important data about an atmosphere are:

- average altitude-density profile;
- average altitude-sound speed profile;
- perturbations model.

Average wind will not be used in this research, for reasons that are explained in Section 3.7. The perturbations model is instead fundamental to be able to properly evaluate the robustness of a guidance logic.

Atmospheric models are divided into two categories: reference and standard. Standard models give the parameters as function of altitude only, are relatively simple, and aim to give an estimate that is reasonably accurate at any latitude, longitude and date of the year.

A reference model gives all the relevant parameters as a function of also latitude, longitude, date of the year, and other variables. Moreover, reference models such as the Earth GRAM-99 include some rather sophisticated perturbations models [Leslie and Justus, 2011].

The use of a perturbation model is particularly important for the training of the neural networks: in fact, any pattern can be learned by the network; this includes an atmospheric profile, if it were always the same (or just multiplied by a constant value as well).

The choices for the atmospheric modeling are listed below, together with the purpose and reason why they were chosen.

1. **US 76 profile**: used in the model of the NPC. It is chosen because it is simple, and aims to be a generic, average profile. Also, it is used in the guidance by Lu et al. [2015], which is reproduced in this research.

2. **GRAM-99 profile**: used in the simulator. It is chosen because it is necessary to have a different model from that of the predictor. Any other model would have worked just as well, as long as it were different from the US76. Moreover, for the same reason, it is sufficient to have a single profile, and not the entire model that varies with space and time. It is also stressed that it is of no interest for this research aiming for the most accurate/recent model. At this stage of the development, an accurate trade-off for the best atmospheric model is not needed.

heat estimated with Equation (3.36) should be multiplied by an emissivity/absorptance; nevertheless, this radiation occurs at a variety of wavelengths. To be conservative, it is safer to let the corresponding emissivity/absorptance be equal to 1.

3. **GRAM-99 perturbation model**: used in the simulator. A perturbation model is needed to evaluate the responsiveness of a guidance logic to highly perturbed environments. Moreover, a neural network would be able to learn any pattern, such as a constant atmospheric profile. This shall be avoided, since the atmosphere is not always the same also in reality. The GRAM-99 model was specifically chosen because its perturbations vary in time and in all three dimensions of space. This leads to encountering an atmospheric profile in the descending leg very different from the one of the ascending leg. Any other perturbation models that include such a feature would have been equivalent for the scope of this research.

In some cases, density perturbations will also be modeled as multiplications by a constant only. Such a constant is chosen to be random, and between 0.5 and 1.5. The speed of sound is instead always modeled as the multiplication by a constant, which varies between 0.85 and 1.15. Justification for these values of the constants can be found in the following subsection. Motivation why no complex perturbations models for the speed of sound are used is given in Section 3.7.

In all cases, the altitude is considered with respect to the reference ellipsoid that has been defined in Subsection 3.5.1.

3.6.1 US STANDARD ATMOSPHERE, 1976

As just mentioned, the US Standard Atmosphere, 1976 (US76) is used for the predictions of the NPC. This atmosphere is defined up to 1000 km of altitude.

Without going into too many details, the US Standard 76 Atmosphere assumes the atmosphere to be a dry, homogeneously mixed perfect gas, and to be always in hydrostatic equilibrium. All the parameters are functions of altitude only [NASA, 1976]. Moreover, the temperature gradient with respect to altitude is piecewise linear at low altitudes (below 91 km). The governing equations corresponding to the previously mentioned characteristics are:

$$p = \rho R T, \tag{3.39}$$

where R is the gas constant[14], function of its molecular mass. The hydrostatic equilibrium assumption implies, instead:

$$dp = -\rho g dz, \tag{3.40}$$

where the gravity acceleration is approximated with a Taylor polynomial of the first order, centered at sea level. The pressure can then be integrated along the altitude eliminating the density:

$$d\log p = \frac{dp}{p} = -\frac{g}{RT} dz, \tag{3.41}$$

where T is defined piecewise. To make the integration feasible, one transforms the altitude into geopotential altitude, such that g can be constant, and the temperature into molecular temperature, such that R can be constant. The definition of these two parameters is of no relevance for the scope of this research.

The profile for the temperature has been derived fitting empirical measurements taken in two different places in the United States, with about 20 measurements for each month of the year [NASA, 1976]. The two locations are at latitudes of 38°N and at 59°N: the fit is done by interpolating in a way such that the profile fits the 45°N.

Being an average of many months, this model does not provide a one realistic atmospheric profile, and therefore GRAM-99 will also have to be used in the simulator. Nevertheless, some of the simulations will also be carried out using this model as an average. In that case, the value of the speed of sound would also be needed.

[14] For air, at sea level, $R = 286.99\ \mathrm{J\,kg^{-1}\,K^{-1}}$.

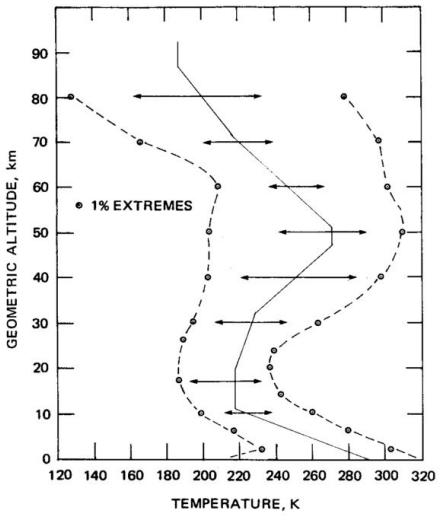


Figure 3.4: Range of systematic variability of density around US76 [NASA, 1976].

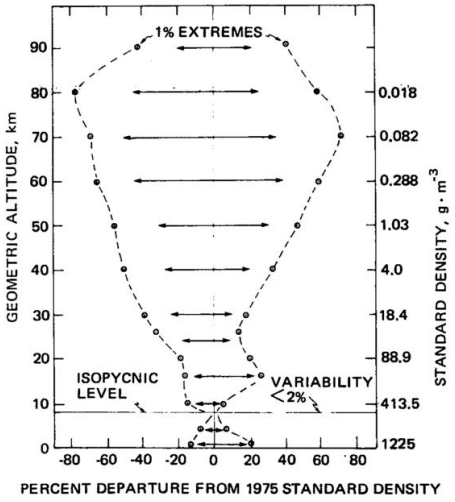


Figure 3.5: Range of systematic variability of temperature around US76 [NASA, 1976].

Figures 3.5 and 3.4 show how different months and latitudes affect the mean values of density and temperature at a certain altitude. The figure justifies the initial use of a basic perturbation model consisting of the multiplication of the density and temperature profiles by a constant. Since, as shown in Section 2.5, the largest amount of depletion of energy occurs between 50 km and 70 km, a basic uniform distribution of ±50% in density and of ±15% in temperature seem reasonable. However, since these variations concern latitudinal and monthly variations (they are, indeed, systematic), once a certain time and position for the initial conditions of aerocapture are defined, the variations are likely to be much smaller. When treating the GRAM-99 perturbation model, it will be shown how the standard deviations of perturbations would be much smaller.

An interesting feature that is worth being mentioned is the presence of discontinuities in density scale height H_ρ. The density scale height is an approximation to $(d\log\rho/dz)^{-1}$ [NASA, 1976], and therefore defines the rate of change of the density with the (geopotential) altitude. It can be computed as [NASA, 1976]:

$$H_\rho = \frac{H_p}{1 + H_p\left(\frac{d\log T}{dz} - \frac{d\log M}{dz}\right)}, \tag{3.42}$$

where H_p is the pressure scale height, and is continuous. From this, it is evident that when the altitude gradient is discontinuous, also the scale height is. This makes the atmospheric density's second derivative (with respect to altitude) discontinuous. More about the non-smoothness of the atmospheric density, and its implications, is discussed in Section 6.3.1.

3.6.2 EARTH GRAM-99 PROFILE

The GRAM-99 model is a combination of 3 different previous models: the Global Upper Air Climate Atlas (GUACA) for altitudes between 0 km and 27 km, the Middle Atmosphere Program (MAP) for altitudes between 20 km and 120 km, and the Jacchia [1970] model for altitudes above 90 km.

It is not the scope of this report to describe these models. All that is needed to know is that the model relies on measurements of pressure and temperature, for different months and for a grid of altitudes, latitudes and longitudes. Those measurements are interpolated vertically assuming hydrostatic equilibrium, and linearly in the other directions. Density is then computed assuming perfect gas. Hence, the same discontinuities as in the US76 Standard occur, but even more frequently.

For the scope of this research, it is sufficient to have a realistic atmospheric profile that differs from the US76.

The sample profile of speed of sound and density has been taken from Appendix E in Justus and Johnson [1999]. Its values have been manually copied. The table also provides difference in percentage between GRAM-99 and US76: these values were used to verify that the no errors were made when manually importing the data. The difference between the US76 and the chosen profile (for density only, in percentage) can be seen in Figure 3.6.

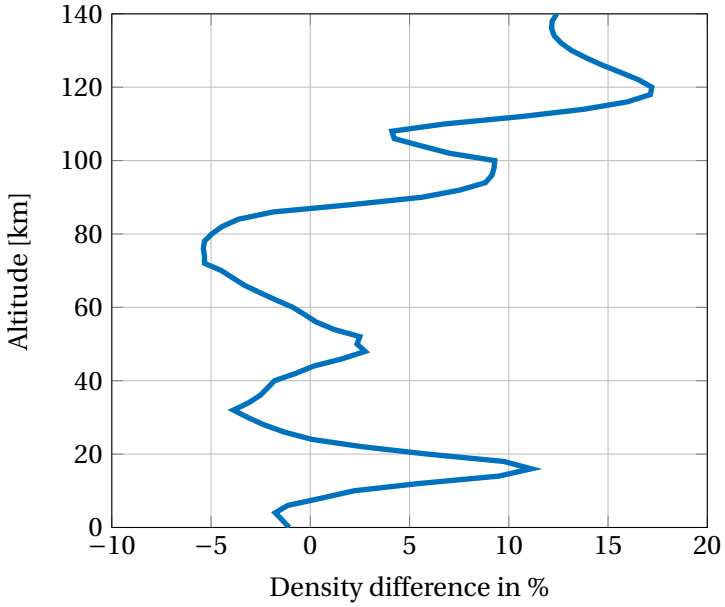


Figure 3.6: Difference in percentage of US76 density between the mean atmospheric density from Appendix E of Justus and Johnson [1999] and the US76 density profile.

The GRAM-99 has been used instead of the newer version, GRAM-2010, because the latter is not publicly available. Despite a similar table would have been available for the newer version as well, the data of the perturbations model is not (but it is for the GRAM-99). Therefore, for reasons of consistency, also the average profile has been taken from the older version. The values taken from Appendix E of Justus and Johnson [1999] are then interpolated in the altitude using a spline. Since the data reported on the GRAM-99 end at an altitude of 140 km, for higher altitudes the US76 atmosphere is used.

One of the unique features of the GRAM-99 is its ability to reproduce spatial and temporal perturbations. This is one of the main reasons that lead to the choice of this model. Since the perturbation model had to be entirely reproduced during this research, because of software compatibility reasons, it is believed that its description deserves a separate subsection.

3.6.3 EARTH GRAM-99 PERTURBATIONS MODEL

The perturbation model is the same for pressure, density, and winds; the only difference between the different parameters is the numeric values being used. For reasons explained in Section 3.7, only density perturbations are reproduced using this model.

According to Justus et al. [1995], atmospheric perturbations can be divided in two different kinds: the small-scale perturbations, due to turbulence, gravity waves, and other processes, and the large-scale perturbations, caused by tides and baroclinic wave processes. Both kinds of perturbations have their standard deviations normalized with respect to the mean, local, value, which are, respectively, $\sigma_{\rho,s}$ and $\sigma_{\rho,l}$; these are also taken from the sample given in Justus and Johnson [1999], and can be seen in Figure 3.7.

It was previously said that the ±50% variation in density, partly deduced from Figure 3.5, is con-

servative. Figure 3.7 proves this claim: in fact, it can be seen that the two perturbations have their standard deviation peak of about 10 % each at 100 km altitude. Between 50 km and 70 km the standard deviation is even lower.

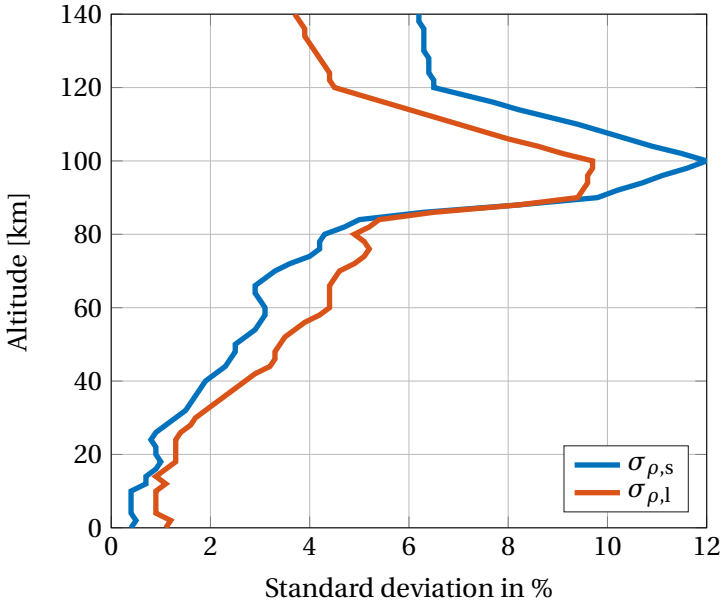


Figure 3.7: Density standard deviations in percentage of the mean value for both small and large scale perturbations, obtained from Appendix E of Justus and Johnson [1999].

LARGE-SCALE PERTURBATIONS

Large-scale perturbations have a sinusoidal form; the normalized (with respect to the average density at the given position and time) large-scale density perturbation ρ_l is:

$$\rho_l = \sigma_{\rho,l}\sqrt{2}A_Q \cos(n\tau + m\delta + 2\pi\frac{z}{\lambda_z} + 2\pi\frac{t}{T} + \phi_Q), \tag{3.43}$$

where z is the altitude in km, A_Q is an amplification factor, n and m are the number of horizontal waves, λ_z is the vertical wavelength, and T is a temporal wave period, and ϕ_Q is uniformly distributed randomized phase. It is clear that the perturbations has the form of a 4-dimensional wave, with a variable vertical wavelength λ_z. A_Q is computed as:

$$A_Q = 0.4808 + 0.96Q, \tag{3.44}$$

where Q is uniformly distributed between 0 and 1. n and m are also randomized, such that the number of waves is between 2 and 6:

$$m = n = round(4 + 0.833Q_{nm}), \tag{3.45}$$

where Q_{nm} is a Gaussian variable, with 0 mean and standard deviation of 1. The vertical wavelength is itself a function of the altitude (in km):

$$\lambda_z = a_v + 0.045\sqrt{\|z^3\|}, \tag{3.46}$$

where a_v is a randomized parameter, that dominates the size of the wavelength at the lowest altitudes. The report does not clarify how such random parameter is distributed; since it is not very relevant for higher altitudes, it was decided to set it between 0 and 1. Eventually, the standard deviation for the large-scale perturbation is also provided in Appendix E of [Justus and Johnson, 1999].

SMALL-SCALE PERTURBATIONS

The second part of the perturbations model concerns the small-scale perturbations, and is taken from [Justus and Johnson, 1999]. According to them, these perturbations can be modeled as an auto-correlated variable:

$$\mu\left(\mathbf{x}'\right) = r\mu\left(\mathbf{x}\right) + \sqrt{\left(1-r^2\right)}q, \tag{3.47}$$

where μ is the perturbation of the density normalized with respect to the standard deviation, $\mathbf{x}$ is the current position, $\mathbf{x}'$ is the next position and r is the autocorrelation value, which, by definition, implies:

$$r = <\mu\left(\mathbf{x}\right)\mu\left(\mathbf{x}\right)'>, \tag{3.48}$$

where the angle brackets are the mean operator. This property will be very useful when verifying the perturbation model in Subsection 6.2.5. r is a function of $\delta\mathbf{x} = \mathbf{x}' - \mathbf{x}$:

$$r(\delta\mathbf{x}) = \exp\left(-\delta h/L_h\right)\exp\left(-\delta z/L_z\right), \tag{3.49}$$

where h and z are, respectively, the horizontal and the vertical displacements, and L_h and L_z are their respective scale-sizes.

The scale-sizes only take either their maximum or their minimum values, which are a function of the altitude, and are very different between L_z and L_h. To decide whether to use the minimum or the maximum value, two artificial scale-sizes are introduced. The artificial scale-sizes also are a autocorrelated variables, and therefore follow Equation (3.47), and have their own scale size (which is the same for the L_z and L_h, and is a function of altitude only). In addition, however, they are cross-correlated with each other, by a cross-correlation factor r_c:

$$r_c = <\mu\left(\mathbf{x}\right)\nu\left(\mathbf{x}\right)> = \min(0.5 + 0.002z, 0.9), \tag{3.50}$$

where ν is the cross-correlated variable, and z is in km again.The artificial horizontal scale size evolve like μ, and the vertical scale size evolve like ν. For all of these cases, ν evolves as follows:

$$\nu\left(\mathbf{x}'\right) = r_\nu\nu\left(\mathbf{x}\right) + r_\mu\mu\left(\mathbf{x}'\right) + r_q q, \tag{3.51}$$

where r is the same as for μ, q is another Gaussian variable, and:

$$r_\nu = r\left(1-r_c^2\right)/\left[1-\left(r r_c\right)^2\right], \tag{3.52}$$

$$r_\mu = r_c\left(1-r^2\right)/\left[1-\left(r r_c\right)^2\right], \tag{3.53}$$

and

$$r_q = \sqrt{1-r_\nu^2-r_\mu^2-2r_\nu r_\mu r_c r}. \tag{3.54}$$

The model provides the values of the average and minimum scale sizes (again, as a function of the altitude), respectively L_{avg} and L_{min} , as well as their standard deviations σ_L. The probability of severe turbulence P_{sev} is defined as the probability of one fo the scale sizes to be smaller than the minimum scale size. From basic statistics, this is:

$$P_{sev} = \frac{1}{\sqrt{2\pi}}\int_{\frac{L_{avg}-L_{min}}{\sigma_L}}^{\infty}\exp\left(-\frac{u^2}{2}\right)du, \tag{3.55}$$

The model then also defines L_{max}, but being it extremely similar to L_{avg}, the latter will be used instead. The artificial scale size are then obtained by multiplying ν and mu by their standard deviation σ_L, and adding them to their average value L_{avg}. If either of the artificial scale sizes goes below L_{min}, then severe turbulence is triggered.

When in severe turbulence, the effective scale sizes are equal to L_{min}; when not, they are equal to L_{max}. Moreover, the normalized variance of the small-scale perturbation becomes $f_{sev}\sigma^2_{\rho,s}$; otherwise, it is $f_{non}\sigma^2_{\rho,s}$. $\sigma_{\rho,s}$ is the small-scale normalized standard deviation, whereas f_{sev} is function of the altitude, and is constant and equal to 12 for altitudes above 16 km; f_{non} is instead equal to:

$$f_{non} = \left(1 - f_{sev}P_{sev}/\left(1 - P_{sev}\right)\right). \tag{3.56}$$

Eventually, once the scale sizes are defined, depending on the artificial scale sizes, the autocorrlation r for the density can be computed, and the small-scale density perturbation ρ_s at $\mathbf{x}'$ is computed as:

$$\rho_s = \sqrt{f}\sigma_{\rho,s}\mu(\mathbf{x}'), \tag{3.57}$$

where f is either equal to f_{non} or f_{sev}, depending on whether severe turbulence is occurring or not.

By so doing, this model simulates turbulence with horiontal and vertical scale sizes that are function of the altitude. Moreover, with probability P_{sev}, the turbulence becomes severe: in that case, the scale sizes become much smaller, and therefore the variability becomes higher, and the standard deviation increases abruptly. This causes a discontinuity in the density, but it is exactly one of the goals of the developers of the model. Moreover, by using the artificial scale sizes, the severe turbulence is "patchy": this means that it usually does not last only one step. As an example, if there is a 1% chance that the turbulence is severe, that will happen in approximately 10 steps in a trajectory that lasts 1000 steps; however, those will not be 10 isolated steps, but most likely will come in "patches" of more than one step.

Once computed both ρ_s and ρ_l, these can be multiplied by the mean value of the density, and then added to it.

3.7 SIMPLIFYING ASSUMPTIONS

In this section, the assumptions made for the development of the simulator are justified. When deciding whether to include or not a force, the first consideration to be done is to see whether it affects the guided part of the entry, the unguided part, or both. Forces that affect the guided part have to be much larger to be included, since the vehicle is subject to larger forces in that part. If the forces affect the unguided phase, and specifically the exoatmospheric phase, then they can be of much smaller magnitudes to be included. This is because the unguided phase lasts much longer, up to half of an orbital revolution, than the atmospheric pass; moreover, no path corrections can be done during it.

3.7.1 HIGH ORDER COMPONENTS GRAVITY FIELD

The effect of the acceleration due to J_2 has been included in the simulator. This is because its effect, according to Wakker [2015], causes oscillations in the semi-major axis up to about 20 km for a highly inclined Low Earth Orbit. Also, it affects the exit leg, in which no control is possible. Hence, not including such a perturbation in the may lead to errors in apoapsis altitude of tens of kilometers. This is why Lu et al. [2015] even include it in their NPC[15].

Other components of the gravity field such as $J_{2,1}$, $J_{2,2}$, J_3, and so on do not have to be considered, simply because their magnitude is, on Earth, around one thousandth, or even less than, that of J_2 [Wakker, 2015].

[15] During this research, an NPC not including $J2$ was compared to an NPC that included it. For conditions where the latter led to errors of maximum 200 m, the former caused errors up to 8 km. The number of simulations done was very small; it is therefore possible that even larger errors may be caused if the $J2$ effect were not included in the prediction. This is shown in Subsection 4.4.1.

3.7.2 THIRD BODY PERTURBATION

Third body perturbations occur on Earth mainly because of the Sun and the Moon. Wakker [2015] estimates the effect of those perturbations on the semi-major axis as:

$$\Delta a \propto \frac{3}{2}\frac{\mu_d}{\mu}\frac{r^4}{r_d^3}, \tag{3.58}$$

where the subscript d stands for the disturbing body. Wakker [2015] estimates that the perturbation due to the Moon would cause an oscillation of the apoapsis of an Earth geosynchronous orbit of 1 km. A Low Earth Orbit would oscillate of 0.7 km instead. The perturbation due to the Sun has half the magnitude, and thus causes smaller perturbations. An error of less than 1 km in apoapsis causes an additional ΔV of about $1.5\,\mathrm{m\,s^{-1}}$. This sort of error should be kept in mind when analyzing the final results, but it is not worth including these perturbations during testing.

3.7.3 ATTITUDE CONTROLLER

In this research ideal attitude control is assumed. This not only includes angle-of-attack and sideslip angle control, which are always trimmed and zero respectively, but also bank-angle control.

Using a real controller for the bank angle would have the double drawback of requiring the design of an attitude controller, as well as that of affecting the performance of the guidance. Since this research is focused on the latter, it is preferred to have the theoretically best attitude controller, but that still respects the constraints of maximum angular acceleration and angular rate. This is because, as shown in Section 2.5, the depletion of energy during an aerocapture can be extremely fast. Its duration may be comparable to the duration of a bank reversal. Since attitude dynamics are not simulated, this effect is reproduced by a secondary guidance. Its design is described in Subsection 6.2.6.

Having a simulator with constrained angular velocity and accelerations can be very important in aerocapture. This is because, as an example, a bank reversal may last up to 15 s, and the duration of the atmospheric pass of an aerocapture can be as short as 150 s. It goes without saying that if the reversal and the dynamic pressure peak are concurrent, having a realistic reversal becomes really important.

3.7.4 ATMOSPHERE

An important simplification done in the atmospheric modeling consists of neglecting winds and winds perturbations. Winds have two effects: one is that of temporarily changing the aerodynamic angles; the other is that of changing the relative air-speed. The first one is automatically not considered due to the assumption of ideal attitude controller. Concerning the second one, a short discussion can be carried out.

At altitudes up to 100 km the maximum average wind speed found in Appendix E of Justus and Johnson [1999] is about $50\,\mathrm{m\,s^{-1}}$, with a standard deviation of $40\,\mathrm{m\,s^{-1}}$. This means that there is about a 1% chance of having a $150\,\mathrm{m\,s^{-1}}$ wind gusts during the guided flight. Assuming a speed at the end of the guided flight of $8000\,\mathrm{m\,s^{-1}}$, this perturbation causes a difference in drag and lift of about 4%. It is of course not negligible, but much smaller than the effect of density. Since the wind perturbations behave like the small-scale density perturbation, but are just much smaller, they are negligible (in comparison, density perturbations cause, with a 1% probability, differences in the aerodynamic force up to 30%). What is less negligible is the average wind, since it causes a systematic error. Nevertheless, because of the very high speed of aerocapture, such error is of less than 1% of the aerodynamic force. Always because of the quadratic relation between velocity and aerodynamic force, wind components not parallel to the velocity are even less relevant.

The last relevant simplification done with respect to the GRAM-99 model concerns the speed of sound: specifically, the non consideration of the temperature perturbation model. This is because

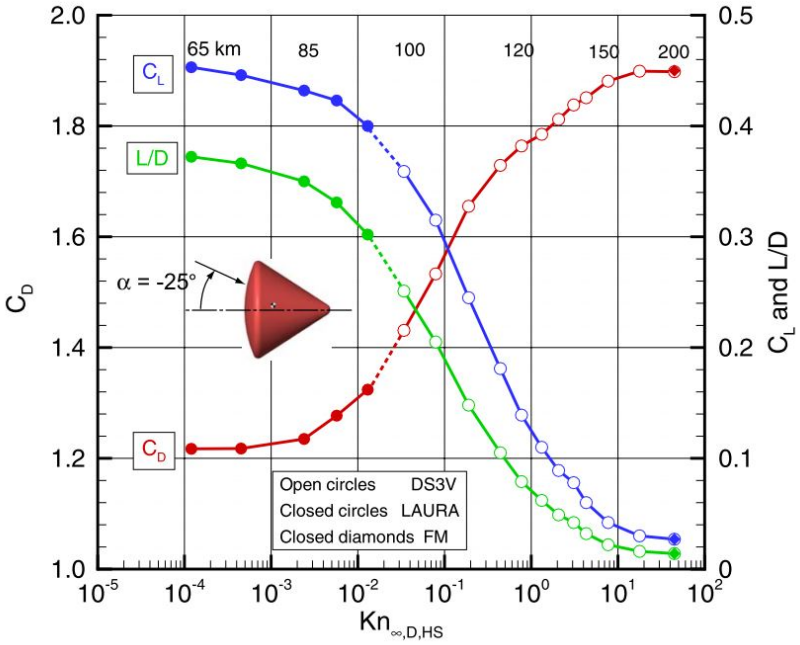


Figure 3.8: Lift to drag ratio and lift and drag coefficients for the Apollo capsule at an angle of attack of −25° as a function of Knudsen number (and with corresponding altitude, on Earth) [Moss et al., 2006].

of the speed of sound is almost unrelevant in aerocapture: according to NASA [1965], the lift-to-drag ratio changes of less than 15% between Mach 10 and Mach 30, and is assumed to be constant for higher Mach numbers. Even assuming a conservative value for the speed of sound of 400 m s^{-1} (which only occurs at altitudes higher than 120 km, where drag and lift are already almost zero), the ascending leg (the slower) would be flown at Mach 20. A variation of even 20% in Mach number due to temperature would cause changes in lift-to-drag ratio of only 3% (assuming a linear relation, which is probably not the case). In addition, during most part of the flight, any variations in speed of sound would not cause the Mach number to go below 30. Therefore, temperature perturbations will only be considered in the form of constant speed of sound variations, but no detailed model will be used.

3.7.5 RAREFIED FLOW

If one looks at Figure 3.8, it can be seen that a continuous flow solver has been used for Knudsen numbers smaller than 10^{-2} (which, on Earth, corresponds to about 95 km); direct Monte Carlo Simulation (an accurate method for solving rarefied flow aerodynamics) has instead been used for higher numbers. From that altitude to 120 km the lift-to-drag ratio drops almost linearly from 0.3 to 0.05, and then becomes almost to zero at an altitude of 200 km. There is no comparison between the continuous and the rarefied flow solvers solutions, and therefore part of the diminution of the lift-to-drag ratio could be due to changes in speed of sound. The change is rather substantial, but it happens at altitudes at which the aerodynamic force is already vary small. At 100 km, where the difference is already 15% of the lift-to-drag, but the density is still much larger than at higher altitudes, drag causes a deceleration of about 0.05 m s^{-1} (the ascending phase is considered, since in the descending phase small errors can be later corrected by the guidance). Such deceleration would only be revelant if it persisted for a considerable time; however, it is continuously decreasing instead, and can therefore be neglected.

In addition, since the lift-to-drag ratio decreases at higher altitude, it can be concluded that

attempting to guide the vehicle while at altitudes higher than 100 km is conceptually wrong: no lift implies indeed no controllability. This is in addition to the reduction in dynamic pressure.

3.8 EQUATIONS OF MOTION: ORBITAL FLIGHT

Orbital flight is here not needed to comprehend the exit part of the flight (which will be computed, both in the simulator and in the NPC, by using the equations for atmospheric flight), but to:

1. understand when/if apoapsis is reached;
2. compute the final ΔV;

The discussion concerns Keplerian orbit.

Skipping any derivation, a parametric solution to the two-body problem is [Wakker, 2015]:

$$r = \frac{a\left(1-e^2\right)}{1+e\cos\theta}. \tag{3.59}$$

Therefore, there is a relation between true anomaly θ and radius r. By setting $\theta = \pi$, one can check if this equation holds, given the radius and the semi-major axis. If it does hold, apoapsis is reached. In the two-body problem the integrals of motion have the following expressions:

$$E = \frac{1}{2}V^2 - \frac{\mu}{r} = -\frac{\mu}{2\,a}, \tag{3.60}$$

$$\mathbf{H} = \mathbf{V} \times \mathbf{r}, \tag{3.61}$$

where E is the specific energy and H is the specific angular momentum. A consequence of Equation (3.60) is the fact that, for a given orbit, speed and radius are biuniquely related:

$$V = \sqrt{\mu\left(\frac{2}{r} - \frac{1}{a}\right)}. \tag{3.62}$$

Consequences of Equation (3.61) are the fact that the motion is planar, and that:

$$r^2\,\dot{\theta} = \|\mathbf{H}\|. \tag{3.63}$$

The total energy E is negative for circular or elliptic orbits, it is zero for parabolic orbits and positive for hyperbolic orbits. Circular velocity is found when $r = a$:

$$V_c = \sqrt{\frac{\mu}{r}}. \tag{3.64}$$

3.8.1 ORBITAL TRANSFERS

The aerocapture maneuver ends with a two-burns orbital transfer. The transfer is ideally a periapsis raise only, but due to unavoidable errors, it will include an out-of-plane correction, together with an apoapsis correction half an orbital period later. For small angles, the out-of-plane correction consists of a burn perpendicular to the velocity vector; the periapsis raise and apoapsis correction consist instead of burns in the direction of the velocity. Because of the Pythagorean theorem, it is most convenient to apply the perpendicular burn during the largest of the two parallel burns, which is the periapsis raise. The amount of that ΔV_{lat} needed to correct the inclination is, for small angles [Wakker, 2015]:

$$\Delta V_{lat} = 2\,V\,\sin\left(\frac{\Delta i}{2}\right), \tag{3.65}$$

where Δi is the inclination error, and V is the orbital velocity at burn. In theory, it is hard to define what exact value of the velocity should be taken, since it varies during the periapsis burn. The initial velocity is taken: since the difference in velocity is relatively small, and this ΔV_{lat} is also small, only a second-order error is done.

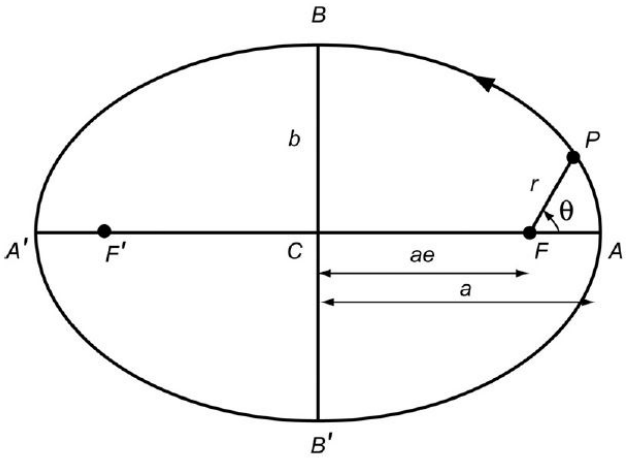


Figure 3.9: Geometry of an elliptical orbit [Wakker, 2015].

3.9 EQUATIONS OF MOTION: ATMOSPHERIC FLIGHT

The equations to propagate the motion in a planetary atmosphere are needed in Cartesian, inertial coordinates, as well as in spherical, relative (in the $\mathcal{R}$ frame) coordinates, for the reasons stated in Section 3.3. Spherical, relative coordinates are common in three different forms: dimensional, dimensionless, and dimensionless using energy instead of time as an independent variable. The last set of equations will be referred to as the "energy-independent" set of equations. A trade-off between the three sets of equations using spherical, rotating coordinates, will be carried out in Section 6.3.1. This is necessary to understand which set of equations is best suited for the propagator in the NPC.

3.9.1 INERTIAL, CARTESIAN

The motion in inertial, Cartesian coordinates is the direct application of Equation (3.19). This, together with the kinematics in Cartesian coordinates, implies:

$$\begin{aligned} \frac{d\mathbf{V}_{\mathcal{I}}}{dt} &= \frac{\mathbf{F}_{\mathcal{I}}}{m}, \\ \frac{d\mathbf{x}_{\mathcal{I}}}{dt} &= \mathbf{V}_{\mathcal{I}}. \end{aligned} \tag{3.66}$$

Both equations have three components, and are first order differential equations.

As previously stated, the only two forces acting on the vehicle are gravity and the aerodynamic force. How those are computed has been explained throughout this chapter.

Gravity is computed directly using inertial, Cartesian coordinates, using Equations 3.23 and 3.26.

The aerodynamic force needs to be rotated instead: in fact, in its form as in Equation (3.30), it is expressed in the aerodynamic reference frame. A force in the aerodynamic reference frame needs first to be rotated into the vertical frame, then in the rotating plane and eventually in the inertial plane. Using Equations 3.4 to 3.7:

$$\mathbf{F}_{A,\mathcal{I}} = \mathbf{C}_{\mathcal{I},\mathcal{R}}\,\mathbf{C}_{\mathcal{R},\mathcal{V}}\,\mathbf{C}_{\mathcal{V},\mathcal{A}}\,\mathbf{F}_{A,\mathcal{A}}. \tag{3.67}$$

By so doing, one obtains $\mathbf{F}_{\mathcal{I}}$, to be substitute in Equation (3.19):

$$\mathbf{F}_{\mathcal{I}} = \mathbf{F}_{A,\mathcal{I}} + m\mathbf{g}_{\mathcal{I}}, \tag{3.68}$$

where, in this case, $\mathbf{g}$ is the gravitational acceleration vector, including the J_2 term.

3.9.2 RELATIVE, SPHERICAL

The relation between rotating, spherical equations and inertial, Cartesian, is a two-step transformation of the inertial, Cartesian equations. This set is the consequence of a rotation of reference frame, from inertial to rotating, and of a state representation transformation, from Cartesian to spherical.

It is reminded that, when doing a frame transformation between an inertial frame and a rotating frame that are not coincident with the point where the forces are applied (as this case), apparent forces appear.

The equations of motion are here reported, for unpropelled flight, and including the J_2 term [Lu et al., 2015]:

$$\begin{aligned}
\dot{V} &= -\frac{D}{m} - g_r \sin\gamma - g_\delta \cos\gamma\cos\chi + \omega_{cb}^2 r\cos\delta(\sin\gamma\cos\delta - \cos\gamma\sin\delta\cos\chi),\\
V\dot{\gamma} &= \frac{L\cos\sigma}{m} - g_r\cos\gamma + g_\delta\sin\gamma\cos\chi + 2\omega_{cb}V\cos\delta\sin\chi + \frac{V^2}{r}\cos\gamma + \\
&\quad + \omega_{cb}^2 r\cos\delta(\cos\gamma\cos\delta - \sin\gamma\sin\delta\cos\chi),\\
V\cos\gamma\dot{\chi} &= \frac{L\sin\sigma}{m} + g_\delta\sin\chi + 2\omega_{cb}V(\cos\gamma\sin\delta - \sin\gamma\cos\delta\cos\chi) + \\
&\quad + \frac{V^2}{r}\cos^2\gamma\tan\delta\sin\chi + \omega_{cb}^2 r\cos\delta\sin\delta\sin\chi,\\
\dot{r} &= V\sin\gamma,\\
\dot{\tau} &= \frac{V\sin\chi\cos\gamma}{r\cos\delta},\\
\dot{\delta} &= \frac{V\cos\gamma\cos\chi}{r},
\end{aligned} \tag{3.69}$$

where g_r and g_δ are obtained with Equations 3.27[16].

Because of the ever-increasing computational capabilities of current on-board computers there is no need to simplify these equations more for an NPC guidance.

3.9.3 DIMENSIONLESS EQUATIONS

The same previous set of equations can be made dimensionless. This is usually done because dimensionless equations may have computational advantages when propagating.

Length is normalized by the equatorial radius of the planet, R_E, velocity is normalized by

$$V_{scale} = \sqrt{\mu/R_E}, \tag{3.70}$$

and time is normalized accordingly.

It is then simply sufficient to normalize all the equations, by opportunely dividing them according to their dimensions. As an example, the derivative of the dimensionless velocity with respect to dimensionless time is (the symbol $\bar{\square}$ stand for normalized variable):

$$\begin{aligned}
\frac{d\bar{V}}{d\bar{t}} &= \frac{d(V/V_{scale})}{d(t/t_{scale})} = \\
&= \left[\frac{t_{scale}}{V_{scale}}\left(-\frac{D}{m} - g\sin\gamma\right) + \bar{\omega}_{cb}^2\bar{r}\cos\delta(\sin\gamma\cos\delta - \cos\gamma\sin\delta\cos\chi)\right].
\end{aligned} \tag{3.71}$$

[16] It shall be specified that it is still possible to use an ellipsoid model for the planet while using these equations. Simply stated, the density is computed using the altitude above the reference ellipsoid, but the variables are still in spherical coordinates. Hence, as an example, the flight-path angle would be with respect to the vertical frame for a sphere, instead of the vertical frame of the ellipsoid. Again, this would imply $V\sin\gamma \neq \dot{h}$.

As previously mentioned, the last set of equations can be obtained by removing time. It is then possible to reduce the system by one equation, using nondimensional negative energy $\bar{E}$ as an independent variable instead of $\bar{t}$:

$$\bar{E} = \frac{1}{\bar{r}} - \frac{1}{2}\bar{V}^2. \tag{3.72}$$

This energy is evaluated in the rotating reference frame. Therefore, the velocity in that relation is the one with respect to the rotating planet. At this point, the equation for the velocity can be neglected, since, where needed, velocity can be computed using:

$$\bar{V} = \sqrt{(2/\bar{r} - \bar{E})}. \tag{3.73}$$

A small error is done when doing this. First, this energy does not include the effect of J_2. Second, in such a potential the centrifugal force is not included[17]. Nevertheless, the magnitude of both these effects is rather small.

The derivative of the energy with respect to a certain reference frame can be computed as the dot product between the sum of all the non-conservative forces and the velocity in that frame. Since lift is perpendicular to the motion, only drag need to be considered:

$$\frac{d\bar{E}}{d\bar{\tau}} \approx \bar{D}\bar{V} > 0. \tag{3.74}$$

Energy is monotonic for an non propelled flight, since drag and velocity are both always positive by definition. Hence, it is possible to use it as an independent variable; consequently, all the above mentioned equations can be simply rewritten as:

$$\frac{d\mathbf{x}}{d\bar{E}} = \mathbf{f}(\mathbf{x}, \sigma, \bar{E}), \tag{3.75}$$

by multiplying the right-hand side of the dimensionless equations by $1/\bar{D}\bar{V}$.

A trade-off will be carried out in Section 6.3.1 to evaluate the best set of rotating, relative coordinates for the propagator of the NPC.

3.10 SUMMARY

The motion of a vehicle in an atmosphere is rather complex, and depends on a variety of models and variables. For the research to be possible, it is necessary to make a few, reasonable assumptions.

The vehicle will be treated as a point mass, with atmospheric coefficients depending on Mach number only. The bank angle is treated as a parameter subject to angular velocity and acceleration constraints. The mean atmosphere is modeled according to either the US76 Atmosphere or a profile extracted from the GRAM-99 model. Additionally, two atmospheric perturbations models have been included from the latter. The planet has an ellipsoid shape. Gravity acceleration include only the central term and the component due to J_2.

[17] In rotating reference frames, centrifugal force should be included into the potential energy. This is done, as an example, to solve the restricted three-body problem, and to find the surfaces of Hill [Wakker, 2015].

4

OPTIMAL AEROCAPTURE NPC GUIDANCE

In this chapter more insight into the atmospheric part of the aerocapture maneuver is given. However, before doing that, the orbital part is treated, since it is the one that sets the requirements at the exit of the atmospheric flight.

Constraints and optimality of the trajectory are analyzed in Sections 4.2 and 4.3. Later on, the optimal aerocapture guidance by Lu et al. [2015] is reproduced. A few modifications to their concept are proposed, and a comparison of the results obtained during this research with those from their paper is shown. The reader can have a look at Chapter 6, to see how the simulator has been developed and verified. Some conclusions about the aerocapture trajectory and the modified NPC are given at the end of the chapter.

4.1 OPTIMAL PERIAPSIS RAISE

In any astrodynamics problem, one usually wants either to minimize the initial mass, or maximize the payload mass. These two statements are equivalent.

In aerocapture, two trajectory-dependent parameters affect the initial mass: one is the ΔV needed to raise the apoapsis; the other is heat load, which partly drives the mass of the heat shield. Unfortunately, there is usually no equation that relates the mass of the heat shield to the heat load explicitly, except for some empirical and inaccurate relations. Hence, the EDL problem is usually a (constrained) multi objective optimization.

To minimize the ΔV, it is necessary to properly understand the exoatmospheric phase, and the periapsis raise. The goal of this section is to properly describe them.

In this research, only the case in which the final target orbit is circular is considered. In fact, as explained in Chapter 2, the case for an elliptical orbit is less interesting.

4.1.1 SINGLE-BURN STRATEGY

The burns in aerocapture, as explained in Section 3.8.1, are two. In this phase of the analysis, the perpendicular component of the first burn is not considered yet,since, ideally, the optimal aerocapture ends in the target plane. In this specific subsection, only the single-burn strategy is considered.

If one sets the apoapsis of the transfer orbit as a constraint, and assumes it to be exact at the end, then it is easy to understand that the fastest is the velocity at apoapsis, the smallest is the ΔV. The velocity at apoapsis is a function of the orbital energy, which, in turn, for a given apoapsis, depends on the periapsis altitude. Therefore, the transfer orbit should have the target periapsis and, at the same time, the highest possible periapsis. Such periapsis, by definition, cannot be higher than the exit altitude[1], since, after that, the trajectory is quasi Keplerian.

[1] the definition of exit altitude is very subjective: in this case, it is meant the altitude after which no relevant control is

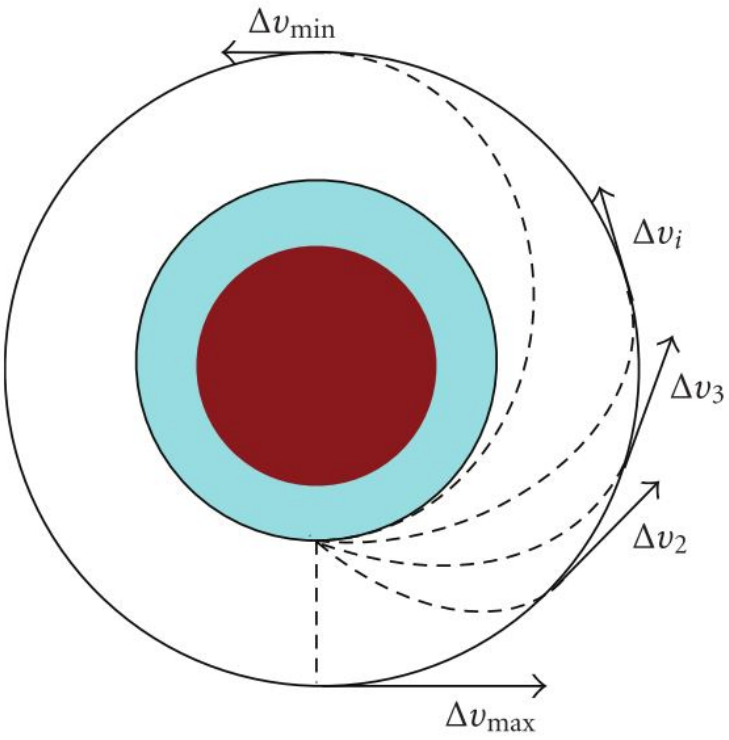


Figure 4.1: Sketch of periapsis raise ΔV [Armellin and Lavagna, 2008].

All the feasible transfer trajectories cross the imaginary sphere of the altitude exit with a flight-path angle that is either 0° or larger. Figure 4.1 is helpful in visualizing this. For a flight-path angle of 0° the corresponding periapsis altitude is maximum, and the ΔV is minimum. Given the following relationships for eccentricity and semi-major axis [Wakker, 2015],

$$e = \sqrt{1 - \frac{r_0 V_0}{\mu}\left(2 - \frac{r_0 V_0}{\mu}\right)\cos\gamma^2}, \tag{4.1}$$

$$a = \frac{r_0}{2 - \frac{r_0 V_0}{\mu}}, \tag{4.2}$$

it can be proven that the relation between flight-path angle at a specific altitude and energy is monotonic: in fact, maximum energy implies maximum semi-major axis, which, in turn, because of the constraints of the apoapsis (which implies $a(1+e) = r_{apo}$), requires minimum eccentricity. With $2 - r_0 V_0^2/\mu$ always positive for closed orbits, minimum eccentricity requires minimum flight-path angle. Therefore, minimizing the ΔV for a one-burn transfer can be translated into minimization of exit flight-path angle, together with apoapsis targeting constraint.

4.1.2 TWO-BURN STATEGY

The situation becomes a little more complicated if a 2-burn strategy is used. At this point, a derivation begins, that aims to see what orbital parameters are most relevant to minimize the in-plane

possible anymore

ΔV. Simply applying the energy equation, the ΔV for a 2-burn transfer is:

$$\begin{aligned}\Delta V = \sqrt{2\mu}\left\|\sqrt{\frac{1}{r_{a,0}}-\frac{1}{r_{a,0}+r_{p,0}}}-\sqrt{\frac{1}{r_{a,0}}-\frac{1}{r_{a,0}+r_{apo}}}\right\| + \\ +\left\|\sqrt{\frac{1}{r_{apo}}-\frac{1}{r_{a,0}+r_{apo}}}-\sqrt{\frac{1}{r_{apo}}-\frac{1}{2\,r_{apo}}}\right\|,\end{aligned} \tag{4.3}$$

where $r_{p,0}$ and $r_{a,0}$ are, respectively, the periapsis and apoapsis of the exit orbit, and r_{apo} is the target orbit apoapsis (and radius, being the orbit circular).

The first term is minimized for large $r_{p,0}$ (up to r_{apo}, which cannot be reached anyway), and for largest $r_{a,0}$; however, the second term is minimized for $r_{a,0} = r_{apo}$. The first derivative with respect to $r_{a,0}$ can be computed, keeping in mind the previous considerations to solve the absolute values:

$$\frac{\partial \Delta V}{\partial r_{a,0}} = \sqrt{2\mu}\left[\frac{\left(\frac{1}{r_{a,0}+r_{apo}}\right)^2-\left(\frac{1}{r_{a,0}}\right)^2}{2\sqrt{\frac{1}{r_{a,0}}-\frac{1}{r_{a,0}+r_{apo}}}}-\frac{\left(\frac{1}{r_{a,0}+r_{p,0}}\right)^2-\left(\frac{1}{r_{a,0}}\right)^2}{2\sqrt{\frac{1}{r_{a,0}}-\frac{1}{r_{a,0}+r_{p,0}}}}+\frac{\left(\frac{1}{r_{a,0}+r_{apo}}\right)^2}{2\sqrt{\frac{1}{r_{apo}}-\frac{1}{r_{a,0}+r_{apo}}}}\right]. \tag{4.4}$$

Assuming that all the values in the square roots are relatively similar to each other, one obtains the following simplification:

$$\frac{\partial \Delta V}{\partial r_{a,0}} = \sqrt{2\mu}\frac{\left(\frac{1}{r_{a,0}+r_{p,0}}\right)^2}{2\sqrt{\frac{1}{r_{a,0}}-\frac{1}{r_{a,0}+r_{p,0}}}}. \tag{4.5}$$

This derivative goes to 0 for $r_{a,0}$ going to infinity; moreover, in that case, ΔV would be a maximum.When changing the sign of the second absolute value, a solution is obtained for $r_{a,0} = 0$, which is also a maximum. This means that the minimum can only lie in the point where there is no derivative, when $r_{a,0} = r_{apo}$. Therefore, for a given periapsis of the transfer orbit, the optimum case for a 2-burn strategy is a single-burn transfer, with apoapsis targeting. This has also been evaluated numerically. This proof does not include, however, out-of-plane burns nor transfer at non-zero flight-path angle.

This derivative is also useful in determining the sensitivity of the ΔV with respect to errors in targeting. For $r_{p,0} = R_E$, the sensitivity is $0.3\,\mathrm{m\,s^{-1}\,km^{-1}}$ (for $r_{a,0} > r_{apo}$; opposite sign for the other case).

The derivative with respect to $r_{p,0}$ is of similar magnitude, but negative:

$$\frac{\partial \Delta V}{\partial r_{p,0}} = -\sqrt{2\mu}\frac{\left(\frac{1}{r_{a,0}+r_{p,0}}\right)^2}{2\sqrt{\frac{1}{r_{a,0}}-\frac{1}{r_{a,0}+r_{p,0}}}}. \tag{4.6}$$

For the same case as before, it is equal to -0.3 $\mathrm{m\,s^{-1}\,km^{-1}}$. Therefore, a a trajectory that has a 0 km miss in the apoapsis might be less efficient than a trajectory that misses the apoapsis by 5 km, but does so while having a periapsis of the transfer orbit that is 10 km higher than that of the aforementioned trajectory.

Therefore, it is proven that optimality is not coincident with accuracy: however, the two are strictly related.

4.2 IN-FLIGHT CONSTRAINTS

During the atmospheric flight, the spacecraft is subject to many constraints. All those constraints are strictly related to the trajectory, since they are function of the flight conditions, mainly density and velocity. In general, those are:

- heat flux;
- derivative of the heat flux;
- load factor (which, for approximately constant aerodynamic coefficients, is proportional to the dynamic pressure);
- angular velocity (in a bank-modulated trajectory, this is just to $\dot{\sigma}$), and angular acceleration ($\ddot{\sigma}$).

The first two were already discussed in Section 3.5. The load factor is the total acceleration due to non-gravitational forces. In general it includes both propulsive and aerodynamics forces, but in the case of non propelled flight, it reduces to:

$$n = \frac{\sqrt{L^2 + D^2}}{m\, g_0} = \frac{\sqrt{C_L^2 + C_D^2}}{m\, g_0} \bar{q} S_{ref} \leq n_{max} \tag{4.7}$$

Quite similar to the load factor is the dynamic pressure constraint. All the constraints named so far are functions of the trajectory and the environment and, in the case of the load factor, also of the instantaneous attitude of the spacecraft. The remaining two, instead, are constraints on the control variables. A limit in angular velocity is also usually due to structural limits, whereas a constraint in angular acceleration might be due to either structural limits or to limited propulsive system. Thus:

$$|\dot{\sigma}| \leq |\dot{\sigma}|_{\max} \tag{4.8}$$

$$|\ddot{\sigma}| \leq |\ddot{\sigma}|_{\max} \tag{4.9}$$

How these will be respected, while still having an optimally controlled attitude, is the topic of Subsection 6.2.6. The reason why the constraints concerning the attitude rate and acceleration were not stated in the mission requirements of Section 2.7 is because they are not enforced by the guidance.

The presence for so many trajectory-related constraints requires the guidance system to be able to predict whether these constraints will be violated at some point during the flight, and have a consequent reaction well in advance.

4.3 OPTIMAL AEROCAPTURE

Optimality of a trajectory depends on what the objective function is. In the case of aerocapture, two cost functions of interest are the ΔV and the total heat load. Also of interest are in-flight constraints, specifically those concerning maximum heat flux and maximum load factor.

In this section, the derivation done by Lu et al. [2015] of the optimal trajectory that minimies the ΔV is reported. This derivation makes use of optimal control theory and is valid for a simplified, planar trajectory.

Later on, two proofs that are, at the best of the author's knowledge, novel, are given. First the proof done by Sigal and Guelman [2001] for the minimum convective heat load trajectory is generalized to include any kind of heat load (hence, radiative heat load as well). Second, it is shown how the minimum ΔV trajectory implies minimization of peak load factor and heat flux.

4.3.1 MINIMUM ΔV TRAJECTORY

In this subsection, the proof by Lu et al. [2015] is reported. Any part of the derivation shown here is motivated by the authors of the proof. After the derivation, some considerations are also given.

Lu et al. [2015] derive the minimum ΔV trajectory for a planar trajectory on a non-rotating celestial body with a central gravity field. Because of these assumptions, Equations 3.69 reduce to (together with their respective initial conditions):

$$\begin{aligned}
\dot{r} &= V\sin\gamma, \qquad r(t_0) = 0,\\
\dot{V} &= -D - \frac{\mu\sin\gamma}{r^2}, \qquad V(t_0) = V_0,\\
\dot{\gamma} &= \frac{1}{V}\left[L\cos\sigma + \left(V^2 - \frac{\mu}{r}\right)\frac{\cos\gamma}{r}\right], \qquad \gamma(t_0) = \gamma_0.
\end{aligned} \tag{4.10}$$

σ is the only control variable, subject to the constraint:

$$0 \le \sigma_{\min} \le \sigma \le \sigma_{\max} \le \pi. \tag{4.11}$$

The proof is valid for a re-entry problem in which constraints and performance index are only function of a subset of the state vector, which they call $x_{lon} = \left(r\ V\ \gamma\right)$, and in case the only control variable is the bank-angle. The angle-of-attack should be a prescribed function. The proof is carried out for any function $J = \eta(\mathbf{x}_{lon}(t_f))$ of the final state, and for any constraints on the final conditions $\mathbf{s}(\mathbf{x}_{lon}(t_f)) = 0$, where $\mathbf{s}$ has dimension $k \le 3$. It is therefore clear that a normal entry problem, in which also the horizontal distance is among the constraints, cannot be included.

The Hamiltonian can then be written:

$$H = \lambda_r V\sin\gamma + \lambda_V\left(-D - \frac{\mu\sin\gamma}{r^2}\right) + \lambda_\gamma\left[\frac{L}{V}\cos\sigma + \left(V^2 - \frac{\mu}{r}\right)\frac{\cos\gamma}{rV}\right], \tag{4.12}$$

where, according to the Pontryagin Maximum Principle:

$$\dot{\lambda}_r = -\frac{\partial H}{\partial r} = \lambda_V\left(\frac{\partial D}{\partial r} - \frac{2\mu\sin\gamma}{r^3}\right) - \lambda_\gamma\frac{\partial\dot{\gamma}}{\partial r}, \tag{4.13}$$

$$\dot{\lambda}_V = -\frac{\partial H}{\partial V} = -\lambda_r\sin\gamma + \lambda_V\frac{\partial D}{\partial V} - \lambda_\gamma\frac{\partial\dot{\gamma}}{\partial V}, \tag{4.14}$$

$$\dot{\lambda}_\gamma = -\frac{\partial H}{\partial\gamma} = -\lambda_r V\cos\gamma + \lambda_V\frac{\mu\cos\gamma}{r^2} - \lambda_\gamma\frac{\partial\dot{\gamma}}{\partial\gamma}. \tag{4.15}$$

The optimal bank-angle σ^* has to be such that, at any moment, the Hamiltonian H is maximized:

$$\sigma^* = \arg\max_\sigma\left\{\lambda_r V\sin\gamma + \lambda_v\left(-D - \frac{\mu\sin\gamma}{r^2}\right) + \lambda_\gamma\left[\frac{L}{V}\cos\sigma + \left(V^2 - \frac{\mu}{r}\right)\frac{\cos\gamma}{rV}\right]\right\}. \tag{4.16}$$

Since $\cos\sigma$ is monotonic in $\sigma \in [\sigma_{min}, \sigma_{max}]$, and $L/V > 0$, the optimal bank-angle can be different from its extrema only if $\lambda_\gamma \equiv 0$.

Lu et al. [2015] now prove that such case cannot happen for a finite interval of time by contradiction. For λ_γ to be constant and equal to zero for a finite time, it is required to have $\lambda_\gamma = \dot{\lambda}_\gamma = 0$. This implies, substituting $\lambda_\gamma = 0$ in Equation (4.15):

$$\lambda_V\frac{\mu}{r^2} - V\lambda_r = 0, \tag{4.17}$$

since $\cos\gamma \neq 0$. In addition, being this a free-time problem, according to the Pontryagin Maximum Principle again, the transversality condition $H \equiv 0$ holds along the entire trajectory. If λ_γ is set equal to zero, the latter transversality condition requires:

$$\lambda_r V \sin\gamma - \lambda_V D - \lambda_V \frac{\mu \sin\gamma}{r^2} = 0. \tag{4.18}$$

Because of Equation (4.17), Equation (4.18) requires that $\lambda_V = 0$, since $D > 0$. Now, $\lambda_V = 0$ implies $\lambda_r = 0$. At this point, all the costate variables are equal to zero, and this is a contradiction to the Maximum Principle, which states that the costate variables cannot all be zero. Therefore, Lu et al. [2015] successfully prove that singular control cannot happen for any trajectory with free final time and only final constraints and objectives.

The fact that the final conditions, nor the objective, do not include cross-range is what allows this proof, and is a key difference between a normal entry problem and some aeroassisted problems. As seen, this fact has large implications from an optimality point of view.

This proof is the base of the optimal aerocapture guidance by Lu et al. [2015]. Generalizing this to a 3-dimensional motion in a rotating atmosphere is a little complicated, and is not done mathematically. However, two considerations can help. The first consists of the fact that the accelerations neglected when using planar motion are rather small. The largest one is the Coriolis which, despite not negligible, is still much smaller than drag and lift[2]. The second consideration is of numerical nature. Miele et al. [1990] optimized the aerocapture trajectory in a rotating planet, and found that the trajectory minimizing the final ΔV is very close to a bang-bang trajectory. This is far from being a mathematical proof, but strengthens the concept that the proof obtained for planar motion in a non rotating atmosphere is at least a good and valid indication to be kept into account when optimizing the 3-dimensional trajectory in a rotating atmosphere.

Finally, a similar result is also numerically obtained by in Section 6.6, where a trajectory is optimized to verify the algorithm for reinforcement learning.

4.3.2 MINIMUM TOTAL HEAT-LOAD AEROCAPTURE

This proof is, at the best of the author's knowledge, novel. It is a generalization of the work by Sigal and Guelman [2001], who proved that the minimum convective heat load aerocapture trajectory is bang-bang. Here, their proof is extended to include any possible formulation for the heat flux $f(\rho, V)$ that is a smooth function of density and velocity only. It is indeed important to extend the proof to heat fluxes other than the convective one because, for example, radiative heat flux becomes very large at the high speeds of aerocapture.

The Hamiltonian is written:

$$H = f(\rho, V) + \lambda_r V \sin\gamma + \lambda_V \left(-D - \frac{\mu \sin\gamma}{r^2}\right) + \lambda_\gamma \left[\frac{L}{V}\cos\sigma + \left(V^2 - \frac{\mu}{r}\right)\frac{\cos\gamma}{rV}\right], \tag{4.19}$$

where $f(\rho, V)$ is any smooth, monotonically increasing function of the density and the velocity, and might be (as it is of interest here) the sum of convective and radiative heat fluxes at cold wall. It is noticed that a singularity occurs for $\gamma \equiv 0$, and a singular arc for $\dot{\gamma} = \gamma = 0$ is seeked. Equation (4.17) holds, since $f(\rho, V)$ is not a function of γ. Now, setting the Hamiltonian equal to zero because of the transversality condition, one obtains, for the singular arc:

$$f(\rho, V) - \lambda_V D = 0. \tag{4.20}$$

[2] This is not true in the upper layers of the atmosphere. In such conditions, however, the trajectory is better approximated by Keplerian motion. During Keplerian motion, the orbital parameters do not change, and therefore neither does the objective function. The transition phase, in which the Coriolis force is comparable to drag and lift, but Keplerian motion cannot be assumed yet, is instead relatively short.

This, together with Equation (4.17), imply:

$$\lambda_V = \frac{f(\rho,V)}{D} \tag{4.21}$$

$$\lambda_r = \frac{\mu f(\rho,V)}{r^2 DV} \tag{4.22}$$

The derivative with respect to time of Equation (4.21) is:

$$\dot{\lambda}_V = \frac{\partial f(\rho,V)}{\partial t}\frac{1}{D} - f(\rho,V)\frac{\dot{D}}{D^2} = \frac{\partial f(\rho,V)}{\partial \rho}\frac{\partial \rho}{\partial t}\frac{1}{D} - f(\rho,V)\frac{\dot{D}}{D^2}. \tag{4.23}$$

These equations can be further expanded considering the following:

$$\frac{\partial \rho}{\partial t} = -\rho\beta V\sin\gamma, \tag{4.24}$$

$$\dot{D} = \rho V S_{ref} C_D + 1/2 V^2 S_{ref} C_D \frac{\partial \rho}{\partial t} = \frac{2D}{V} + \frac{D}{\rho}\frac{\partial \rho}{\partial t}. \tag{4.25}$$

Hence, Equation (4.23) becomes:

$$\begin{aligned}\dot{\lambda}_V = &-\frac{\partial f(\rho,V)}{\partial \rho}\frac{\rho\beta V\sin\gamma}{D} + \\ &+\frac{\partial f(\rho,V)}{\partial V}\frac{\left(-D-\mu/r^2\sin\gamma\right)}{D} + \frac{f(\rho,V)}{\rho D}\rho\beta V\sin\gamma - \frac{2f(\rho,V)}{VD}(-D-\mu/r^2\sin\gamma).\end{aligned} \tag{4.26}$$

According to the costate equations, the derivative of λ_V must also satisfy (setting already $\lambda_\gamma = 0$):

$$\dot{\lambda}_V = -\frac{\partial f(\rho,V)}{\partial V} - \lambda_r\sin\gamma + \lambda_V\frac{2D}{V}. \tag{4.27}$$

Setting Equation (4.27) equal to Equation (4.23), and using Equations 4.21 and 4.22:

$$\begin{aligned}&-\frac{\partial f(\rho,V)}{\partial \rho}\frac{\rho\beta V\sin\gamma}{D} + \frac{\partial f(\rho,V)}{\partial V}\left(\frac{-D-\mu/r^2\sin\gamma}{D}+1\right) + \\ &+ f(\rho,V)\left[+\frac{\rho\beta V\sin\gamma}{\rho D} - \frac{2}{VD}\left(-D-\mu/r^2\sin\gamma\right) + \frac{\mu\sin\gamma}{r^2 DV} - \frac{2}{V}\right] = 0.\end{aligned} \tag{4.28}$$

A possible solution is $\gamma = \pm\pi/2$, which is not acceptable[3]. Simplifying, and dividing the entire equation by $\sin\gamma \neq 0$ and $D > 0$, one has a more clear picture of the situation:

$$-\frac{\partial f(\rho,V)}{\partial \rho}\rho\beta V + \frac{\partial f(\rho,V)}{\partial V}\left(-\frac{\mu}{r^2}\right) + f(\rho,V)\left(\frac{3\mu}{r^2 V} + \beta V\right) = 0. \tag{4.29}$$

At this point, two possibilities for the singular arc still exist. The first, is that a trajectory is flown, such that, depending on the formulation of the heat flux, the former equation is satisfied. Nevertheless, it adds a unique relation between altitude (through ρ and r) and velocity, which is, in many cases, impossible. As an example, if one sets $f(\rho,V) \propto \sqrt{\rho}\,V^2$, the relation is a constant velocity at any altitude, which is obviously infeasible (this specific example implies also that, if one were interested to minimize the integrated load-factor, a bang-bang trajectory would be optimal). Therefore, it can

[3]The main reason why such a situation is not acceptable is that $\gamma = \pm\pi/2$ cannot last for finite period of time for a lifting body. In fact, the lift would cause the flight-path angle to have a finite derivative with respect to time.

simply be checked whether the formulation of the heat flux allows a feasible altitude-velocity profile; if it does not, then the optimal control trajectory is, again, bang-bang.

The second possibility for the singular arc to exist occurs if the heat flux has a formulation such that Equation (4.29) is identically equal to zero for any value of the density and the velocity. Since, by definition, $f(\rho, V)$ cannot be a function of β, μ nor r, this only happens if:

$$\frac{\partial f(\rho, V)}{\partial \rho} = \frac{f(\rho, V)}{\rho}, \tag{4.30}$$

$$\frac{\partial f(\rho, V)}{\partial V} = 3\frac{f(\rho, V)}{V}. \tag{4.31}$$

This means that singular arc cannot happen, since the only function satisfying Equations 4.30 and 4.31 is $f(\rho, V) \propto \rho V^3$. No relations for the heat flux used in this thesis have such a formulation[4]. Instead, if the function $f(\rho, V)$ to be minimized were such that only one, (or none) between Equations 4.30 and 4.31 holds, singular control cannot happen. As an example, this is what happens for the convective heat flux. In fact, if one substitutes $f(\rho, V)$ with the Tauber-Sutton relation, the case studied by Sigal and Guelman [2001] occurs.

In conclusion, it has been proved that, except for extremely unlikely conditions, any trajectory minimizing the total heat load should be of the bang-bang kind. It depends on the heat flux function itself, whether the lift control history should be up-down or down-up. In the case of convective heat flux only, Sigal and Guelman [2001] proved that such a trajectory should be up-down. It will be shown, in Subsection 4.8.5, that there are good empirical indications for it to be opposite when only radiative heat flux is considered. Eventually, it is likely that when both are considered, the minimum heat load trajectory is case dependent.

4.3.3 MINIMUM LOAD FACTOR AND CONVECTIVE HEAT-FLUX PEAKS TRAJECTORY

At the best of the author's knowledge, a derivation of the aerocapture trajectory that minimizes both load factor and convective heat rate peaks is novel.

It is generally not necessary to look for the trajectory that minimizes the constraints: nevertheless, minimizing the constraints implies that, when possible, the constraints will always be satisfied. Impossibility depends on the initial conditions and on the final target.

In addition, it will be shown that such a trajectory is the same as the one described in Subsection 4.3.1. Therefore, there is no loss in performance when striving to minimize these constraints.

Because of Equation (4.7), the problem becomes that of finding the control history $u(t)$ such that:

$$\min_{u(t)} \left[\max_{t} (\bar{q}) \right] \tag{4.32}$$

In addition, minimizing the convective heat flux, as approximated by the Detra-Hidalgo equation for cold-wall, is equivalent to the following:

$$\min_{u(t)} \left[\max_{t} \left(\sqrt{\rho} V^{3.15} \right) \right] \tag{4.33}$$

In this case, the proof will be carried out with partial aid of numerical methods. Therefore, it cannot be generalized. The proof will be shown for aerocapture with initial velocity of $13\,000\,\mathrm{m\,s^{-1}}$ on Earth. In this case, the equations of motion from Subsection 3.9.1 will be used. The atmosphere will be the average unperturbed US76.

The derivation consists of four steps:

[4] This does not mean that such a formulation cannot exist. It is possible that empirical relations, that might include, as an example, ablation, or coupling between ablation and convection, or so on, may still be $f(\rho, V) \propto \rho V^3$. It is very unlikely though, since it is difficult for an empirical relation to have exactly integer exponents.

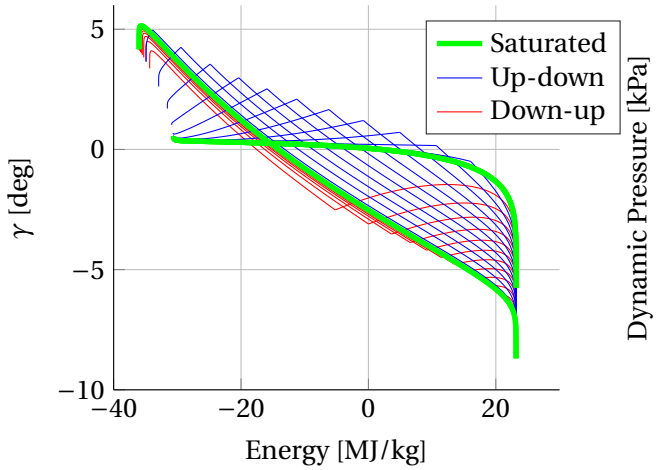


Figure 4.2: Lift-up lift-down and lift-down lift-up trajectories, in the $E-\gamma$ plane.

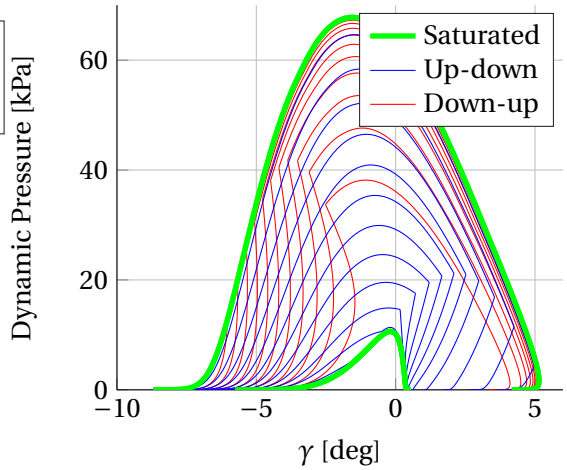


Figure 4.3: Lift-up lift-down and lift-down lift-up trajectories, in the $\gamma-\bar{q}$ plane.

1. Proving that, on what concerns the constraint peaks, the ascending leg can be neglected (considering an atmosphere whose density only varies in altitude).

2. Showing that the up-down bang-bang trajectory is the one that always has the least negative flight-path angle (during the entire descending leg).

3. Proving that having the least negative flight-path angle leads to smaller future dynamic pressure and convective heat flux.

4. Proving that the latter step implies minimizing the peaks for both dynamic pressure and convective heat flux peaks during the entire trajectory.

Of these, the only step making use of numerical simulations is the second, and it is likely that, at least for planar motion, it is also provable mathematically.

The first step is very easy to prove. During the ascending leg, velocity is decreasing, since, in a non-rotating celestial body, $\dot{V} = -D/m - g\sin\gamma$. In a rotating body, the Coriolis acceleration due to the rotation of the body is perpendicular to the velocity, hence it cannot change its magnitude; forces due to centrifugal acceleration are instead negligible on Earth. Also, the density is also decreasing. Therefore, being both the dynamic pressure and the convective heat flux monotonic functions of both density and velocity, their maximum can never be in the ascending leg. As a consequence, the derivation can be limited to the descending leg of the trajectory.

The second step is probably the core of the derivation. It might be possible to prove this mathematically, but here it is done numerically. An aerocapture corridor is generated including all the possible boundary trajectories that can lead to the target apoapsis. In this case, the corridor is seen as a three-dimensional space whose dimensions are relative velocity, altitude and flight-path angle, which are the three variables included when analyzing planar motion. However, one might substitute velocity and altitude with dynamic pressure and energy. Dynamic pressure could also be substituted with the convective heat flux. The boundary surface consists of lift-up lift-down and lift-down lift-up trajectories, starting with the range of allowed initial flight-path angles[5]. They consist of the boundaries of the three-dimensional corridor because, before the switch, the command

[5]Allowed initial flight-path angles are those that are between the minimum and the maximum initial flight-path angle. For aerocapture, these are always obtained by looking for, respectively, a full lift-up and a full lift-down trajectory that, starting from the set initial conditions (all except for the flight-path angle) lead to the desired apoapsis altitude. This reasoning implies that by allowed, the constraints are not taken into account.

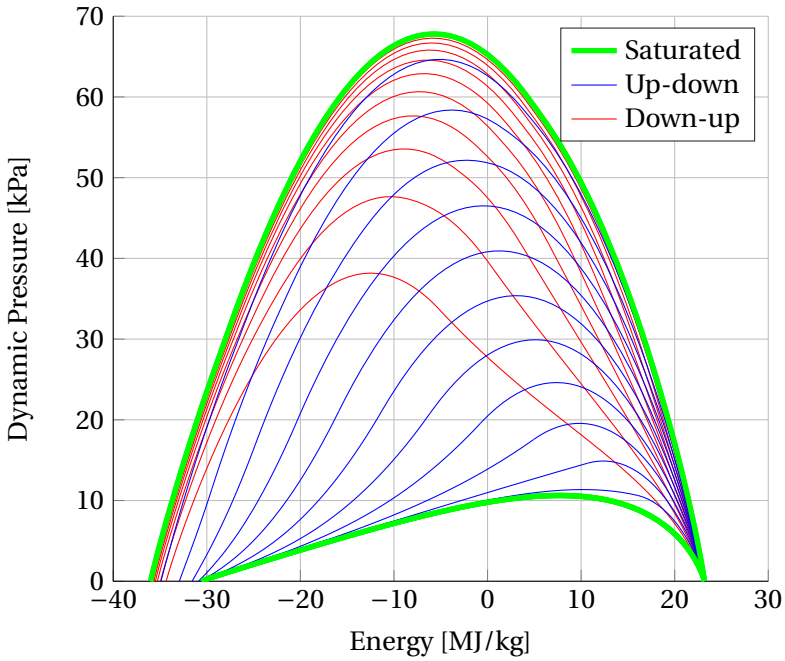


Figure 4.4: Lift-up lift-down and lift-down lift-up trajectories, in the $E - \bar{q}$ plane.

is saturated, and, after the switch, the only way to achieve the desired apoapsis is to have the command saturated in the opposite direction. A hypothetical vehicle inside of such a three-dimensional corridor could not get out of it before crossing the surface defined by the ensemble of the points in which the bank is switched, and could not go back inside if it were outside of it after that surface is crossed (this is true for planar motion on a non-rotating planet).

Figures 4.2 to 4.4 show such a corridor in all its three dimensions, two at a time. For any point on the $E - \bar{q}$ of Figure 4.4, if it is met by having flown an up-down trajectory instead of a down-up trajectory, the corresponding flight-path angle is the shallowest. This fact can be intuited from Figures 4.2 and 4.3, but can only be shown while displaying a three-dimensional, moving plot, which is not possible on paper. The intuition goes as follows: following a constant $\bar{q}$ or a constant E line in one of those plots, one can find that most trajectories that are encountered first are down-up[6]: this fact implies that those trajectories are always steeper. Again, this, as described here, is just an intuition. Nevertheless, even though it is not possible to see it here, if Figure 4.4 were three-dimensional, and the Z-axis were the flight-path angle (increasing in the direction of the reader), all the down-up trajectories would not be visible, because, for equal E and $\bar{q}$, they cause a steeper (smaller, in the sense more negative) flight-path angle.

This is very important, because, as shown by Lu [2014], a shallow flight-path angle implies smaller derivatives of both dynamic pressure and convective heat flux. This proves that, in any moment, the lift-up lift-down trajectory is the one that minimizes the future load peaks: hence, since this is valid at any moment in time, the peaks are minimized for the entirety of the trajectory. Moreover, the lift-up control is the control that maximizes the derivative of the flight-path angle. This completes the proof. It was found that the same happens for convective heat flux.

[6]This is for the descending leg only, corresponding to the point in which the flight-path angle is negative. For previously discussed reasons, there is no need to consider the situation in which the flight-path angle is positive.

4.3.4 SUMMARY

So far, it has been shown, thanks to the proof by Lu et al. [2015], that the minimum ΔV is obtained with a bang-bang, lift-up lift-down, trajectory. The proof is limited to flight in a non-rotating atmosphere, but this is not very limiting, and numerical simulations have shown that in many case the same is valid in a rotating atmosphere.

Also, it was mathematically proven, using the Pontryagin Maximum Principle, that a bang-bang trajectory minimizes also the integral of any function of density and velocity (unless such a function is proportional to ρV^3, which is not an interesting case). In the case of a function such as the convective heat rate (computed with the Sutton-Graves relation) the case reduces to the one investigated by Sigal and Guelman [2001], who showed that the integral is minimized by a lift-down lift-up trajectory. This is opposite to the trajectory minimizing the ΔV. It will be shown, in Subsection 4.8.5, that there are good indications that the trajectory minimizing the radiative heat-flux is lift-up lift-down, and that the trajectory minimizing the sum of convective and radiative heat-flux is case dependent. Also this proof is limited to a non-rotating atmosphere, but also in this case, numerical simulations reported in Subsection 4.8.5 show that, at least in this case, the assumption of a non-rotating atmosphere is not a problem.

Eventually, it has been shown how the same trajectory minimizing the ΔV also minimizes the peaks of both dynamic pressure and convective heat flux. This was done with aid of numerical methods, and therefore one should be careful when generalizing this result.

4.4 OPTIMAL AEROCAPTURE GUIDANCE

Lu et al. [2015] developed an optimal guidance that minimizes the final ΔV by flying a bang-bang trajectory as shown in Subsection 4.3.1. They developed four different modes for this guidance, but only Mode 1 will be reported here, the others being less or equally performing, and quite equivalent in concept. Some minor modifications are brought to this concept, which are reported in the following subsection. The guidance with these minor modifications will be referred to as *Lu*.

In addition, some major modifications are also brought to the concept. These concern the numerical methods, and are given in Subsection 4.4.2, and the propagation, and are given in Subsection 4.5.

4.4.1 ORIGINAL OPTIMAL AEROCAPTURE GUIDANCE

The main concept underlying the optimal aerocapture guidance consists of planning the trajectory as divided into two phases. In the first phase, the capsule flies at a constant bank-angle σ_0. Ideally, such a value should be equal to 0°, but it is instead slightly larger to allow lateral maneuverability. During the entirety of this research, σ_0 will be set very close[7] to 0°, and equal to 15°, as done in Lu et al. [2015]

In the second phase, a constant, as large as possible, bank-angle σ_d is planned. In an ideal world, σ_d =180° would be optimal. However, such a planning causes the bank-angle to have no margins for correction. Hence, σ_d has to be chosen to be smaller than 180° for reasons of robustness. Optimality for a deterministic, simplified system may be very far from optimality in a more complex, stochastic environment. This is particularly true for aerocapture, in which the optimal trajectory in a deterministic, simplified environment is bang-bang. Hence:

> σ_d has to be chosen as a trade-off between optimality and robustness.

Lu et al. [2015] show how the choice of σ_d affects both parameters. Figure 4.7 shows that the final in-plane ΔV decreases for increasing σ_d. This leads to saturation of the control, as seen in 4.8, which

[7] 15° are not close to 0°, but the cosine of 15° is equal to 0.966, very close to 1. Hence, the difference in vertical component of the lift between a bank-angle of 0° and one of 15° is less than 3.5 %. There is instead a large difference in lateral component of the lift, since the sine of the bank-angle goes from 0 to 0.259.

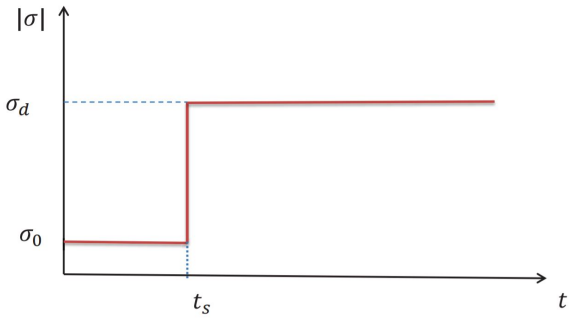


Figure 4.5: Bank-angle planning during Phase 1 in the original guidance [Lu et al., 2015].

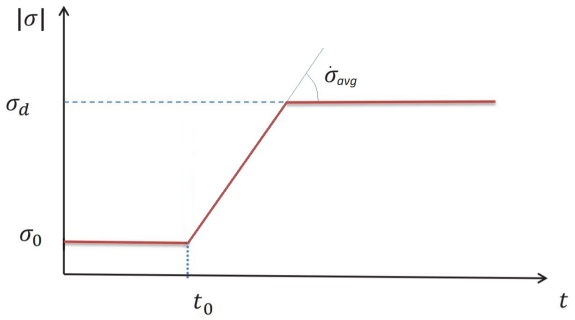


Figure 4.6: Bank-angle planning during Phase 1 in the *mod* guidance.

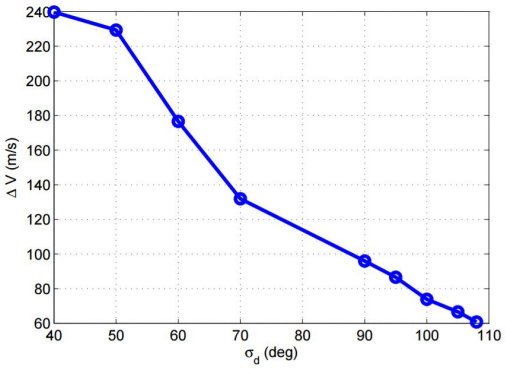


Figure 4.7: Variation of apogee orbital insertion ΔV versus σ_d for Orion MPCV for a 200 km apoapsis altitude orbit [Lu et al., 2015]. Mission conditions as described in Lu et al. [2015].

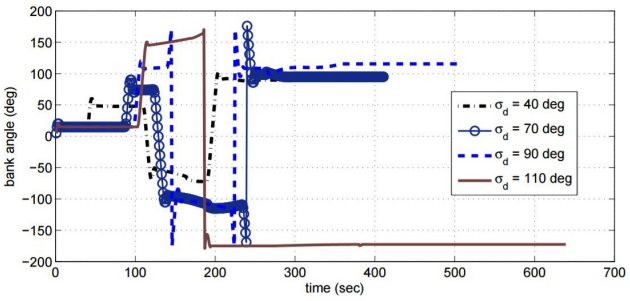


Figure 4.8: Closed-loop bank-angle profiles for different values of σ_d [Lu et al., 2015]. Mission conditions as described in Lu et al. [2015].

in turn causes decreases in robustness impossibility of lateral control. Hence, after some values of σ_d, any benefits in terms of in-plane ΔV become negligible with respect to the disadvantages in out-of-plane ΔV. For even larger values of σ_d, also the in-plane ΔV would start increasing.

Figures 4.9 and 4.10 show the same trajectories of Figure 4.8 in the $h-t$ and $h-V$ planes, respectively. Keeping in mind Figures 2.5 and 2.6 from Section 2.5.1, it is evident how a large value for σ_d makes the trajectories more similar to a full lift-down trajectory. As shown in that same subsection, a full lift-down trajectory leads to a final flight-path angle that is very close to zero which, in turn, leads to a lower ΔV, as explained in Section 4.1.

Eventually, they show how they had to manually tune it for different entry conditions. For this reason, σ_d is a function of initial flight path angle and velocity[8]. The concept for Phase 1 is also shown in Figure 4.5. In Figure 4.6 is instead shown the *mod* concept, which will be described in Subsection 4.5.1; such a figure is reported here to facilitate the comparison between the concepts, but will be discussed later.

The high-level algorithm of their method is reported in the frame below.

Original optimal aerocapture guidance by Lu et al. [2015].

[8]It is likely that the optimal trade-off for σ_d would be a function of many other parameters as well. Such a relation was not investigated, nor here, nor by Lu et al. [2015]

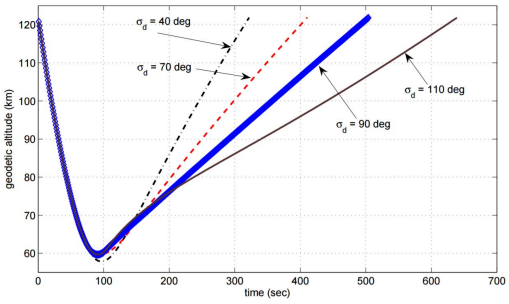


Figure 4.9: Variation of apogee orbital insertion ΔV versus σ_d for Orion MPCV for a 200 km apoapsis altitude orbit [Lu et al., 2015]. Mission conditions as described in Lu et al. [2015].

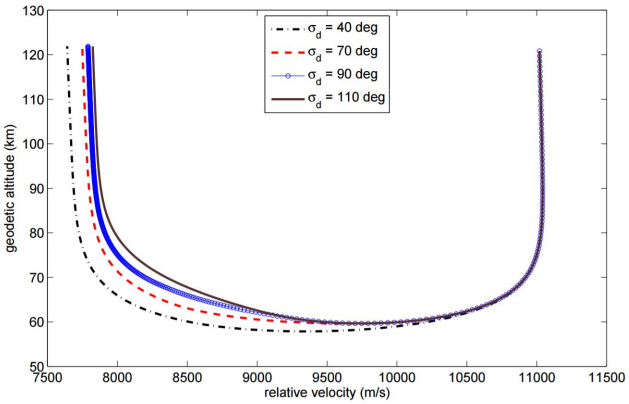


Figure 4.10: Closed-loop bank-angle profiles for different values of σ_d [Lu et al., 2015]. Mission conditions as described in Lu et al. [2015].

At iteration k:

1. Phase 1:
   (a) interpolate $\sigma_d = f(\gamma_0, V_0)$;
   (b) plan trajectory as in Figure 4.5;
   (c) solve[9,10] $r_{apo}(t_s) = r^{\star}_{apo}$;
   (d) **if** $t > t_s$, go to Phase 2;
   else, set $\sigma_{cmd} = \sigma_0$, and end iteration k.
2. Phase 2:
   (a) plan trajectory with constant σ;
   (b) solve $r_{apo}(\sigma^{\star}) = r^{\star}_{apo}$, with predictive load-relief active;
   (c) set[11] $\sigma_{cmd} = \sigma^{\star}$, and end iteration k.

This algorithm can be complemented with the flow-chart in Figure 4.14, which is given in Section 4.6, concerning the specific architecture of the guidance logic. Such a figure refers specifically to the guidance modified in this research, but it is still helpful in understanding the work by Lu et al. [2015].

During Phase 1 the guidance logic computes, at each call, the time t_s at which the switch between Phase 1 and Phase 2 should occur, based on current conditions, to target the apoapsis. During Phase 1, the predictor integrates a trajectory in which the bank-angle is equal to σ_0 up to t_s and is then equal to σ_d afterwards (refer again to Figure 4.5). It does so by using Brent's method [Brent, 1973], which is a combination of bisection, and linear and quadratic interpolations. Eventually, when the current time is larger than the computed t_s, Phase 2 is triggered.

In this research, the Phase 1 is reproduced as an integration at constant bank-angle, equal to σ_d; if the predicted apoapsis is larger than the target, Phase 2 is triggered. Otherwise, nothing happens. This is equivalent to what Lu et al. [2015] do, except for causing a delay of maximum one second (which is the frequency of the guidance call); however, one second of delay is not relevant if compared to the average time needed to rotate the capsule attitude from σ_0 to σ_d.

During Phase 2, the guidance logic iterates the propagation of motion to find the constant bank-

angle that would lead to target the apoapsis. Again, Brent's method is used, and the extrema of the search space are always the entirety of the bank-angle domain (either $0 \leq \sigma_d \leq \pi$ or $-\pi \leq \sigma_d \leq 0$).

For reasons that will be explained in Section 6.3.1,the equations of motion used are dimensionless spherical relative, in which time is used as independent variable[12]. In addition, to compute the density, the altitude with respect to the reference ellipsoid is used also in the prediction. As a last remark, the model of the numerical predictor corrector uses the US76 atmospheric model (for the density only), but corrects the density as follows:

$$\rho_L = L/L^\star, \qquad \rho_D = D/D^\star, \tag{4.34}$$

where L and D are the sensed accelerations, and $L^\star$ and $D^\star$ are the accelerations estimated using the US76 Standard Atmosphere model, and nominal mass and aerodynamic coefficients (and sensed velocity). To reduce possible noise, due by either the sensors or some density perturbations, both ρ_L and ρ_D are filtered:

$$\tilde{\rho}_L^{(k+1)} = \tilde{\rho}_L^{(k)} + \left(1-\beta\right)\left(\rho_L - \tilde{\rho}_L^{(k)}\right); \tag{4.35}$$

(same happens for ρ_D). After a small batch of reduced Monte Carlo runs, in which both small scale and large scale perturbations were included, it was decided to set $\beta = 0.95$. The lift acceleration is then computed by multiplying, the nominal lift acceleration by $\tilde{\rho}_L^{(k+1)}$, along the entire predicted trajectory, at the guidance call $k+1$; the drag acceleration is instead multiplied by $\tilde{\rho}_D^{(k+1)}$.

This guidance is reproduced here, with some minor modifications, as a benchmark; also, a few major modifications are proposed in the next subsections. The discussion about the numerical method is carried out first. Then, a test and comparison of the results from Lu et al. [2015] is also done.

A this point, a short discussion about the lateral logic is necessary. The lateral logic used by Lu et al. [2015] is specific to the algorithm, but is never described in their paper. Thus, it is preferred to not include it when reproducing the work by Lu et al. [2015], since using a different lateral logic from the original one might affect the performance. Later on, another lateral guidance will be used in the modified version of the NPC.Such a lateral guidance is described in Subsection 4.5.2.

Another difference consists of the following. As previously mentioned, Lu et al. [2015] set the σ_d of each trajectory as a function of initial flight-path angle, as well as entry velocity. The function is done by interpolation of manually tuned results. However, it is specific to the vehicle, which, in their case, is Orion MPCV. Since in this work Apollo Command Module is used, their tuning would not be appropriate. Moreover, such a tuning may be very demanding, and will therefore not be done in this research.

The last modification consists of propagating the orbit in the NPC with the full equations of motion, and especially including the effect of J_2. In fact, it was seen during the initial phases of testing that assuming Keplerian motion from atmospheric exit until apoapsis would lead to errors up to 8 km. A small sensitivity analysis showed that such an error would be mainly caused by the effect of J_2.

Eventually, the minor modifications to the original work are summarized in the box below:

***Lu* concept: guidance with only minor modifications to the original guidance by Lu et al. [2015]**

1. The algorithm does not compute t_s in Phase 1, but only checks if the propagation with constant $\sigma = \sigma_d$ leads to a low or a high apoapsis (or to a crash, or a hyperbolic trajectory).

[12] In the original work, the dimensionless equations of motion are used, which use energy ad independent variable instead of time

This is justified because the time to turn the bank-angle from σ_0 to σ_d is much larger than 1 second.

2. The solution in Phase 2 is not obtained by using Brent's method, but by using the bisection-secant method, described in Subsection 4.4.2. The two methods do not perform very differently, but the bisection-secant is much simpler to implement.

3. The predictive load-relief is not included[13], nor in Phase 1 nor in Phase 2. This design choice is a strict consequence of what was shown in Subsection 4.3.3.

4. The lateral guidance is not included. This is because the original paper does not describe what lateral guidance is used. From a conceptual point of view, this is not a great issue.

5. The function $\sigma_d = f\left(\gamma_0, V_0\right)$ used here is obtained by interpolation of the work by Lu et al. [2015]. In their work, the Orion MPCV is used, whereas here the Apollo Command Module is used. Because of different ballistic coefficient and lift-to-drag ratio, it is believed that the optimal function would be different for the Apollo Command Module. Such a difference is minor from a conceptual point of view, but the consequent difference in performance may not be small.

6. Dimensionless equations, with time as independent variable, instead of energy, are used. This is a consequence of a trade-off done in Section 6.3.1.

7. Motion is not assumed to be Keplerian from atmospheric exit until apoapsis. This is not a minor modification, since it leads to much higher accuracy. Nevertheless, it is a difference that was implemented in all of the guidance systems.

From now on, when referring to the *Lu* guidance, it will be meant a guidance with these modifications with respect to the original one by Lu et al. [2015]. It is stressed that, among all the concepts implemented during this research, the *Lu* is the concept that most resembles the original work by Lu et al. [2015].

4.4.2 NUMERICAL METHODS

In the work here reproduced, numerical methods only concern Phase 2. This is a strict consequence of the first minor modification listed in the box describing the *Lu* concept.

As previously mentioned, Lu et al. [2015] use Brent's method [Brent, 1973]. The method is extremely reliable, in that it is certain to find the root for which the function changes sign. The solution $f(x) = 0$ needs not even to exist, and a simple change of sign is sufficient.

In Lu et al. [2015], the guidance needs on average between 10 and 20 iterations to compute the bank-angle with an accuracy of 10^{-6} radians[14].

Brent's method is a good solution, but it has a few drawbacks. The first is the fact that the method, in certain cases, may need as many as three times the iterations needed by bisection to converge [Brent, 1973]. When aiming for a reliable system, it is better to rely on a method that always takes the same, maybe large, number of iterations (as bisection does), rather than a method that often needs only a small number of iterations, but sometimes needs a very large number of iterations.

The remaining problems concerning the method can be understood by analyzing Figure 4.11, and are all related to the fact that Brent's method makes no use of the knowledge of the function.

[14] The information was obtained via personal communication with P. Lu, on March 2, 2016

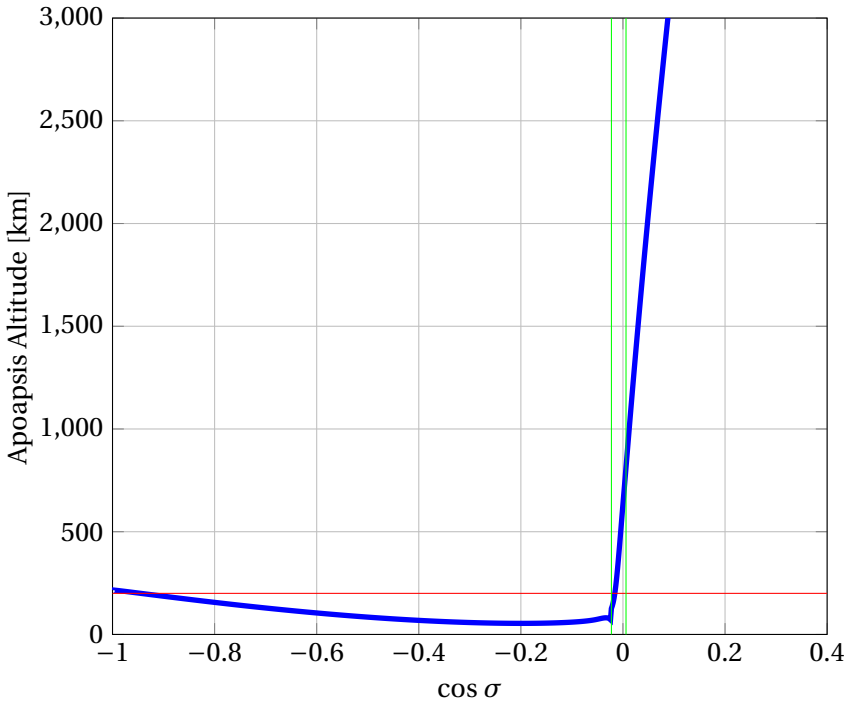


Figure 4.11: Final predicted apoapsis as function of the cosine of the bank-angle.

The figure shows the relation between $\cos\sigma$ and the altitude of the apoapsis. First, it is seen that the function has a large portion of the domain in which the function is almost constant. That happens because the guidance logic interrupts the prediction, since crashing conditions occur. Where the function to be solved is constant, Brent's method brings no advantages with respect to bisection. Also, it can be seen that, on the left, an additional solution occurs. That happens because of the stopping conditions again, which, among many, include the flight-path angle[15]. With $\cos\sigma = -1$ the flight becomes very steep very soon, such that the prediction is interrupted when the orbital energy is still large enough to obtain an apoapsis higher than target. This is, of course, a false positive. Similar to this, is the case in which the guidance predicts a hyperbolic exit trajectory: the parabolic trajectory between the closed and open final orbits would be seen by Brent's method as an additional solution.

The problems are of two natures, and are summarized below:

1. Brent's method may be, in some cases, much slower than bisection.

2. Additional checks are needed to verify if the solution obtained is not a false positive. False positives may occur at both ends of the domain.

Technically, the second problem is solved by modifying the function, rather than the numerical method.

Concerning the first problem, two novel[16] methods are proposed:

1. The *bisection-secant* method: a simplified version of Brent's method, but more reliable in terms of numbers of maximum iterations;

[15] This may or may not happen in the guidance designed by Lu et al. [2015], since the details of their design are not given. It happens in the guidance designed in this work though.

[16] It is debatable whether these methods are really novel. A few methods have been combined together in both cases, together with smaller features that fit the problem of aerocapture.

2. The *modified damped Newton* method: a theoretically less reliable method, which only needs two iterations at a time, opposed to the tens needed by the previous method and Brent's method. Efforts of empirical nature were made to find the formulation that suits the problem the best.

BISECTION-SECANT METHOD

The first method consists of using bisection until both extrema are in the neighborhood of the target apoapsis. After that, the secant method is used. The boundaries for the use of the secant method are, on the lower side, the set exit altitude, and on the upper side, the target apoapsis altitude plus ten times the difference between the exit altitude and the target apoapsis altitude. An upper limit is needed because of the asymptotic behavior of the target apoapsis. Using the secant method to solve a function that has an asymptote may lead lead to computation time much larger than bisection.

The algorithm is stopped when the predicted apoapsis has an error of 100 m or less, or after 40 iterations. The method also stops if a bank-angle of zero leads to a low apoapsis (or a crash), or if a bank-angle of 180° leads to a high apoapsis (or a hyperbolic trajectory).

This method is justified by looking once again at Figure 4.11. Such relation is close to linear in proximity of the solution. The part in which the solver uses a secant method is the one between the two green lines. It is not large (in this case), but it is enough make the numerical method faster than simple bisection.

MODIFIED DAMPED NEWTON METHOD

The second method consists of a damped version of Newton-Raphson. The derivative of the predicted apoapsis with respect to the bank-angle is approximated by finite difference. The initial guess for the bank-angle is the bank-angle commanded at the previous iteration. Because of the discontinuities of the target function, a small perturbation could cause the previously commanded bank-angle to lead to a trajectory that brings to a crash in the following prediction.

A simpler solution to this could be that of commanding a larger bank-angle every time this happens. However, it is problematic to quantify how much larger the bank-angle should be. It happened in many attempts that a constant increase in the bank-angle was either too small, leading to a final crash of the outer loop trajectory, or too large, leading to a largely oscillating situation. This, eventually, does not lead to a crash, but to much larger than expected fuel consumption. Also, it is a solution far from being optimal. Therefore, a damping term is included in the Newton-Raphson equation, such that:

$$\sigma^{(k+1)} = \sigma^{(k)} - \alpha_1 \frac{dh_{apo}/d\sigma}{\Delta h_{apo}}. \tag{4.36}$$

α_1 is a coefficient that changes dynamically. After some trial and error, it turned out that a good function for α_1 is:

$$\begin{aligned} \alpha_1^{(k+1)} &= \alpha_1^{(k)} \frac{5^o}{\Delta\sigma} \qquad \text{if } \Delta\sigma \le 5^o, \\ \alpha_1^{(k+1)} &= \alpha_1^{(k)} \times 30^{1/6} \qquad \text{otherwise}, \\ 0.03 \le \alpha_1^{k+1} &\le 1 \qquad \text{always}, \end{aligned} \tag{4.37}$$

where

$$\Delta\sigma = \left\| \sigma^{(k+1)} - \sigma^{(k)} \right\|. \tag{4.38}$$

This damping is used for when there is a large difference $\Delta\sigma \ge 5°$ between the previously commanded bank-angle and the currently commanded one. The formulation is such that α_1 goes back to 1 in about maximum 6 seconds if the difference between the two commanded bank-angles is small enough. In fact, α_1 is bounded between 0.03 and 1.

On the other hand, if the prediction leads to a crash, the cosine of the commanded bank-angle is increased. To avoid possible overshoots, however, the increase happens very slowly. Again, a formulation that turned out to be successful is the following:

$$\sigma^{k+1} = \sigma^k - 15^o \alpha_1 \alpha_2, \tag{4.39}$$

where

$$\alpha_2^{(k+1)} = \min\left(\alpha_3 e^K, 1\right) \tag{4.40}$$

where $\alpha_3 = 1/1000$, and K is the number of consecutive guidance calls in which the prediction has crashed. By so doing, there is quite some damping in case of some navigation errors or large dynamic pressure perturbations, but after 5 seconds of continuous non convergence, there occurs a commanded difference in bank-angle that is about 2.5°, which is the maximum bank-angle difference that can be actuated if the initial derivative of the bank-angle is close to zero (as it usually is), since the maximum angular acceleration is set to be $5\,°\,s^{-2}$. Of course, this damping is risky, since, in case the error in the prediction is not due to any noise of any kind, the correction begins with a delay of about 5 seconds. Nevertheless, its benefits are larger: in fact, it is highly unlikely that a trajectory that is predicted to normally achieve the apoapsis, suddenly becomes a crashing trajectory, whereas it often happens that because of noise a prediction may turn out to be wrong, and predict a crash.

As a last remark, an additional feature is necessary. To increase the stability of the method, the propagation shall always begin from the current, sensed, bank-angle, and then linearly move to σ_{cmd}. This is important during large shifts such as the switch between the two phases, or during the reversal. If such a modification were not included, the initial guess at each iteration would always be too far from the solution, causin instability. Such modification is implemented only during Phase 2, since in Phase 1 the current bank-angle is equal to σ_0. This is also the reason why the modified Newton method can only work with either the *mod* or the *modlat* concepts, described in the next section.

4.5 CONCEPTUAL MODIFICATIONS TO THE OPTIMAL AEROCAPTURE NPC

Two main modifications are done to the optimal aerocapture NPC (in addition to the two numerical methods). One is the lateral guidance, as mentioned in Section 4.4, which is described in Subsection 4.5.2, and another one is the bank-angle profile used in the prediction, described in the following subsection.

4.5.1 FINITE BANK-ANGLE RATE (*mod* CONCEPT)

The main difference between this concept and the original one consists of the fact that, in this one (from now on labeled *mod*) no discontinuities ever occur in the predicted bank-angle. To better understand this, one may refer to Figure 4.6 (reported in Subsection 4.4.1 to facilitate the comparison with the *Lu* concept). As an example, the bank-angle linearly shifts from the value of Phase 1 to the value of Phase 2 with a constant angular velocity[17] $\dot{\sigma}_{avg} = 10.5°$. The angular acceleration is instead infinite in the prediction. This concept is applied in Phase 1 by including a linearly changing bank-angle between σ_0 and σ_d. In Phase 2, this is only applied to the latter.

Such a modification, despite seemingly minor, gives major benefits. Specifically, as will be shown in Section 4.7, it acts in such a way that no manual tuning of $\sigma_d = f(\gamma_0, V_0)$ is needed, as it is instead in the *Lu* concept. In that section it will be shown that, in same conditions, at the end of the transition between Phase 1 and Phase 2, the bank-angle of the *mmod* guidance stabilizes with

[17] The value for such an angular rate has been estimated analytically, and is equal to approximately the total time needed to rotate the spacecraft from $\sigma_0 = 15°$ to 120°, divided by the angular distance between the two. The total time is computed assuming $\ddot{\sigma} = 5\,°\,s^{-2}$ and $\dot{\sigma} = 15°\,s^{-1}$. For vehicle with different characteristics, or for different missions, a different value of $\dot{\sigma}_{avg}$ shal be computed.

a difference from σ_d that is about 18 times smaller than that of the *Lu* concept. Such a difference is very important for two reasons:

1. It makes the concept more robust: the vehicle behaves much more closely to what is predicted. In other words, the prediction is more realistic.

2. It makes the guidance free of the demanding tuning needed to find $\sigma_d = f(\gamma_0, V_0)$.

Such a difference in final bank-angle happens because an approximation of the rotation time is included during the prediction. Being the aerocapture a short duration maneuver, the rotation time plays an important role in the entirety of the trajectory. In addition, an important difference occurs if the rotation happens when dynamic pressure is large or small. As previously mentioned, steep entry aerocaptures have much larger dynamic pressures, because they reach lower altitudes. If the switch happens when dynamic pressure is large, the difference between predicted and flown trajectory becomes much larger, if attitude kinematics are not approximated in the prediction. This is the reason why in the optimal aerocapture guidance by Lu et al. [2015] σ_d is smaller for steep entries than for shallow entries: the large dynamic pressure causes in fact larger offsets between the desired σ_d and the one actually obtained at the end of the transition between Phase 1 and Phase 2. In the *mod* concept, an approximation of the rotational dynamics is included in the prediction, and thus there is no need to find complex functions to estimate $\sigma_d = f(\gamma_0, V_0)$. σ_d can be constant for all the different possible scenarios in the same mission[18]. After some trial and error, it was found that $\sigma_d = 130°$ was the largest value for which good reliability was ensured in an unperturbed atmosphere, for a small subset of initial flight-path angles. To have some margins, the *mod* guidance was tested using $\sigma_d = 120°$, but also with $\sigma_d = 110°$, for comparison. The high-level algorithm of the *mod* concept is given in the box below.

Modified (*mod*) optimal aerocapture guidance.

At iteration k:

1. Phase 1:
   (a) propagate trajectory with bank-angle planned as in Figure 4.6. Here, σ_d is not interpolated, and is the same independently of the initial flight-path angle;
   (b) **if** final energy is negative, go to Phase 2;
   elseif $r_{apo} > r^\star_{apo}$, go to Phase 2;
   else, set $\sigma_{cmd} = \sigma_0$, and end iteration k.
2. Phase 2:
   (a) plan trajectory with constant σ;
   (b) solve[19] $r_{apo}(\sigma^\star) = r^\star_{apo}$;
   (c) set $\sigma_{cmd} = \sigma^\star$, and end iteration k.

4.5.2 LATERAL GUIDANCE (*modlat* CONCEPT)

The lateral guidance used in this modified version consists of planning a single bank reversal.

[18] This does not mean that σ_d cannot be a function of initial velocity or target apoapsis, but these are parameters that do not change for a given mission. There is instead no dependency (or only marginal) only on the initial flight-path angle.

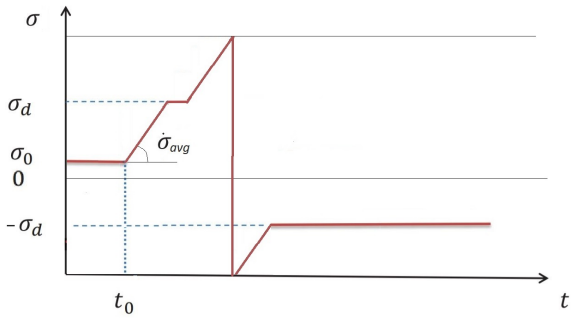


Figure 4.12: Bank-angle planning in the prediction of the *modlat* concept, during Phase 1.

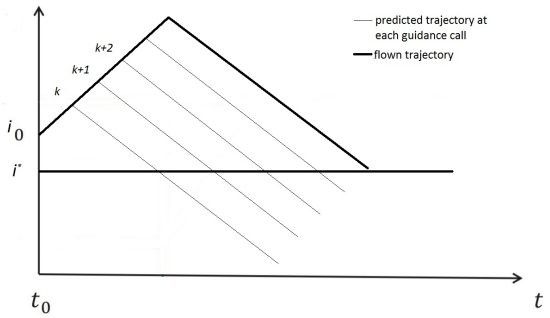


Figure 4.13: Logic of the reversal trigger.

This is done by integrating the motion in the prediction with a bank-angle history planned as shown[20] in Figure 4.12.

This way, during Phase 1, it is assumed that the bank-angle reversal occurs 3 s after the transition between Phase 1 and Phase 2 has occurred. During Phase 2, before inversion, σ_d in the figure is substituted by σ_{cmd}, and $r_{apo}\left(\sigma_{cmd}\right) = r^{\star}_{apo}$ is solved. After inversion, Phase 2 is the same as in the *mod* and *Lu* concept. The reversal is always assumed to go in the same direction, which is upwards. Also in this case, Figure 4.14 helps understanding the architecture of the guidance.

As shown in Figure 4.13, if the predicted final inclination error has different sign from the current one, nothing happens; but once the sign becomes the same, then the bank reversal begins.

Thanks to this lateral guidance, two benefits occur. First, only one reversal is done. Second, the prediction already keeps into account the future reversal, in which the Coriolis acceleration would be acting differently from how it would if the prediction was done without inverting the bank-angle.

As a final remark, the sine of the bank-angle is initially chosen such that the trajectory diverges from the target inclination. Although this might sound counterintuitive, and one might think that it is a choice that may decrease the robustness of the trajectory, this is an important factor, since it makes the reversal occur as soon as possible: consequently, the duration of the period before the reversal is reduced.

Instead, during Phase 1, the sine of σ_0 is chosen to minimize the initial offset. In addition, it changes if a maximum inclination error occurs. This is needed because, for very steep entries, the switching time to Phase 2 may be reached too late, once lateral control is not possible anymore.

A final, but very important feature, of this lateral guidance, is the fact that it is not triggered when the commanded bank-angles are showing large instabilities between one iteration and the next one. This is because, as an example, a predicted crashing trajectory gives final inclinations that are very far from the current and target ones. This is quite likely to happen with the modified Newton method, especially in highly perturbed environments.

4.6 NPC ARCHITECTURE

Figure 4.14 shows the block diagram for the NPC guidance. Without going too much into detail, the guidance has (ideal[21]) navigation data as input, together with two counters: the Phase counter, and the Inversion counter, which it receives from the previous iteration, and are initialized to 1 and 0, respectively.

At each iteration, as long as the Phase counter is still equal to 1, the guidance integrates the trajectory considering a bank-angle planned according to the respective guidance logic in used (Figure 4.15 shows the architecture for the *modlat* concept, but it can be generalized to the *Lu* and *mod*

[20] σ_0 and σ_d might as well both be positive, or have opposite sign.
[21] Navigation is assumed ideal, in the sense that no estimation errors occur.

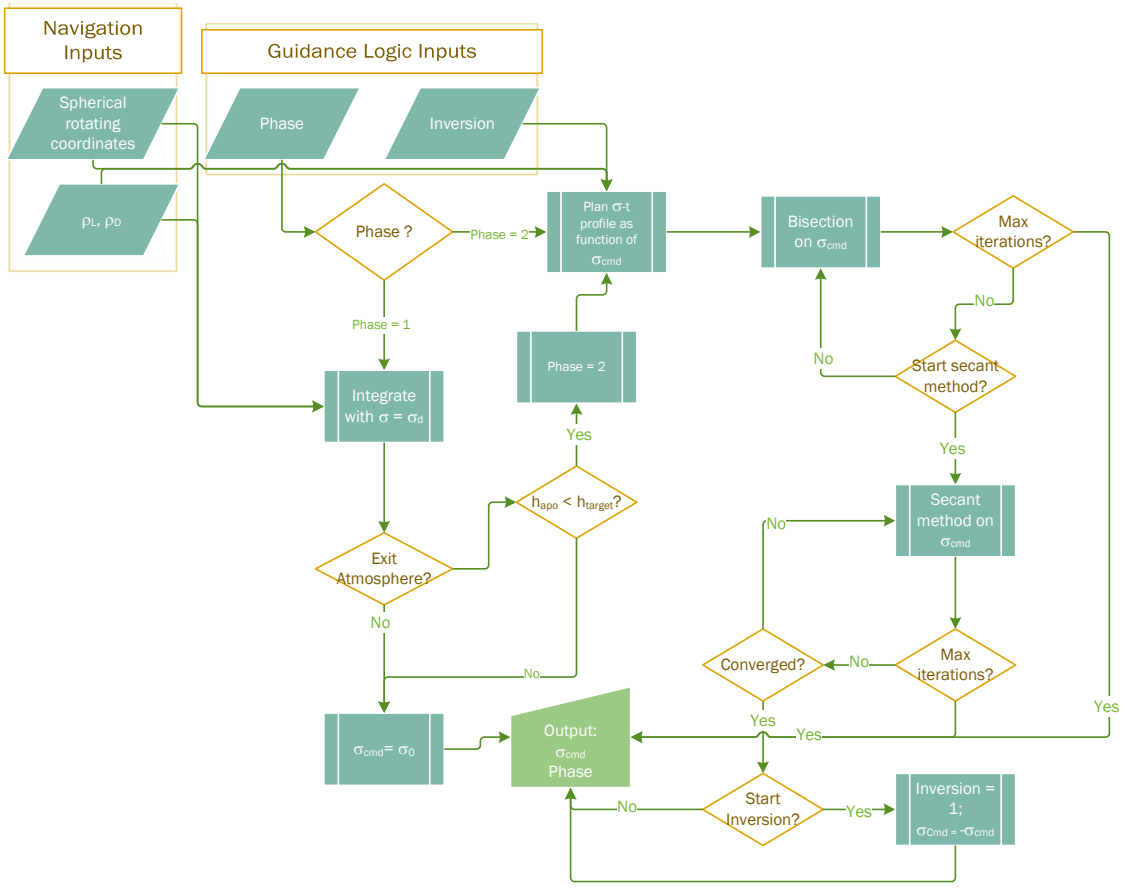


Figure 4.14: Flow-chart for the *modlat* NPC guidance, with bisection-secant method as a solver.

concepts, as long as the Inversion counter is set to 1 since the beginning).

Once the integration is done, it is checked twice. First, it is checked whether the integration went past the altitude $h_e x$. Afterwards, the altitude of the apoapsis is checked. If it is higher than the target, Phase 2 is triggered; otherwise, guidance stays in Phase 1, and the commanded bank-angle remains σ_0.

These checks are strictly related to the fist issue stated in Subsection 4.4.2.

During Phase 2, while Inversion= 0, plans again the bank-angle history, which is now a function of σ_{cmd}. The logic then solves the trajectory that leads to the target apoapsis for σ_{cmd}. The solver can either be the combined bisection-secant method or the modified Newton-Raphson method, both described in Subsection 4.4.2.

Once the solution is obtained, the guidance checks whether the conditions for reversal occur. If such conditions hold, the guidance begins commanding the reversal.Once Inversion= 1, the planned bank-angle profile is solved as a constant profile.

As a last remark, the choice for the equations of motion and propagator used is the consequence of a trade-off carried out in Section 6.3.1.

4.7 VERIFICATION

It is not necessary to verify each block of Figure 4.14, since most of those either will be verified in Chapter 6, or they can be easily verified by sanity checks. Therefore, the verification of the optimal aerocapture guidance consists of two parts only. First is testing the logic on an environment that is as much as possible the same as the one used by the predictor. This way, the trajectory should end up being extremely similar to the planned one. This way, it is verified whether the predictor works correctly. The second part consists instead of testing the original optimal aerocapture guidance in a perturbed environment, possibly very similar to that used in their work. By doing so, it is possible to verify whether the entire system works properly.

The next three figures (Figures 4.15, 4.16, and 4.17) all show the bank-angle for a flight in an environment in which the atmosphere is modeled according to the unperturbed[22] US76 atmosphere. All the trajectories begin with same initial conditions, and lead to a final apoapsis error of less than 100 m. During the entirety of Phase 2, the bank-angle does not change of more than 1°. This proves that the guidance is properly designed, at least for what concerns Phase 2. Moreover, it also proves what was said in Section 3.6, about how the changes in temperature, and therefore Mach number, and aerodynamic coefficients, are irrelevant in aerocapture (on Earth, at least). Aerodynamic coefficients are in fact modeled as constants in the guidance logic, and any displacement occurring in the bank-angle is due to this mismatch. Being such displacement extremely small, it is concluded that the coefficients do not change much. Additionally, these figures also prove that the use of an atmosphere whose perturbation is just the multiplication by a constant factor is very limiting since, because of the filtering, the perturbation is easily kept into account by the prediction. For what concerns Phase 1 (and the lateral guidance as well), a more specific discussion follows.

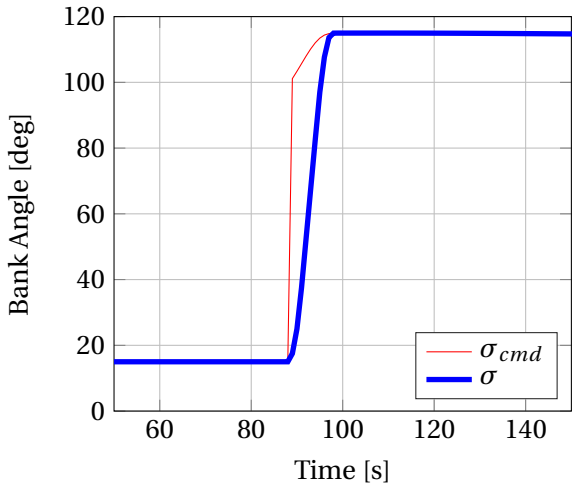


Figure 4.15: Verification of the *Lu* guidance. $\sigma_d = 100°$.

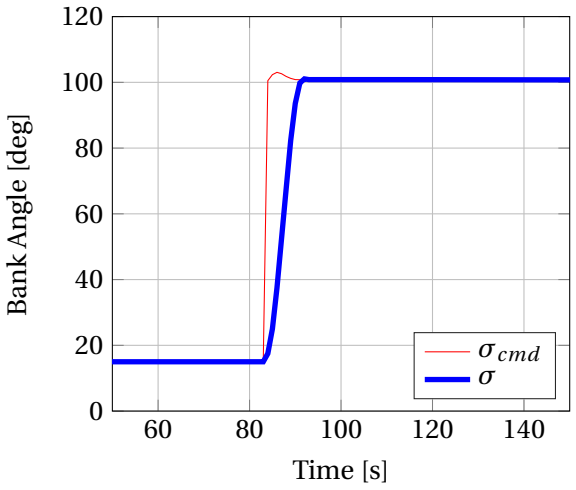


Figure 4.16: Verification of the *mod* guidance. $\sigma_d = 100°$.

Figure 4.15 shows the commanded bank-angle and the bank-angle history for a flight in which the planned σ_d is equal to 100°. The first commanded bank-angle is slightly larger, and equal to 101.1°. This is due to the fact that Phase 2 is triggered at the first call in which Phase 1 shows that apoapsis is reached, instead of the exact moment in which the shift should be, as it is instead in Lu et al. [2015]. However, the error due to such delay is much smaller than that due to the attitude dynamics. In fact, the commanded bank-angle increases up to 115° during the time in which the real bank-angle catches up.

Something different happens for the modified NPC, shown in Figure 4.16. In this case, the com-

[22] Precisely, the entire altitude-density profile has been multiplied by a constant. The guidance logic can account for such a perturbation perfectly, thanks to the filter implemented as in Equations 4.35.

manded bank-angle increases only for a short time after Phase 2 is triggered. That is due to the fact that the guidance expects the bank-angle to immediately rotate at $10.5\,°\,s^{-1}$, without considering any acceleration. However, the peak of 103° is reached after 2 seconds, and the commanded bank-angle slowly decreases while the bank-angle rate is saturated. Eventually, the actual bank-angle and the commanded one stabilize at 100.8°. This happens with a very small overshoot (of about 0.2°) in the actual angle, due to the fact that the commanded angle is still slightly decreasing while the actual bank-angle is already slowing down with maximum deceleration. In the end, the *mod* concept leads to a prediction in bank-angle that is 18 times more accurate than that of the *Lu* concept.

The situation is even more different in Figure 4.17, which shows the $modlat$ guidance. Initially, the same behavior of the commanded angle as in the previous case happens. However, in this case, the overshoot is much larger, of 5°; nevertheless, the bank-angle then decreases to 102°. At this point, a slow but steady increase in the bank-angle occurs. This is due to the fact that at each call, the guidance uses as a prediction a profile in which the reversal occurs immediately, with a constant bank-angle rate of $10.5\,°\,s^{-1}$. Therefore, it predicts a trajectory in which the Coriolis acceleration acts in the opposite direction than in reality, and in which there is a reversal that does not occur. This trend is relatively steep, and causes a change in bank-angle of about 15° in about 20 s. It stops once the reversal actually occurs. Such reversal lasts about 10 seconds, and causes a change in bank-angle of about 130°, which begins when the bank-angle is 115°. Once again, the bank-angle after the reversal is displaced by 3° only, proving that including the reversal in the prediction adds robustness to the guidance.

To see the effect of including the reversal in the prediction, Figure 4.18 shows another guidance concepts, which was eventually not used. That concept (labeled $modlat\infty$) can be defined as a limit case of the $modlat$, in which $\dot{\sigma}_{avg} = \infty$.

In such a concept the ever increasing trend before the reversal is much smaller than in the *modlat*, and causes an increase of 1° only. This means that the main cause of the increase in Figure 4.17 is not the wrong direction of the Coriolis forces, but the inclusion of a reversal that does not happen. The bank-angle after the reversal is displaced by another 10°; however, this time, the displacement is in the opposite direction, and almost entirely counterbalances the initial displacement. The final bank-angle is around 105°.

What can be concluded from this is that the inclusion of the reversalin the prediction (as in the *modlat*) has the twofold consequence of, on one hand, causing an increasing bank-angle in the first phase, and, on the other hand, having a good prediction once the reversal actually happens. The first one is a negative effect, while the second one is beneficial. However, the modified lateral guidance also has the advantage of planning a single reversal.

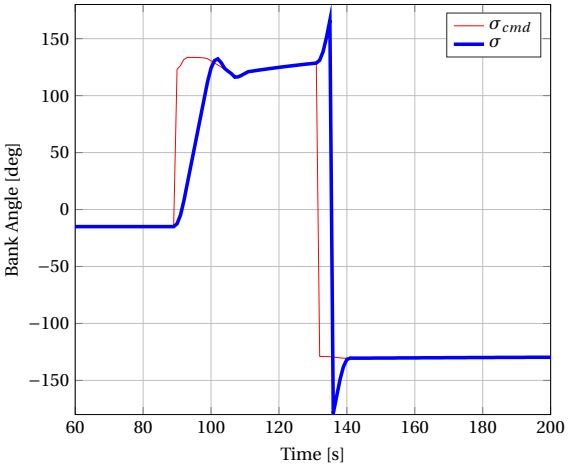


Figure 4.17: Verification for the *modlat* concept.

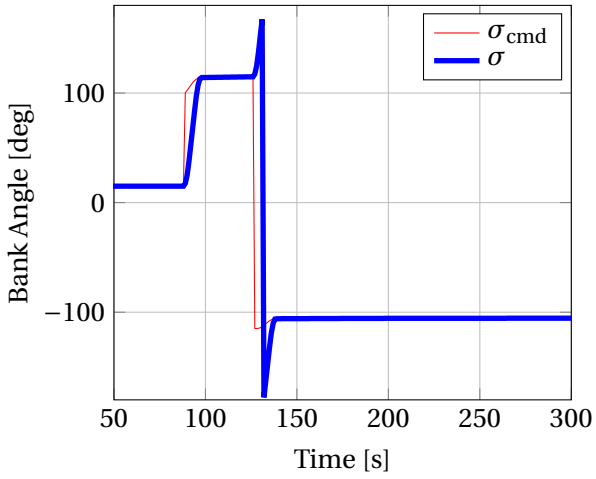


Figure 4.18: Verification for the $modlat\infty$ concept.

Table 4.1: Initial conditions and dispersions.

	γ_0	χ_0	V_0	h_0	δ_0	τ_0	C_D	C_L	Mass	Density (if *c*)	Speed of sound
Initial conditions	−5.725°	−1.6789°	11.055 m s^{-1}	121.9 km	−46.66992°	−242.75°	-	-	5.500 kg	-	-
Dispersions	±0.6475°	±5 m s^{-1}	-	-	-	±0.25°	±10 %	±10 %	±82.5 kg	±50 %	±15 %

Table 4.2: Summary of scenarios and guidance logics tested.

	GRAM99 *ls*	GRAM99 *l*	GRAM99 *c*	US76 *ls*	US76 *l*	US76 *c*
modlat σ_d =110°	**X**	**X**	**X**	**X**	**X**	**X**
modlat σ_d =120°	**X**	**X**	**X**	**X**	**X**	**X**
mod σ_d =110°	**X**	**X**	**X**			**X**
mod σ_d =120°	**X**	**X**	**X**			**X**
Lu	**X**	**X**	**X**			**X**
simple	**X**					**X**
predload	**X**					**X**

4.8 RESULTS

Three NPC guidance logics have been tested on different sets of 1000 Monte Carlo runs which included various kinds of perturbations. These are the original NPC without the lateral guidance (labeled *Lu*), the modified NPC without the lateral guidance (labeled *mod*), and the modified NPC with the lateral guidance (labeled *modlat*). The first two use the bisection-secant solver; the latter uses the modified Newton-Raphson solver. There is no particular reason for this choice, except for the fact that there was no time to adapt the modified NPC with lateral guidance to the bisection-secant solver. The initial conditions and dispersions can be seen in Table 4.1, at the end of the chapter. The state variables are in the rotating reference frame. The dispersions are the same for all the Monte Carlo runs (except for the density, which differs if large-scale or small-scale perturbations are triggered).

The design of the entire simulation setup, together with its verification and validation, is described in Chapter 6.

Six different scenarios have been simulated, three for each atmosphere model: the perturbations consist of either constant atmospheric perturbations (labeled *c*), large-scale perturbations (labeled *l*), and large-scale and small-scale perturbations at the same time(labeled *ls*). Lastly, for reasons that were stated in Appendix A, the predictive load-relief was not used in the previously mentioned NPCs.

As comparison, two additional guidance systems were tested: these are a guidance without Phase 1 (labeled *simple*), and a guidance without Phase 1, but with dynamic pressure predictive load relief active (labeled *predload*).

Table 4.2 shows all the different Monte Carlo scenarios and guidance combinations that have been tested. The *modlat* has been tested in all possible scenarios, and in two different versions. The *mod* and the *Lu* have been tested for all the GRAM-99 atmospheres, and the US76 *c*, which is interesting from a concept point of view. Eventually, the *simple* and the *predload* have been tested only in the GRAM-99 *ls* (which is, in theory, the most difficult one), and the US79 *c*. This totals to 28 Monte Carlo runs for the NPCs, which will be discussed in the upcoming sections. Due to the large amount of data, only the most interesting results will be discussed.

All tables summarizing the results can be consulted at the end of this chapter, together with the initial conditions and ranges of dispersions.

This section is divided into a few subsections. Subsection 4.8.1 is aimed to find and eliminate

from the discussions those cases in which the environment conditions are too adverse. Afterwards, in-plane performance and robustness is evaluated for the *Lu* and the *mod* guidance systems in Subsection 4.8.2. Out-of-plane performance is discussed in Subsection 4.8.3, for the *modlat* logic. Subsection 4.8.4 discussed instead how the *mod* guidance respects the constraints, compared to the *simple* and the *predload* guidance. In Subsection 4.8.5, the total heat load along the trajectory is treated. Along all the subsections, a special regard will be given to how the atmospheric model affects the performance. Eventually, results are summarized in Subsection 4.8.7.

4.8.1 OUTLIERS

In very Monte Carlo run there have always been a few outliers, for which the initial conditions or perturbations made it impossible for the guidance to reach the desired apoapsis. These cases should be considered separately from the normal cases in which the guidance was not successful, simply because the reason of the unachieved success is not due to the guidance logic.

In the cases considered, the outliers were always trajectories that were flown full-lift up for the entirety of the flight, and still went lower than the target apoapsis. The opposite case, in which a trajectory was flown full-lift down, and still went higher than the target, did not occur. From this definition of outliers, it can be concluded that the guidance logic could not be the cause.

Outliers have been included only in the results for the *modlat* guidance, in Table 4.3, to be compared with the statistics of Table 4.4, and see how they affect the data. In the rest of the results, outliers will not be considered. There were 10 outliers in any Monte Carlo run with no large-scale nor small-scale perturbations, and 9 in the others.

The pattern found among the outliers for the constantly perturbed atmosphere is the following: they all had large negative initial flight-path angles (smaller 12 %), dense atmosphere (densest 50 %), large drag coefficients (largest 50 %), and small lift coefficients (smallest 30 %). Multiplying all these percentages together, one has about a 1 % chance of having all these situations occurring at the same time, which is the frequency with which these outliers have occurred. The trends can be seen in Figure 4.19. Each horizontal line refers to a single trajectory; 0.5 is the mean of the distribution of each parameter. For the case of atmosphere with large-scale perturbations the percentages change in some cases. The correlation with γ_0 stays the same, the one with the drag coefficient increases (largest 30 %), and the one with the lift coefficient decreases (smallest 20 %). Interestingly, of the 10 outliers occurred for the constantly perturbed atmospheres, 6 happen with the same exact conditions (except for the atmosphere) as 6 of the 9 occurring with constantly perturbed atmospheres, giving additional confirmation of the 50 % correlation with the density. Moreover, the cases that are outliers for the large-scale perturbations, but are not for the constant perturbation, all have a very thin atmosphere in the constant perturbations case. The outliers in the fully perturbed cases occur for the exact same conditions of the cases perturbed with only large-scale perturbations.

4.8.2 LONGITUDINAL GUIDANCE

A $30\,\mathrm{m\,s^{-1}}$ ΔV causes approximately an initial mass about 1 % larger. For this reason the ΔV is the main parameter according to which performance is evaluated. ΔV includes also out-of-plane corrections, but these will be discussed in the next subsection. In this subsection, the *Lu* guidance is compared to the *mod* guidance, both in terms of ΔV, accuracy, and robustness. Comparisons with third guidance systems will be skipped, since that is already done extensively by Lu et al. [2015], and the *mod* guidance is very similar to it.

In terms of ΔV, one can see from Tables 4.5 and 4.6 that the *mod* guidance achieves, on average, $10\,\mathrm{m\,s^{-1}}$ less, and is thus more performing. However, as previously mentioned, the *Lu* guidance is tuned for the Orion MPCV, and is thus not optimal for this case.

On what concerns the accuracy, this is less performing in the *mod* concept; also, the worst cases worsen much between the *mod* with $\sigma_d = 110°$ and $\sigma_d = 120°$. In fact, with the latter, some (even

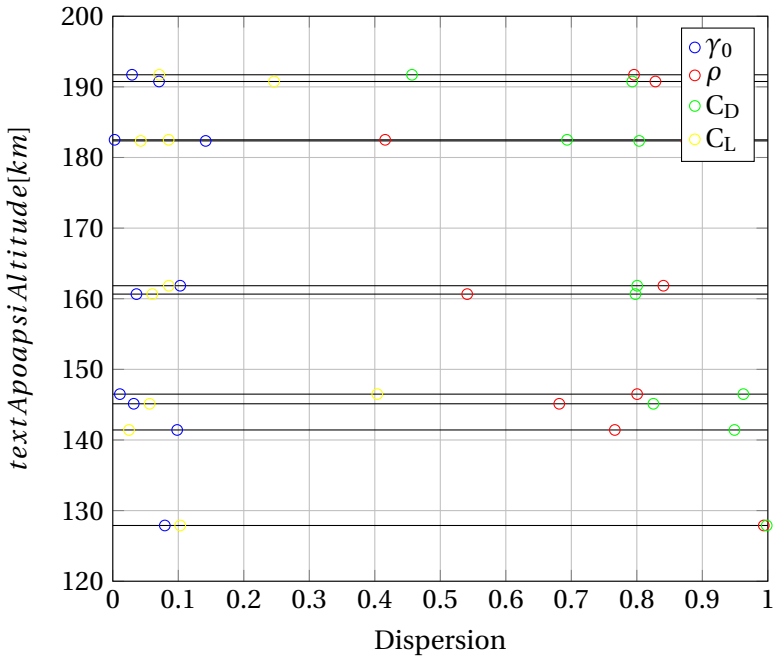


Figure 4.19: Initial conditions for the outliers in the US76 *c* simulation .

though very rare) large misses occur, up to some hundreds of kilometers. It is therefore not adviced to attempt using even larger values for σ_d. Nevertheless, this is a case in which the accuracy is not a good evaluator of the performance, since the less accurate *mod* achieves smaller ΔVs than the more accurate *Lu*. Again, these differences are probably due to tuning.

It can then be concluded that the new concept is very successful, in that it is about as performing as the original one (a more thorough study and tuning would be required to give a definite result on this, but it is likely that both methods, properly tuned, will give very similar performances), but it has a more robust concept, as demonstrated in Section 4.7, and requires much less tuning.

Figures 4.20 and 4.21 show the performance of the *Lu* guidance, as in Lu et al. [2015]. However, a couple of remarks are needed: first, it is unclear what perturbation model has been used by them (it is known, however, that the atmosphere used is the GRAM-2010); second, it is also not clear at what altitude the guidance is shut down. It is in fact very unlikely that such high accuracy can be achieved, if at least large-scale perturbations are modeled, by shutting down the guidance before the 120 km altitude. This is because atmospheric perturbations of that kind can cause, between the 100 km and the 120 km lines, errors in the order of magnitude of 1 km. For reasons explained in Section 3.7, all guidance logics in this chapter have been shut down at 100 km altitude. The last unclear element is the fact that in the original work by Lu et al. [2015] the J_2 effect is not included in the propagation of the exit leg. As stated in Subsection 4.4.1, not including this leads to errors up to 8 km in apoapsis altitude. It is therefore believed that in their work Keplerian motion after atmospheric exit was assumed also for the outer loop. Alternatively, it might be possible that they propagate the motion (both in the inner loop and in the outer loop) using a Keplerian orbit, but using orbital energy computed as in Equation (3.28), which accounts for J_2. Nevertheless, this would still not account for the effect of J_2 on the eccentricity.

Figures 4.22 and 4.23 show the same performance parameter as the previous two figures, but for the *simple* and the *mod* guidance concepts that were simulated in this research. In this case, the environment used is the GRAM-99 *ls*. It is seen that the the accuracy is much worse than the cases by Lu (and many data points lie outside of the range in the figure); nonetheless, the ΔV curve of the

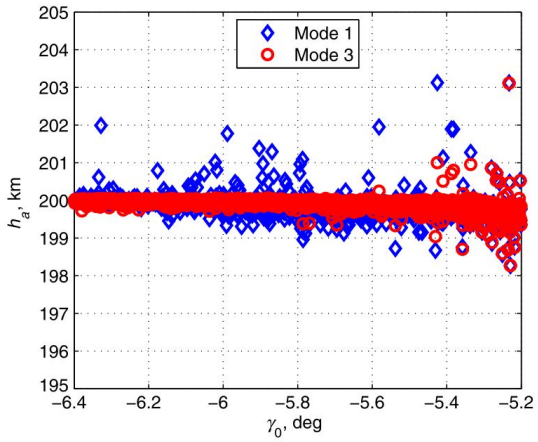


Figure 4.20: Apoapsis altitude as a function of initial flight-paht angle, from Lu et al. [2015]. The guidance reproduced in this research corresponds to *Mode 1*.

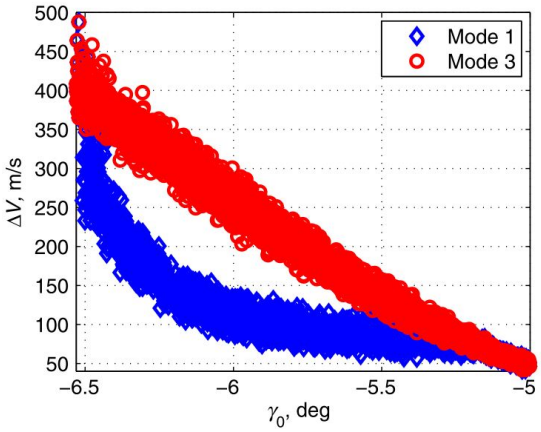


Figure 4.21: In-plane ΔV as a function of initial flight-paht angle, from Lu et al. [2015]. The guidance reproduced in this research corresponds to *Mode 1*.

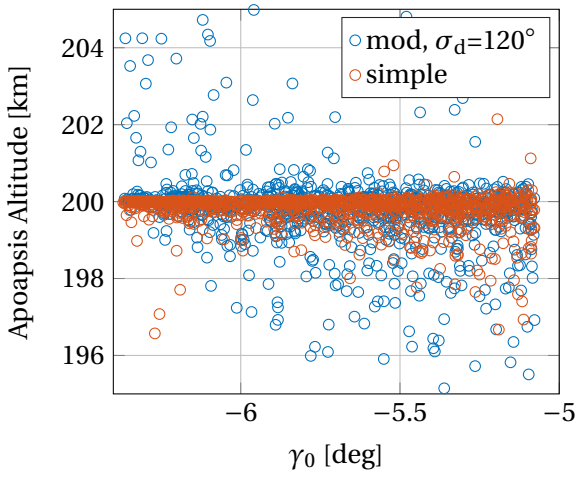


Figure 4.22: Apoapsis altitude for two guidance concepts in the GRAM-99 *ls* environment.

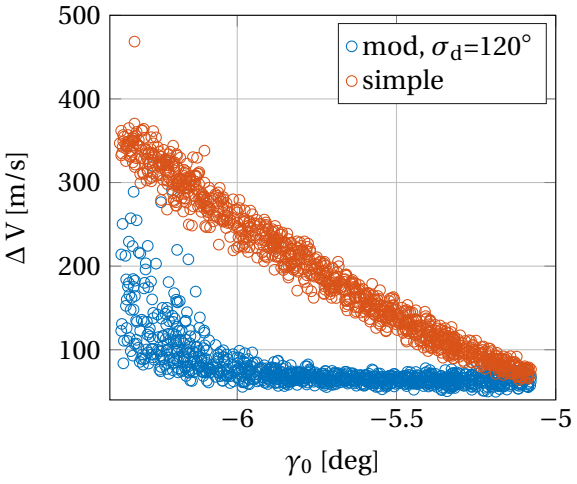


Figure 4.23: ΔV for two guidance concepts in the GRAM-99 *ls* environment.

simple is very similar to that of Lu's Mode3, even though it has an averagely smaller ΔV, whereas the curve of the *mod* is rather similar to that of Lu's Mode 1, but is much flatter. Both differences are probably caused by the fact that in the vehicle used in this research has a lift-to-drag ratio that is larger than that used by Lu. Especially in the case of the *mod* guidance, one can find a light correlation between larger lift-to-drag ratio and smaller ΔV.

Figure 4.24 shows the effect of different perturbation models on the apoapsis accuracy for the *Lu* concept. Both the *ls* and the *l* environments cause many trajectories to have errors larger than 5 km, and up to 100 km. It is clear that, despite having smaller perturbations in magnitude, the *ls* and *l* models cause more difficulties than the *c* perturbations. Their effect is asymmetric though, and the errors when the apoapsis is low are much smaller. This happens because, as explained in Section 2.5.2, lift-up trajectories are more stable than their lift-down counterpart. In this case, this is a fortunate behavior, since these perturbations never cause any crashes.

In addition, small-scale and large-scale perturbation models do not cause any sensible differences in ΔV with respect to constant perturbations model.

In conclusion, recalling the problem statement of Section 2.7, there is a very wide range of entry angles for which the final ΔV is, in the worst case scenario. A guidance such as the *mod*, with

σ_d = 120°, in an environment such as the GRAM99 *ls*, satisfies the ΔV requirement with a ΔV_{max} of 80 m s^{-1} for entry angles between -5.1° and -5.8°. The range of flight-path angles is even larger than required, and in the tested set the success rate is 100 %. Hence, the *mod* guidance satisfies the requirement concerning the ΔV, and the vehicle can be designed with propellant enough for a ΔV of 80 m s^{-1}. This is valid if no specific target inclination is desired.

The remaining subsections will define the limit values for the other requirements. But first, the same requirement on ΔV is defined, for the case in which a certain target inclination is desired.

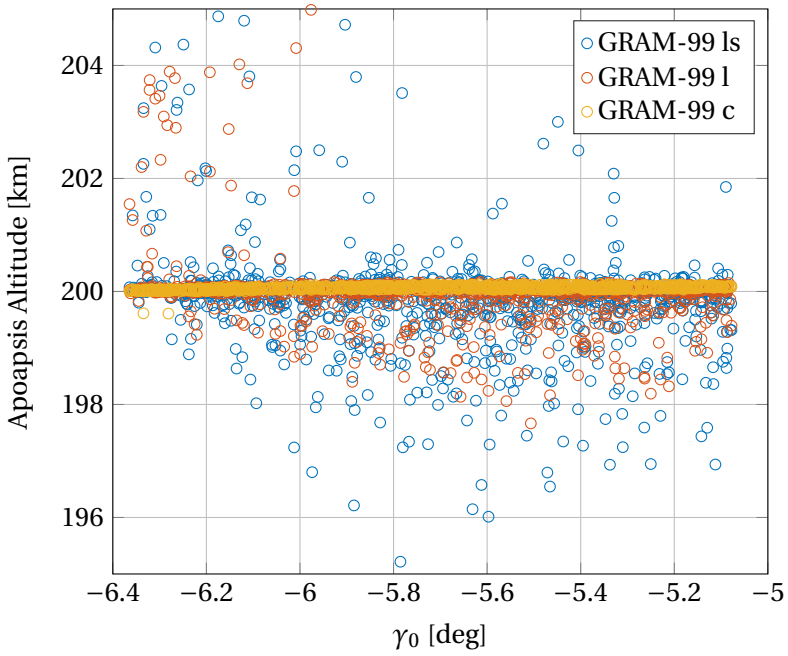


Figure 4.24: Apoapsis altitude obtained with the *Lu* guidance for in different atmospheres..

4.8.3 OUT-OF-PLANE ΔV

This subsection only concerns the *modlat* concept. The accuracy of the lateral logic is usually judged in terms of precision in the final inclination. However, a different approach will be taken here. In fact, also in this case, one does not care how much difference there is in inclination when the atmospheric part of aerocapture is completed, but rather how much propellant should be spent to correct the inclination. This amount can be very different from case to case, even for same magnitude of inclination error. The same error in inclination may cause an almost negligible increase in propellant consumption if the first burn is large, whereas it might cause a much larger increase instead (in absolute value, and even larger in relative terms), if the first burn is small. From Table 4.4 it can be seen that the average increase in ΔV due to the out-of plane component is approximately 6 m s^{-1} when also small-scale perturbations are simulated, and around 2 m s^{-1} otherwise.

Despite being quite small, it is much larger than other lateral guidance concepts. However, an average lateral guidance is also more invasive, in the sense that it usually affects the in-plane performance more than this one, because of the theoretically unlimited number of bank reversals allowed. As an example, for the US76 *c* atmosphere, and with σ_d = 110°, the average in-plane ΔV of the *modlat* is only 0.5 m s^{-1} more than the average of the *mod*. It should be studied how much other lateral guidance logics affect the in-plane performance. Thus, before refusing this concept because of the averagely large component of out-of-plane ΔV it causes, it should be compared to other guidance logics in terms of total ΔV. Once again, here comes the importance of considering the in and out-

of-plane ΔV instead of simply the error in the final inclination. This comparison was not done in this research.

An additional remark comes from the average difference between the guidance with σ_d =110° and the one with σ_d =120°. The latter has usually a slightly better performance in the in-plane component; however, such margin disappears when considering the full ΔV. In fact, a large σ_d decreases the lateral controllability for two reasons: first, it increases the time in which $\sigma = \sigma_0$; second, the average bank-angle in Phase 2 has consequently, on average, a smaller sine. Figure 4.25 shows the relative component of out-of-plane ΔV for the *modlat* in atmosphere US76 *c*, and Figure 4.26 shows the same parameter fot the guidance tested in the atmosphere GRAM99 *ls*. In the first case, a correlation between either extremely steep or extremely shallow flight-path angles is evident, which further proves the previous reasons. The second case shows instead a clear problem and limitation of the concept of this lateral guidance. That is relying too much in having an atmospheric profile after the reversal that does not differ much from the one before the reversal. In the case with σ_d = 110°, about 10 % of the cases have an out-of-plane ΔV component that is 25 % or larger than the total. Also, correlations with the initial flight-path angle disappears.

Therefore, a correction in the lateral concept is advised. This consists of using the concept of single reversal twice. This might sound like a contradiction, but it is not if the first reversal is anticipated. This modification would make the guidance logic more similar to the concept developed by Shen and Lu [2004]. The amount of time by which the first reversal should be anticipated would be a parameter to be tuned, or a function of either state or command to be tuned.

As a last reamark, it can be said that the use of the modified Newton-Raphson method (which was used in the *modlat* only) does not cause any relevant drawbacks to the average in-plane performances. in fact, the in-plane performances are very similar to those of the *mod* concept, in which the bisection-secant method had been used.

It is unknown, though, whether the method causes any problems to the lateral guidance instead. In any case, the modified Newton-Raphson method has been obtained with empirical trial and errors, and its concept is much less robust than that of the bisection-secant method, or Brent's method. Therefore, a good amount of skepticism is always adviced before making use of it.

Concerning the requirements stated in Section 2.7, the answer depends on which environment in use. The fully perturbed environment is probably more realistic. It is, however, only a model, and other models for perturbations exist. At this point, there is no reason to believe that different perturbation models may lead to better or worse performance. Hence, it is decided that such a lateral guidance requires more investigation, in differently perturbed environment, before a ΔV_{max} that includes out-of-plane components can be properly set.

4.8.4 LOAD FACTOR AND HEAT-FLUX PEAKS

The goal of this subsection is that of proving that, no matter what, the concept of the optimal aerocapture by Lu guidance minimizes any sort of trajectory peak. Using a predictive load relief in Phase 2 only, would be equivalent to raising the σ_d, and therefore, making the guidance less robust. Of course, it might be desirable to have a large error in ΔV rather than large load factors for example. However, such kind of decision is strictly dependent on the mission, and a trade-off should be done at more advanced stages of the design. It will also be shown that simply using a predictive load-relief model (the *predload* guidance) always causes larger peaks than the optimal aerocapture guidance.

To prove the first statement, it is sufficient to see when the trajectory peaks occur during with a *mod* guidance. If the peaks all occur before the switch to Phase 2, there is no use in including the predictive load relief in Phase 1. Also, if that is the case, they cannot minimized further, since a full-lift up command was given until that moment. Figures 4.27 and 4.28 show, respectively, the dynamic pressure and heat rate peaks, as a function of the difference between when those occur, and when Phase 2 occurs. It is clear that, in both cases, for the trajectories with highest peaks,

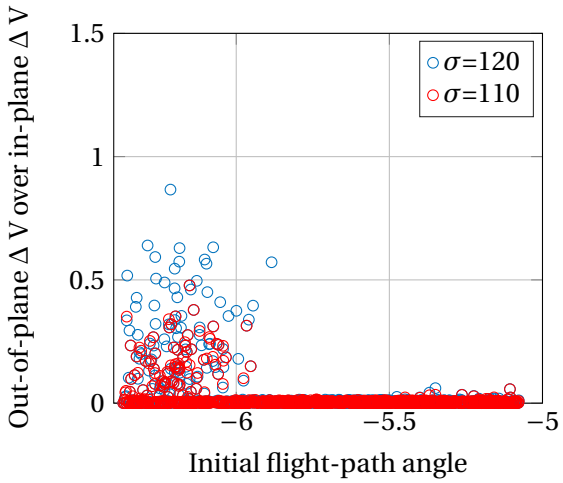


Figure 4.25: Lift-up lift-down bang-bang trajectories, in the US76 *c*.

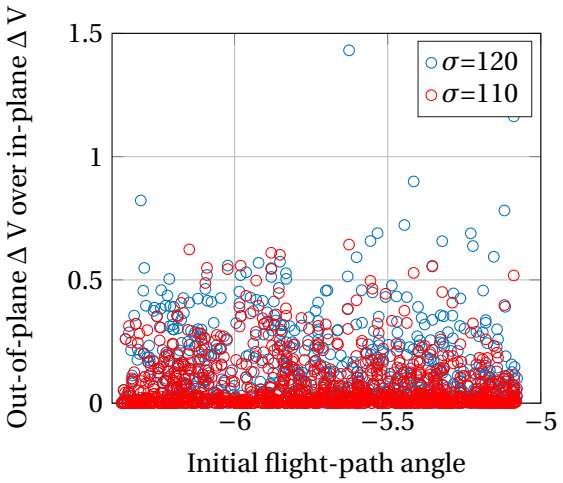


Figure 4.26: Lift up-lift down bang-bang trajectories, in the GRAM99 *ls*.

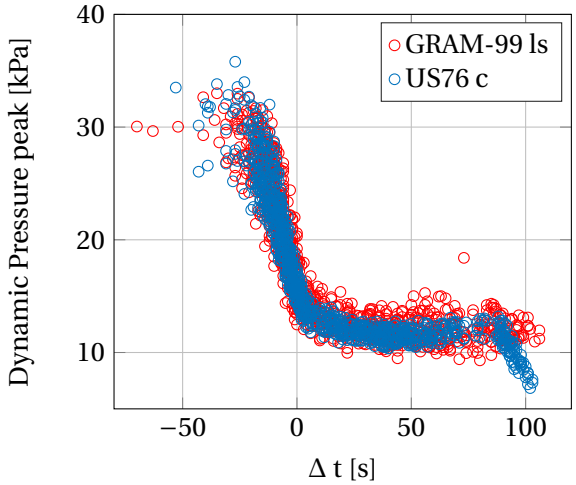


Figure 4.27: Dynamic pressure peaks with *Lu* guidance as a function of $\Delta t = t_{\max q} - t_s$.

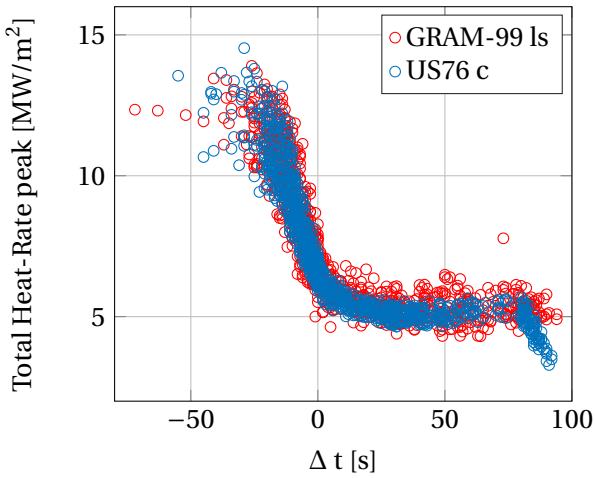


Figure 4.28: Total heat rate (radiative and convective) peaks with *Lu* guidance as a function of $\Delta t = t_{\max \dot{q}} - t_s$.

the peaks occur in Phase 1. This is valid for the *Lu*, the *mod* and the *modlat*, in all atmospheres. However, of course, the trend is more neat when a *c* perturbations atmosphere is used. However, it is necessary to specify that peaks due to sudden perturbations cannot be avoided by the predictive load-relief anyway. Attempting to minimize the peaks that occur in Phase 2 would therefore be equivalent to increasing the value of σ_d, which would make the whole system less robust. Also, it would not be necessary, since both the dynamic pressure and the total heat rate are not large in Phase 2.

In addition, when looking at the figures, one should also consider the fact that for about the first 2 to 3 seconds after the peak, the bank-angle has not changed much: thus, also the peaks occurring after cannot really be minimized in any other way.

The *predload* guidance has been set to respect constraints on the load factor only. The settings include $k_0 = 10$, $\delta = 8$ s, and $a_{\max}$ =2.5 g. The very low k_0 has been chosen together with a very low constraint because, otherwise, the predictor would have had generated trajectories with exceedingly large cosines of the bank-angle. This is because the adaptive step-size integrators used in this research turned out to be incompatible with constraining the cosine of the bank-angle in the pre-

dictive load-relief, as it is shown in Appendix A. This is a limitation to the current research only, and not of the method itself. It is shown in Figure 4.29 that, no matter what, the optimal aerocapture guidance always leads to lower dynamic pressure peaks than the *predload*. And this is without even considering the very large inaccuracies that the *predoad* leads to (see Table 4.7). The predictive load-relief does work properly, since it causes dynamic pressure peaks that are, in most cases, lower than those of the *simple* guidance: however, it can be said that a finer tuning could lead to better results.

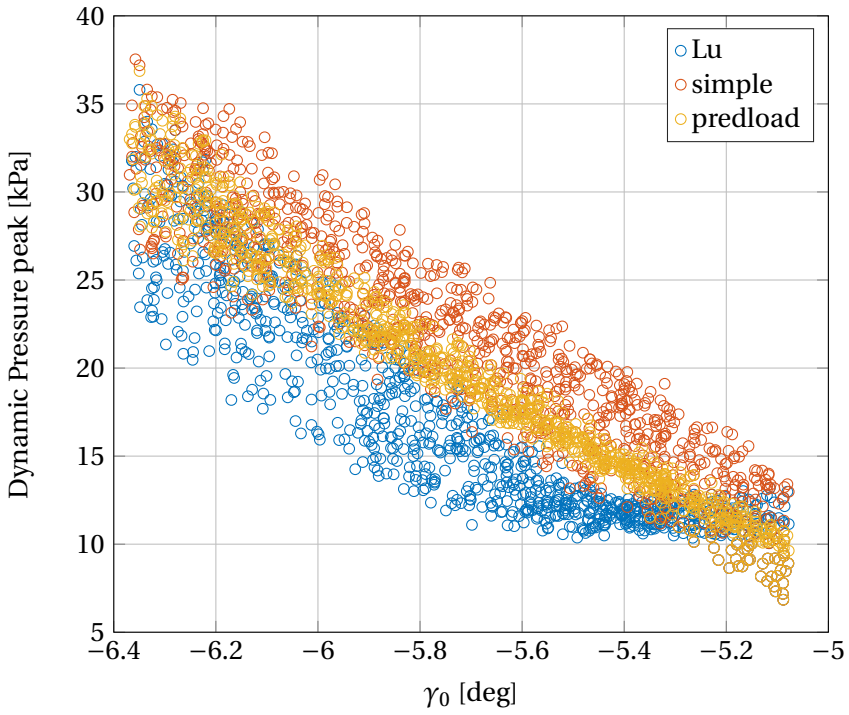


Figure 4.29: Dynamic pressure peaks as function of entry angle for different guidance concepts. Atmosphere used is the US76 *c*.

Figures 4.30 and 4.31 show a comparison of the bank-angles and the trajectories with the three different concepts. The choice of the trajectory is unfortunate, since it is one of the cases in which the *simple* guidance outperforms the *predload* in terms of peak. However, the figures are useful in understanding the concepts on which the guidance systems rely. In the predload case, for example, it is possible to see the moment in which the guidance shifts to full lift-up. It is also evident that the guidance takes that into account before hand, by setting an initial bank-angle larger than that of the *simple*. This is what causes the *predload* to be outperformed by the *simple*. Setting a larger δ in the *predload* would anticipate the shift to full-llift up. However, it might as well cause an overreaction that might lead to a skipout. Eventually, Figure 4.31 clearly shows the reason why the *Lu* concept is superior in terms of load relief: in fact, by setting the initial bank-angle to full lift-up since the beginning, for the same velocity, the trajectory is always flown at a equal or higher altitude, until Phase 2 is triggered. For this reason, even a perfectly tuned *predload* could not outperform the *Lu*.

In conclusion, one should refer again to the mission requirements of Section 2.7. It has been shown that in no case $\dot{q}_{max} = 18\,\mathrm{MW\,m^{-2}}$ is exceeded. It is reminded that there is no intent of looking for the entry conditions in which such value is minimized, since such limit is given by the PICA, which is the ablator material.

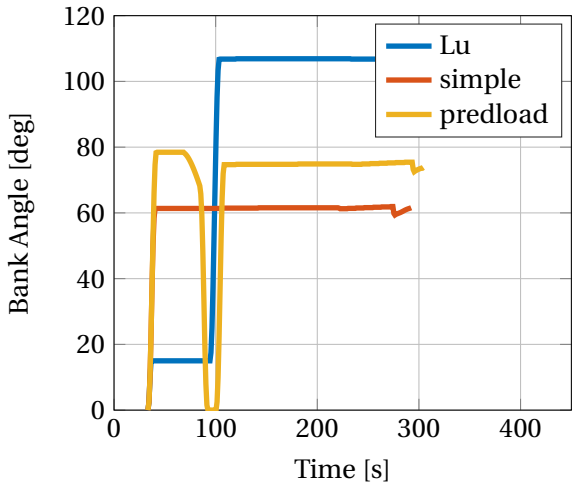


Figure 4.30: Commanded bank-angle history for a single trajectory for different guidance concepts. Atmosphere used is the US76 *c*.

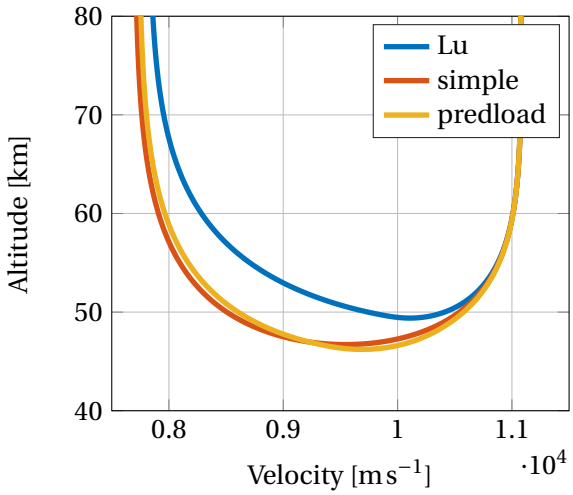


Figure 4.31: Altitude-velocity profiles for a single trajectory for different guidance concepts. Atmosphere used is the US76 *c*.

The load factor can be estimated from the dynamic pressure. The relation is the following:

$$n = \bar{q}\frac{1+\left(\frac{L}{D}\right)^2}{BCg_0}. \tag{4.41}$$

Being the nominal value of the ballistic coefficient at $M \geq 29.5$ equal to 356 kg m^{-1}, it is increased to 466 kg m^{-1} to account for Mach numbers as low as 10 (which are never reached anyways), and perturbations in drag coefficient of 10% and in mass of 3 %. With a conservative lift-to-drag ratio of 0.34:

$$n \approx \frac{\bar{q}}{3.8\,\text{kPa}}. \tag{4.42}$$

The maximum load factor occurred with the *Lu* concept is thus smaller than 10. The values are similar with the *mod* and *modlat* concepts. In addition, there is a range of initial flight-path angles in which the load factor is limited to less than 5. The range goes from-5.6° to -5.1°. Hence, with such a concept, a requirement for manned aerocapture is met. This is not the case with the *simple* concept instead.

This concludes the discussion about load factor and heat rate peaks. Hence, it has been shown that these are inherently minimized by the *Lu* (and, by analogy, by the *mod* and *modlat* concepts as well). The only way to reduce these peaks even further (provided that it is even needed) is to delay the time at which Phase 2 is triggered, which would make the guidanc less robust. Whether this is or not a good option should depend on the mission.

4.8.5 TOTAL HEAT-LOAD

In Subsection it was proved that the integral of any function of density and velocity[23] is, for an aerocapture, minimized by a bang-bang trajectory. The order of the commands would depend on the function itself. In aerocapture, radiative heat is usually dominant; however, in this case, the initial speed is relatively low, and thus radiative and convective heats are in the same order of magnitude. In this subsection, it will be attempted to understand which trajectory minimizes the radiative heat, and how that affects the total heat load. The study will be done for the *mod* with σ_d =120°: in fact, it is important for this analysis that the bank-angle after the shift is always pretty much the same.

[23] Except if such function is $f(\rho, V) \propto \rho V^3$.

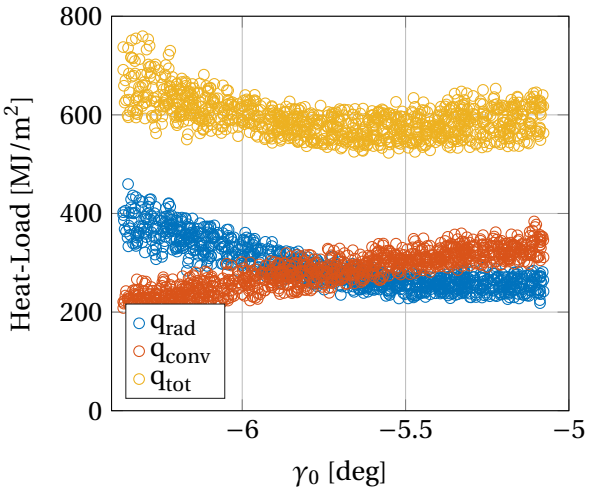


Figure 4.32: Different heat loads for the *mod* concept with $\sigma_d = 120°$. Atmosphere used is the US76 *c*.

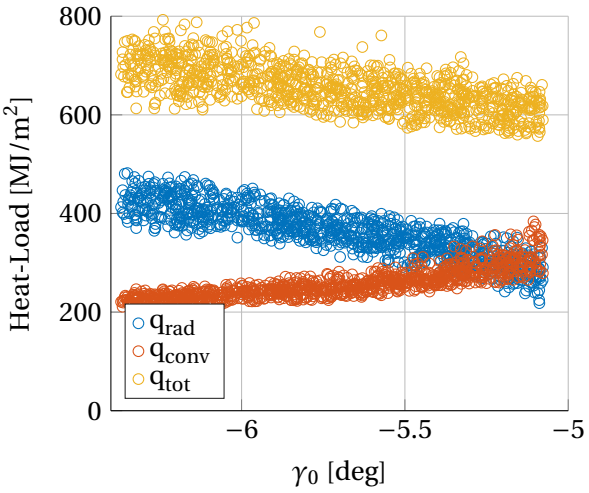


Figure 4.33: Different heat loads for the *simple* concept. Atmosphere used is the US76 *c*.

The analysis is done for the atmosphere US76 *c*. It is then checked whether the conclusions of this analysis still hold in a more realistic evironment, by comparing the results obtained with the more realistic atmosphere GRAM99 *ls*.

From Figure 4.32 one can easily see that, as proved by Sigal and Guelman [2001], the shallower the initial flight-path angle, the larger the convective heat load. However, the radiative heat load follows an opposite trend. Hence, the trajectory minimizing the total radiative heat load is the same minimizing the total ΔV. On what concerns the total heat, meaning the sum of convective and radiative heat loads, the situation becomes more complex. In fact both previously mentioned trends are nonlinear, and their nonlinearity differs from one another. For most of the possible initial flight-path angle ranges the trend of the radiative heat load is dominating. However, at some point, the opposite occurs. Thus, there is a small region in which the total heat load is minimized. Such a region coincides to where the ΔV becomes almost insensitive to the initial flight-path angle.

When computing the heat load in the GRAM-99 *ls* atmosphere the results are very similar (there is not even a noticeable change in variance), and therefore an equivalent plot is not reported. It is however interesting to see how the *simple* guidance performs. Again there is a decreasing trend for shallower initial flight-path angles. Also, for a same initial flight-path angle, the convective heat load is lower for the *simple* (as predicted by Sigal and Guelman [2001]), whereas the radiative and total heat loads are both larger (on average) for the *simple* concept. This is an additional indication of the fact that the trajectory minimizing the radiative heat is also bang-bang.

Since radiative heat load dominates at very high speed entries, it is expected that the equivalence between minimum heat load and minimum ΔV aerocapture becomes more clear for higher entry velocities. This statement is partly supported by the results obtained by Robinson et al. [2009], who show that, for higher entry velocities, the full lift-down trajectory causes lower total heat loads than the full lift-up trajectory. However, there is no clear increasing trend in their results. Also, it should be stated that the empirical formulas here used become less reliable at high speeds (those for convective heat are quite unreliable at 12,000 $\mathrm{m\,s^{-1}}$, whereas the Tauber-Sutton relation for radiative heat is quite reliable up until 16,000 $\mathrm{m\,s^{-1}}$), and therefore it is hard to tell what happens in those cases.

In conclusion, considering again the requirements from Section 2.7, also in this case there is a large range of entry angles that minimizes the total heat-load. Such a range is rather large, and goes from -6° to -5.1°. For these conditions, a maximum total heat load of 650 $\mathrm{MJ\,m^{-2}}$ is ensured to not be exceeded.

4.8.6 BANK-ANGLE HISTORY AND PERTURBATIONS

It has been seen that the large-scale and small-scale perturbations do not affect the average performances. However, every single one of the tables at the end of the chapter shows how they increase the standard deviation of the accuracy, as well as the average error in apoapsis. It has been repeated many times how the accuracy is not a very good indicator of the performance. Nevertheless, the accuracy is still an indicator of how robust the concept is.

The reason of this is the fact that every single concept in this chapter is designed with, as target, a certain apoapsis altitude. Every solver, in Phase 2, aims to find the bank-angle that leads to a target apoapsis. Hence, if the guidance leads to a well performing trajectory in terms of ΔV, but misses the apoapsis largely, it cannot be considered reliable. It still performs well, but it does not perform the way it should. Of course, as long as the misses are of only a few kilometers, this does not lead to any problems, but issues begin when misses are of the order of magnitude of the apoapsis altitude.

It is clear that perturbations affect the robustness of the guidance negatively, even though they might end up, in some cases, improving the performance (in many cases, the trajectory with minimum ΔV is simulated in a *l* or *ls* atmosphere). Hence, it is interesting to see why such a decrease in robustness happens. The best way to do this is looking at the bank-angle profile: a very fast varying bank-angle implies that the perturbations affect the trajectory much. In addition, a bank-angle brought to saturation is also an indicator of lack of robustness. Nevertheless, if that happens towards the end of the trajectory, it is not really a problem. Figures 4.34 and 4.35 show 5 trajectories flown with the exact same perturbations, in the US76 *c* and the GRAM-99 *c* atmospheres. In Figures 4.36 and 4.37 all perturbations are the same again, except for the atmospheres, which are, respectively, the GRAM-99 *l* and the GRAM-99 *ls*. The two cases share the large-scale component of the perturbation.

The trajectories in Figure 4.34 all behave like shown in the verification. They differ from each other only (except for duration and Phase 2 trigger time) for the constant value of the bank-angle. All the trajectories have a very small and smooth decrease in bank-angle during the middle of the flight, du to change in aerodynamic coefficients. In addition, a sudden, small step occur right before the end.In Figure 4.34, duration and trigger time of the guidance systems remain quite the same as in the previous case. All trajectories deviate in the same way, because the atmospheric profile is always the same. All the thousand simulated trajectories do share the same behavior, with an initial small bump and then a rather fast decrease in bank-angle. Many even reach saturation, which is what causes some trajectories to fly few hundreds of meters lower than the target apoapsis: this is clear by the fact that any Monte Carlo flown with a GRAM-99 *c* atmosphere has an average apoapsis lower than target. This trend in decreasing the bank-angle (and thus increasing cosine) is due to the fact that the ratio between the currently used profile and the US76 atmosphere used by the NPC is continuously increasing from an altitude of 70 km to 100 km, as shown by Figure 3.6.

In Figure 4.36 the trajectories really start to look different from one another. In some cases the bank-angle deviates towards larger values, in other cases towards lower values, down to saturation. Since the basic profile is the same as before, the latter case is more likely. Also, the number of changes in derivative of the bank-angle is variable. At this point the trajectories are very different from each other, despite the large-scale perturbations model allows for maximum deviations in density of about 20 %, whereas the deviations in the previous two cases could reach up to 50 %.

The last case is the one including the small-scale perturbations. These trajectories look very noisy, and they are. Also, it should be kept in mind that the density variations are filtered: if it weren't for that, the changes in bank-angle would be much larger (even though they would still be limited by the attitude dynamics). In this case, the number of inversions of $\dot{\sigma}$ cannot be counted. A general trend (valid also for the GRAM-99 *l* atmosphere) is divergence: in fact, the longer the shorter the remaining part in the atmosphere, the larger the correction in bank-angle must be to counteract the perturbation. This often causes saturation, which only rarely causes important differences in in

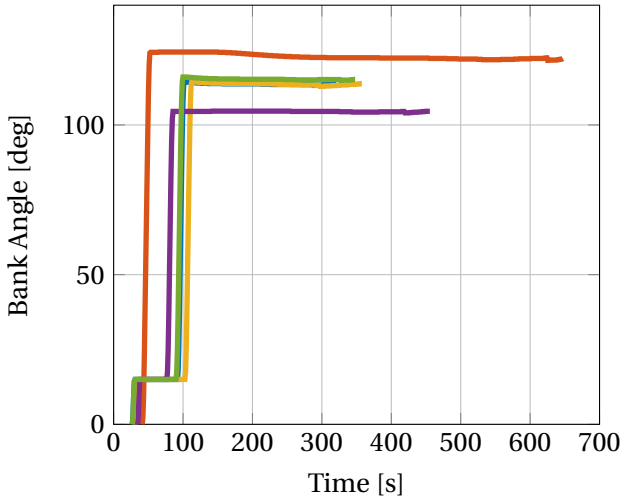


Figure 4.34: Bank-angle history for 5 trajectories flown with the *Lu* guidance in the US76 *c* atmosphere.

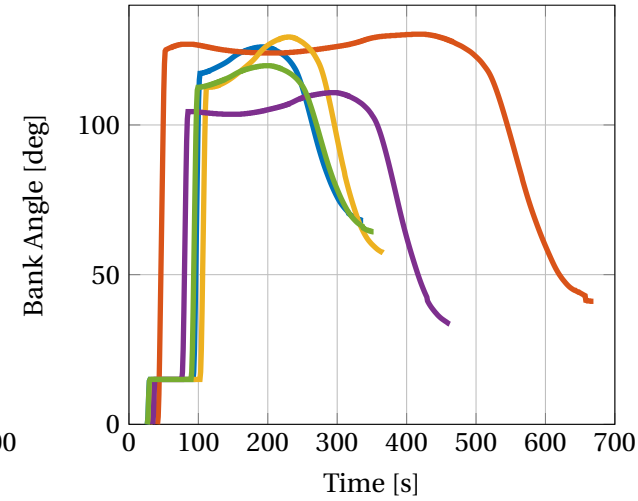


Figure 4.35: Bank-angle history for 5 trajectories flown with the *Lu* guidance in the GRAM99 *c* atmosphere.

apoapsis altitude.

Hence, it has been shown in detail why trajectories simulated in an environment that includes either large or small-scale perturbations have much larger inaccuracies, despite their smaller dispersions in density. These sort of perturbations do not cause any difference in average performance, but are very important when evaluating the robustness of a concept. It is more significant to evaluate the robustness of a guidance in an *l* or *ls* atmosphere than in a *c* atmosphere, even if in the latter the dispersions were much larger.

Some bigger issues occur when using the modified damped Newton solver in the *ls* environments. In such a case, the large noise caused by the small-scale perturbations and the turbulence makes it more likely that the guess from the previous iteration leads to either a crash or a hyperbolic exit. In those environments, the average error in apoapsis altitude of the $modlat$ using the modified damped Newton solver is about 4 times that of the other concepts. Of course this might be due to the concept itself, instead of being caused by the solver, but there is no evident reason why that should be the case. It is then believed that such a difference is caused only, or in most part, by the solver. Moreover, since the main problem is that of using the previous iteration as initial guess, it is possible that the same problem might occur when using an APC guidance, such as the concepts by Cerimele and Gamble [1985], Masciarelli et al. [2000], or Hamel and Lafontaine [2006]. All such concepts rely indeed on the previous command as an initial guess.

4.8.7 CONCLUSIONS

During this chapter it has been shown that aerocapture can be ensured with any of the *Lu* and *mod* concepts, as well as with the *simple* concept. A reliable lateral logic is still needed though.

The *Lu* and *mod* concept can satisfy the requirement GD-1 of Section 2.7 ensuring the following bounds.

1. $\Delta V_{max} = 90\,\mathrm{m\,s^{-1}}$, in-plane.
2. $Q_{max} = 650\,\mathrm{MJ\,m^{-2}}$.
3. $n_{max} = 5$. These optimal concepts derived from Lu et al. [2015] are the only ones, among those tested, that can meet the requirements on load factor for manned aerocapture.

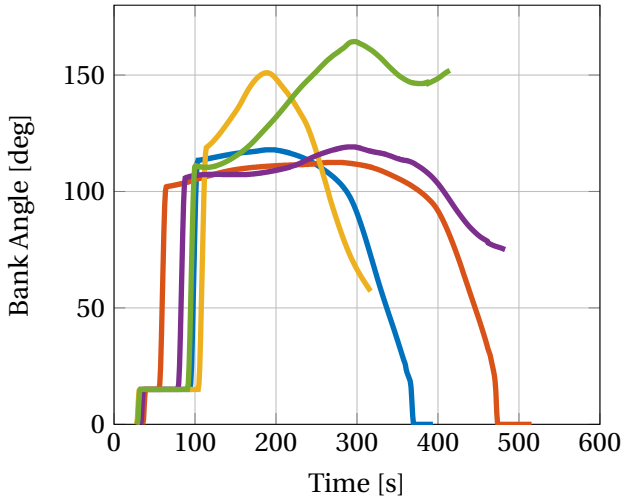


Figure 4.36: Bank-angle history for 5 trajectories flown with the *Lu* guidance in the GRAM99 *l* atmosphere.

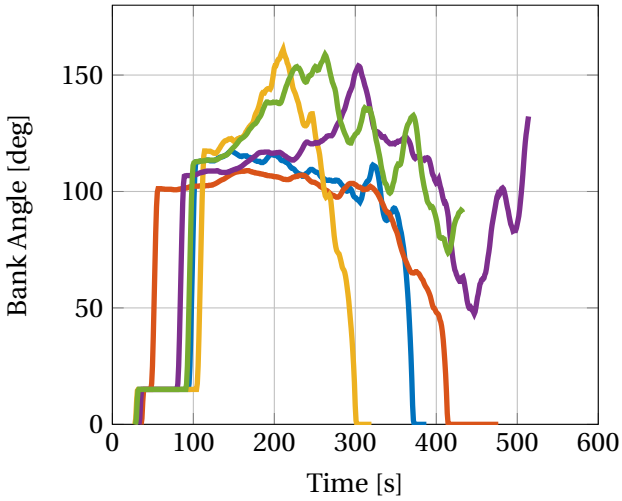


Figure 4.37: Bank-angle history for 5 trajectories flown with the *Lu* guidance in the GRAM99 *ls* atmosphere.

4. $q_{max} = 15\,\mathrm{MW\,m^{-2}}$. There was no attempt to look for a range to minimize such a value.

This maximum values are all ensured for a range of flight-path angles spanning from a minimum of $-5.7°$ to $-5.1°$.

The aim of this chapter was, among many that of answering the two sub-questions:

What are the most useful parameters to evaluate optimality in aerocapture?

How does the state-of-the-art guidance solution perform in highly perturbed atmospheres?

The first one is easily answered. The total ΔV is the main parameter to be considered when optimizing aerocapture, at least from lunar return conditions, on Earth. It was shown mathematically that the same trajectory minimizing the ΔV would either minimize or maximize the heat load. Testing showed that the first situation is true for the case in object. One limitation is the entry velociy, since radiative heat load follows different equations at different entry velocities, and the relative importance of radiative and convective heat fluxes also change. Another limitation is the atmospheric composition, for the same reasons. Before generalizing this result one should therefore be careful. An indication of the fact that this conclusion could be generalized is given by the work of Robinson et al. [2009], who show that for some ballistic coefficients, the higher the entry velocities, the more there is a difference in total heat load between a full lift-up and a full lift-down trajectory. Indeed, ballistic coefficients also play a role. In any case, the trajectory minimizing the total heat load would still be of the bang-bang kind, but maybe opposite.

It was also expected that the same optimal trajectory would minimize heat rate and load factor constraints, and the simulations confirmed the expectations. This result is instead more general. As the minimization of the ΔV, the minimization of the constraints goes against robustness. A mission dependent trade-off for these should also be done.

Hence, by minimizing the ΔV, one obtains a trajectory that also minimizes any constraints, as well as the total heat load. When including the ΔV due to change of plane, the constraint on final inclination can also be removed. This means that a potentially multi-objective, constrained optimization is reduced to a single objective unconstrained optimization problem. In view of using RL, this is a very important and useful finding.

On what concerns the second subquestion, it has been shown that large-scale and small-scale perturbations do indeed affect the performance and robustness of the optimal NPC by Lu. They can cause apoapsis that are 100 km higher than the target. Nonetheless, their effect is fortunately asymmetric, because, as explained in Section 2.5.2, lift-up trajectories are more stable than their counterpart. Hence, these perturbations never cause any crashes. Moreover, they generally do not affect the performance in terms of ΔV more than the constant perturbations do.

These conclusions, together with some minor ones, can be summarized in a few statements:

1. The optimal NPC guidance from Lu is also optimal in terms of minimizing load factor and heat flux peaks, without any need of predictive load-relief, and total integrated heat load. The latter is a consequence of including radiative heat into the equation, which becomes dominant at very high speed entries. In addition, it can meet the requirements for manned aerocapture, for the mission examined here.

2. Small-scale and large-scale perturbations affect the robustness of a guidance logic much more than constant density perturbations. This is especially true for guidance logics whose solvers use the previous iteration's bank-angle as initial guess. This is the case of the modified Netwon-Raphson, but it might cause problems also to some APC guidance concepts. This is problematic, since this category of guidance logics (those using the previous command as initial guess) are also the fastest ones. This sort of perturbations are also important to quantify the realiability of the lateral guidance proposed in the *modlat*. In fact, in such environments, the robustness of the lateral guidance cannot be ensured.

3. The *mod* guidance is more robust in the sense that it requires much less tuning. It also has an averagely better performance than the original one, but this is probably due to the fact that the tuning of the original guidance was done for the Orion MPCV.

4. The lateral guidance proposed in this chapter is not a very robust concept. It performs particularly bad when large-scale and small-scale perturbations are active. An easy improvement can be included, which would make it more similar concept by Shen and Lu [2004]. Nonetheless, because of the feature stated at the end of Subsection 4.5.2, the large errors in the *ls* environment are likely caused by the coupling with the modified Newton-Raphson method.

5. The modified Newton-Raphson method seems to be an acceptable solution. This means that the number of iterations can be reduced from 10-20 to 2 iterations per guidance call. It suffers particularly turbulent and noisy environemnts, displaying an average error 4 times larger than the bisection-secant method in the same environment. In addition, its success cannot be ensured, proved or demonstrated in any way.

6. There is a large range of initial flight-path angles for which all the performance parameters are rather constant, and very close to optimal. An optimal and robust aerocapture should aim to have a nominal entry flight-path angle in the middle of such range. In such a range, aerocapture can be ensured with a maximum final $\Delta V = 80\,\mathrm{m\,s^{-1}}$, a total heat load of $650\,\mathrm{MJ\,m^{-2}}$, a maximum heat rate well below $18\,\mathrm{MW\,m^{-2}}$, and a load factor that does not exceed 5.

7. Not including the effect of J_2 in the propagation of the exoatmospheric leg causes errors in apoapsis up to 8 km in altitude. This is a result expected from the theory of perturbations,

since according to Wakker [2015], the effect of J_2 causes oscillations in semi-major axis of up to 20 km.

In addition, from these conclusions, two general guidelines can be drawn for the next chapters, in which an artificial intelligence will be designed to guide the vehicle:

1. The design of the artificial intelligence can focus on the minimization of (in and out-of-plane) ΔV only: this will automatically lead to an aerocapture that also minimizes any sort of constraints peaks, as well as the total heat load. For this reason, in the artificial intelligence concepts, there will be no analysis of any parameter other than the ΔV.

2. It is of valuable interest generating a guidance that is faster than the previously evaluated ones. This is despite the success of the modified Newton-Raphson method, since its capability cannot be generalized.

Table 4.3: Summary of *modlat* guidance systems performances, outliers included.

		In-plane ΔV [$\mathrm{m\,s^{-1}}$]				$\|\Delta i\|$ [°]	Δi [°]				ΔV_{tot} [$\mathrm{m\,s^{-1}}$]				$\|\Delta r_{apo}\|$ [km]	r_{apo} [km]			
σ_d	Atmosphere	Mean	Min	Max	Std	Mean	Mean	Min	Max	Std	Mean	Min	Max	Std	Mean	Mean	Min	Max	Std
110	GRAM99 *ls*	82.1	47.4	577.3	49.4	0.17	-0.08	-0.60	0.83	0.22	87.5	48.9	577.4	50.0	3.49	202.3	141.1	368.1	12.2
110	GRAM99 *l*	81.3	46.3	566.5	49.1	0.14	-0.05	-0.56	0.78	0.19	85.0	51.5	566.6	50.1	1.80	201.2	142.5	299.3	7.9
110	GRAM99 *c*	82.8	47.9	692.5	57.1	0.12	-0.02	-0.28	0.85	0.19	85.6	47.9	692.6	58.6	0.45	199.7	128.5	207.1	4.2
110	US76 *ls*	83.9	46.1	584.4	51.3	0.14	-0.03	-0.60	0.84	0.20	87.5	49.7	584.5	52.4	3.69	203.1	140.5	335.9	9.0
110	US76 *l*	82.6	49.3	573.4	51.2	0.12	-0.01	-0.44	0.82	0.19	85.4	51.2	573.5	52.7	1.54	201.0	141.9	228.1	4.2
110	US76 *c*	84.1	47.2	702.4	59.4	0.09	0.03	-0.09	0.93	0.19	86.3	48.4	702.5	61.6	0.38	199.6	127.9	200.0	4.3
120	GRAM99 *ls*	80.1	46.2	577.3	49.6	0.20	-0.06	-0.91	1.04	0.27	87.8	46.3	577.4	50.8	4.94	203.8	141.1	425.9	16.6
120	GRAM99 *l*	79.2	43.1	566.5	49.4	0.16	-0.02	-0.61	0.88	0.23	84.4	49.2	566.6	51.0	1.99	201.4	142.5	294.1	8.3
120	GRAM99 *c*	80.5	47.9	692.5	57.3	0.14	0.00	-0.28	0.97	0.22	84.5	47.9	692.6	59.3	0.46	199.8	128.5	206.2	4.2
120	US76 *ls*	81.6	45.9	584.4	51.5	0.17	-0.00	-0.68	0.99	0.25	87.2	47.6	584.5	53.6	4.79	204.2	140.5	333.2	10.9
120	US76 *l*	80.2	46.3	573.4	51.5	0.14	0.02	-0.48	0.90	0.23	84.4	46.6	573.5	53.8	1.96	201.5	141.9	240.0	4.7
120	US76 *c*	81.4	47.2	702.4	59.7	0.12	0.06	-0.08	1.14	0.23	85.3	48.4	702.5	62.9	0.38	199.6	127.9	200.0	4.3

Table 4.4: Summary of *modlat* guidance systems performances, outliers excluded.

		In-plane ΔV [$\mathrm{m\,s^{-1}}$]				$\|\Delta i\|$ [°]	Δi [°]				ΔV_{tot} [$\mathrm{m\,s^{-1}}$]				$\|\Delta r_{apo}\|$ [km]	r_{apo} [km]			
σ_d	Atmosphere	Mean	Min	Max	Std	Mean	Mean	Min	Max	Std	Mean	Min	Max	Std	Mean	Mean	Min	Max	Std
110	GRAM99 *ls*	78.9	47.4	339.3	35.9	0.17	-0.08	-0.60	0.83	0.22	84.4	48.9	339.5	37.2	3.28	202.6	194.1	368.1	11.8
110	GRAM99 *l*	78.1	46.3	334.1	35.4	0.14	-0.06	-0.56	0.78	0.19	81.8	51.5	334.3	37.2	1.58	201.4	198.2	299.3	7.3
110	GRAM99 *c*	78.7	47.9	363.0	38.2	0.12	-0.02	-0.28	0.85	0.19	81.4	47.9	363.2	40.7	0.10	200.1	200.0	207.1	0.0
110	US76 *ls*	80.6	46.1	346.2	38.1	0.14	-0.03	-0.60	0.84	0.20	84.3	49.7	346.4	39.9	3.47	203.4	196.7	335.9	8.4
110	US76 *l*	79.4	49.3	340.4	37.9	0.12	-0.01	-0.44	0.82	0.19	82.2	51.2	340.6	40.2	1.30	201.3	199.8	228.1	2.8
110	US76 *c*	79.8	47.2	370.9	40.9	0.09	0.03	-0.09	0.93	0.19	82.1	48.4	371.1	44.2	0.01	200.0	200.0	200.0	0.0
120	GRAM99 *ls*	76.9	46.2	339.3	36.0	0.20	-0.06	-0.91	1.04	0.27	84.6	46.3	339.5	38.3	4.74	204.1	193.9	425.9	16.4
120	GRAM99 *l*	75.9	43.1	334.1	35.7	0.16	-0.02	-0.61	0.88	0.24	81.2	49.2	334.3	38.3	1.76	201.7	198.3	294.1	7.7
120	GRAM99 *c*	76.3	47.9	362.9	38.2	0.14	0.00	-0.28	0.97	0.22	80.3	47.9	363.0	41.6	0.11	200.1	200.0	206.2	0.2
120	US76 *ls*	78.3	45.9	346.2	38.3	0.17	-0.00	-0.68	0.99	0.25	84.0	47.6	346.4	41.5	4.59	204.5	196.5	333.2	10.4
120	US76 *l*	76.9	46.3	340.4	38.1	0.14	0.02	-0.48	0.90	0.23	81.1	46.6	340.6	41.5	1.72	201.7	199.9	240.0	3.5
120	US76 *c*	77.2	47.2	370.8	41.1	0.12	0.06	-0.08	1.14	0.23	81.1	48.4	371.0	46.0	0.01	200.0	200.0	200.0	0.0

Table 4.5: Summary of *Lu* guidance performances, outliers excluded.

		In-plane ΔV [$\mathrm{m\,s^{-1}}$]				$\|\Delta r_{apo}\|$ [km]	r_{apo} [km]			
Guidance	Atmosphere	Mean	Min	Max	Std	Mean	Mean	Min	Max	Std
Lu	GRAM99 *ls*	88.1	62.0	324.6	30.7	1.17	200.5	193.3	256.2	4.2
Lu	GRAM99 *l*	87.6	62.5	351.0	31.5	0.45	200.2	197.7	218.0	1.6
Lu	GRAM99 *c*	87.3	48.0	346.2	32.0	0.06	200.1	200.0	200.1	0.0
Lu	US76 *c*	89.5	47.9	353.9	33.8	0.01	200.0	200.0	200.0	0.0

Table 4.6: Summary of *mod* guidance performances, outliers excluded.

		In-plane ΔV [m s^{-1}]				$\left\|\Delta r_{apo}\right\|$ [km]	r_{apo} [km]			
σ_d	Atmosphere	Mean	Min	Max	Std	Mean	Mean	Min	Max	Std
$\sigma_d = 120°$	US76 *c*	79.2	48.0	353.9	36.5	0.01	200.0	200.0	200.0	0.0
$\sigma_d = 120°$	GRAM99 *c*	78.1	48.0	346.2	34.4	0.09	200.1	200.0	200.2	0.0
$\sigma_d = 120°$	GRAM99 *l*	78.1	49.1	351.0	33.9	0.86	200.6	197.6	263.0	3.8
$\sigma_d = 110°$	US76 *c*	87.1	48.0	353.9	35.8	0.01	200.0	200.0	200.0	0.0
$\sigma_d = 110°$	GRAM99 *c*	85.2	48.0	346.3	33.8	0.38	200.1	200.0	200.1	3.1
$\sigma_d = 110°$	GRAM99 *l*	79.1	50.9	330.5	32.3	1.12	200.9	197.5	264.2	4.9
$\sigma_d = 110°$	GRAM99 *ls*	85.7	56.6	324.5	32.4	0.98	200.2	194.5	255.7	3.1

Table 4.7: Summary of *simple* and *predload* guidance performances, outliers excluded.

		In-plane ΔV [m s^{-1}]				$\left\|\Delta r_{apo}\right\|$ [km]	r_{apo} [km]			
Guidance	Atmosphere	Mean	Min	Max	Std	Mean	Mean	Min	Max	Std
simple	US76 *c*	194.2	47.9	385.0	86.3	0.01	200.0	200.0	200.0	0.0
simple	GRAM99 *ls*	192.1	64.0	370.5	84.4	0.23	199.8	196.7	203.2	0.4
predload	US76 *c*	137.1	44.0	1,090.0	85.1	7.70	207.6	168.0	3,867.6	139.6
predload	GRAM99 *ls*	129.9	54.9	608.2	82.4	0.86	199.5	140.4	227.0	3.8

5

LEARNING

The main research question stated in Section 1.1 asks:

> *Can an artificial intelligence guide a spacecraft to achieve aerocapture in an optimal and robust way?*

Up to this point, artificial intelligence has been described only quantitatively. It is now of interest to understand how artificial intelligence, and in general, artificial learning, works. Also, it is desirable to understand how such a technology can be implemented to solve the problem in question.

This chapter aims to provide the foundations of artificial intelligence required to develop a neural guidance. It will be shown how and why an ANN is a good candidate as a guidance logic, and how and why both supervised learning and RL can be used to optimize the ANN. Before any description of the concepts or of the mathematics involved in artificial intelligence, it is desired to describe why such a technology should be of interest. This is done in the first section of this chapter.

Then, in Sections 5.2 and 5.3, it will be shown how such a technology can be used in light of the optimal aerocapture guidance problem, stated in Section 2.7. In those sections the artificial intelligence will be treated as a black-box. At that point, the only things one would need to know is that a neural network is, ideally, capable to reproduce any desired nonlinear function. In the Section 5.2, it will be shown how a network will be set to be used as a guidance logic. In Section 5.3, a few examples are given, of how such a technology has successfully been used for control in aerospace problems, with a stress on guidance for Solar System Exploration.

Afterwards, the more theoretical part of this chapter begins with Section 5.4. At first, artificial neurons and different neural networks are explained. Later on, different training methods are discussed. These include both supervised learning methods, in which the network is taught from existing data, as well as RL, in which the network learns from experience. Concerning the latter, a stress will be given on the so-called policy gradient methods, and specifically the PGPE. Motivations for the choice of certain methods instead of others will also be given. The chapter also includes the verifications for all the algorithms implemented for this research.

5.1 MOTIVATION

The main reason why a neural network is of interest for the optimal aerocapture guidance problem is the fact that it is an object that can give a closed-form approximation of any nonlinear function. In addition, such an approximation is given almost instantaneously, since it is the result of a large quantity of simple nonlinear function, which can be computed, in most part, in parallel.

In Section 4.3 it was displayed how the guidance had to solve iteratively a nonlinear function at each call. Such a solution required the propagation of the entire trajectory at least twice, but up to 20 times, depending on the solver in use. Also, the obtained solution was rather simple, in the sense that it assumed a constant bank angle for almost all the entirety of the remaining trajectory (in the *mod* the bank-angle was linearly varying in the initial seconds of the propagated trajectory).

Hence, in theoretical terms, the solution given by the optimal aerocapture NPC can still be improved, in two ways:

1. Solve the problem faster.
2. Allow for more complex solutions.

There is not much to discuss about the first item: a neural network would sure be faster than an NPC, unless networks with hundreds of layers were used[1].

Concerning the second item, a natural idea would be that of teaching a network how to give the output of an optimal trajectory at any state. Instead of needing an optimization process at each guidance call, the network would be able to output the corresponding bank-angle almost instantaneously. Nevertheless, such a function should be unique at every state: this means that a global optimum should be sought off-line for every possible state. Even off-line, aiming to obtain a global optimum for a variety of trajectories and conditions can be extremely demanding. An alternative would be modifying the function such that it would dispay a single global optimum. One way of doing this is convexification [Acikmese and Blackmore, 2011]. Such a process is very demanding though. An additional problem comes from the fact that, as shown in Section 4.3, the optimal aerocapture trajectory is inherently not robust.

Hence, in this research, supervised learning will be used to simply imitate the *simple* concept, which gives a constant bank-angle trajectory. The reasons why this would still be interesting are the following:

1. It is believed that a neura network trained to give the same input as the *simple* concept could easily be made imitate the *Lu* concept, by setting $\sigma = \sigma_0$ as long as $\sigma_{output} < \sigma_d$.
2. If such a guidance were able to properly imitate the *Lu* concept, it would be preferred to it, since it would be able to do so faster than the former concept. This is in accordance with what stated about real-time requirements at the end of Section 2.7.
3. A network trained with supervised learning could be used as an initial guess for a RL training, which will be later explained.

Despite what stated as a third motivation, a network trained with supervised learning will not be used as initial guess during this research, mainly because that would have required further effort. The third reason could then relate to recommendations for future work.

Given the fact that the optimal trajectory is not robust, RL provides a very interesting possible solution to the problem. RL is in fact mainly used for a Markovian decision process (MDP). In this class of problems, which will be described more in detail in Section 5.8, the environment is not deterministic. Hence, without going into details, a network trained using RL methods will be able to give the optimal command, given all the possible future perturbations. Such perturbations have to be included in the training model. Hence, the more such a model is realistic, the better the outcome is supposed to be.

Because of what concluded from Chapter 4, the network will be trained with RL having as only objective that of minimizing the final ΔV. This leads to some issues though, which will be described in Section 6.6, and solved.

[1] At the best of the author's knowledge, the current state-of-the-art of deep networks consists of using around 20 layers. Still, this research will make use of much smaller networks than those.

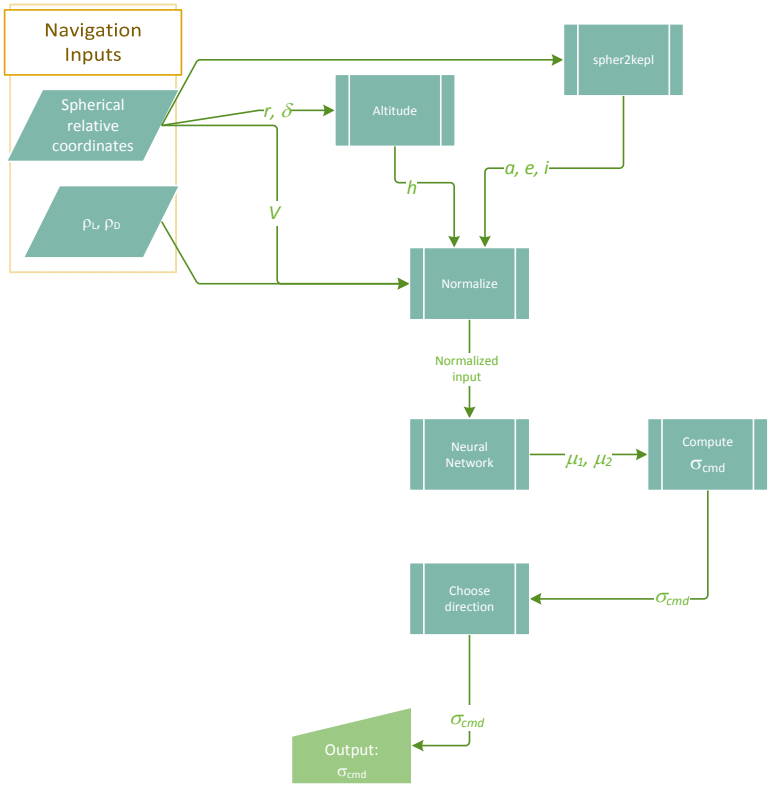


Figure 5.1: Architecture of a neural guidance.

From the description just given, a network trained with RL can theoretically be a very powerful tool. Such a potential does not come without any drawbacks, as it will be specified in Section 5.8.

5.2 NEURAL GUIDANCE DESIGN

Before giving any example, it will be shown here how an ANN will be used in this research. While this is being described, the reader should refer to Figure 5.1. At this point, the neural network is still to be considered as a black-box, and its proper functioning will be clearer at the end of this chapter.

The navigation input is first transformed into the inputs that is used for the neural guidance, and then normalized. The choice for the network inputs will be explained in Section 6.4. Normalization is needed because a neural network works best with inputs on the order of magnitude of about 1, and symmetrical around zero. The neural network then elaborates the input, and gives an output. The commanded bank-angle is eventually a function of that output. An example of such a function is, in the case of aerocapture, given in the next section, by Equation (5.1).

5.3 NEUROCONTROLLERS AND NEURAL GUIDANCE

The output of a controller is a non-linear function of the state; for the computations to occur real-time, such function has to be relatively simple. Thus, it is difficult to have complex control functions.

A neural network can be trained to approximate the input of an optimal control problem: when used as a controller, it would then be able to give the approximate output in real-time of a very complex and computationally demanding function. The output could satisfy the requirements for real time control, since the number of computations done by a neural network is very limited.

As an example, Zhou [2002] applied the concept of teaching an ANN to follow an optimal-control problem to the guidance of homing missiles. The use of neural networks as controllers is widespread nowadays, and in spaceflight is not rare, especially in academic research. In addition, neural networks have been used to approximate the behavior of specific controllers, such as inverse dynamics control [Zhou et al., 2014], H_∞ control for nonlinear systems [Cheng and Lewis, 2007]. Eventually, very recently Sanchez-Sanchez et al. [2016] used, with excellent results, deep networks for terminal landing guidance. The networks successfully reproduced optimal control trajectories in closed-loop simulations.

The use of a universal approximator to reproduce optimal control solutions is clearly very attractive. However, something better could even be done, with RL. This branch of artificial intelligence, described in Section 5.8, allows an adaptive element such as an ANN to learn the control that is optimal for a stochastic environment. As an example, Gelly and Vernis [2009] used neural networks trained with RL as a guidance system for aerocapture. Always Gelly and Vernis [2009] applied the same concept to terminal landing guidance.

5.3.1 NEURAL GUIDANCE IN SOLAR SYSTEM EXPLORATION

This subsection aims to provide two examples in which neural guidance was used in solar system exploration. These are Gelly and Vernis [2009] and Sanchez-Sanchez et al. [2016]. The first one was chosen because it is a case of RL applied to the same problem of this research, together with another case of applied to terminal landing; the second because it used supervised learning to teach deep neural network (DNN)s optimal control solutions.

The guidance developed by Gelly and Vernis [2009] was applied to the MSR mission, and consisted of an aerocapture on Mars with an entry velocity of $5687\,\mathrm{m\,s^{-1}}$, initial flight-path angle of -10.24° at 120 km altitude. The target orbit had an inclination of 50° and an apoapsis altitude of 500 km.

They used an feed forward neural network (FFNN) with five neurons in the input layer, twelve neurons in the hidden layer, and two neurons in the output layer. The two outputs μ_1 and μ_2 were defined such that:

$$\begin{aligned} \cos\sigma_{cmd} &= \frac{\mu_1}{\sqrt{\mu_1^2+\mu_2^2}} \\ \sin\sigma_{cmd} &= \frac{o_2}{\sqrt{\mu_1^2+\mu_2^2}} \end{aligned} \tag{5.1}$$

The inputs consisted of orbital energy, eccentricity and inclination of the current orbit, velocity (it is not specified whether it was the relative or the inertial velocity) and non-gravitational acceleration.

In their work, the optimization used was a genetic algorithm, consisting of 20 individuals evolved along 500 generations, with a mutation rate of 1%. To each individual corresponds a specific set of parameter for the neural guidance. The fitness of each individual was evaluated according to its performance in 600 dispersed simulations. It is easy to compute that the training involved the simulation six million trajectories. For this reason, they needed to use a simplified model during the training. As it will be explained in Section 5.9, this is the major drawback of actor-only methods.

In the first generations, the fitness function was such to heavily punish guidance laws leading to crash or skipout. In the later generations, once the guidance was such that target apoapsis was ensured with a limited error, the fitness function was then changed such to minimize the final ΔV and the bank angle consumption, which is a scalar value somehow proportional to the integral in

time of $\dot{\sigma}$. The inclusion of the bank angle consumption in the fitness function was very important to allow the guidance to achieve aerocapture with one only bank reversal, which is a very interesting capability, as it was stated in Chapter 4.

The guidance was then evaluated on 1000 Monte Carlo dispersed trajectories, different from the 600 used during training. Their guidance scored, on average, much better than their benchmark traditional guidance (called feedback trajectory control) in all the performance parameters that were evaluated: maximum g-load, maximum heat-flux, accuracy in apoapsis altitude and in inclination, final ΔV needed. It also turned out to be more robust than the benchmark, by scoring better than it in the worst case for all parameters except for one, the altitude error at apoapsis. Thus, improvement in robustness can be made. Moreover, the worst case for the neural guidance was, for many parameters, better than the average case of the benchmark guidance, specifically for error in inclination and final ΔV. Analysing Figure 5.2, it seems that in this case the neural guidance behaves as one would expect, by following, with some margins, a bang-bang trajectory, except for the very initial states, in which it is not very clear what is happening.

Their analysis has two drawbacks, however: one consists of the fact that an extremely large number (6,000,000) of simulated trajectories has been used during training. The second is the fact that they have not completely exploited the RL framework at its best, since they did not use very perturbed environments in the training (they used the equivalent of the US76 *c*, but for Mars, and pretty much the same kind of variations in initial conditions that were used in this research).

Gelly and Vernis [2009] used the exact same method to optimize also the terminal landing phase on Mars. It is of no interest now discussing their exact setup. However, it is interesting to look at Figure 5.3: the neural guidance that minimizes the consumption behaves in a way that is hard to understand, showing two peaks in thrust, whereas the Apollo-E guidance behaves relatively smoothly, despite showing a few "hiccups". This shows another problem of RL: it is likely that the resulting network may behave in ways difficult to understand from an engineering point of view, making the verification process harder. This is instead more rare in supervised learning.

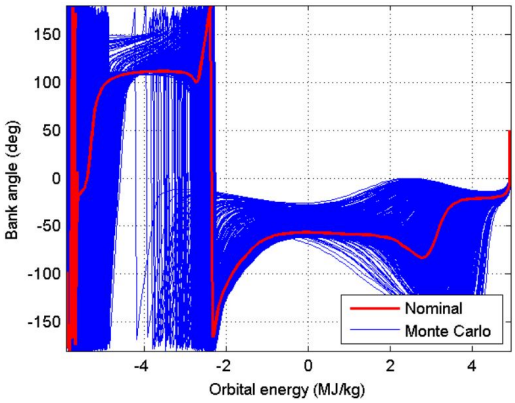


Figure 5.2: Bank-angle profile of neurally guided aerocapture [Gelly and Vernis, 2009].

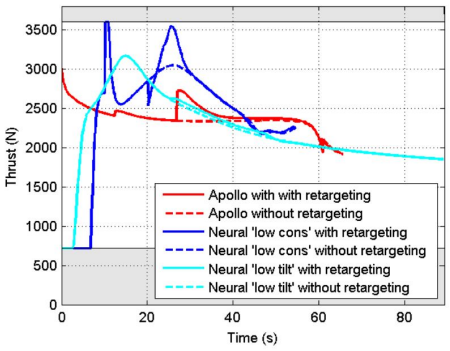


Figure 5.3: Command history of neurally guided terminal landing [Gelly and Vernis, 2009].

Eventually, the work done by Sanchez-Sanchez et al. [2016] is worth being mentioned. They showed how single layer networks cannot be taught (with supervise learning) to successfully approximate optimal control solutions for terminal landing, and DNNs should be used for that purpose. Their training is based on 150 thousand optimal control trajectories, with 100 data points for each trajectory, and is done with stochastic gradient descent. Moreover, they also showed how DNNs with rectified linear units (ReLU) as activation functions are more successful for the goal.

This concludes the description of ANNs used in examples similar to that of aerocapture guidance. It has so far been shown that the neural guidance schemes have always turned out to be very high performing, both using RL and supervised learning. In some cases, they might give outputs

that are counterintuitive to an engineer. An additional drawback would be the fact that they are computationally very demanding to program. This is not a problem per se, but it implies the fact that if there are sudden changes in mission requirements, a different solution should be used.

5.4 ARTIFICIAL NEURONS

The neuron is the fundamental element of an ANN. Its task is quite simple, and consists of three phases: first, it multiplies every input x_i it receives by their respective weights w_i and adds a defined threshold value, or bias, b to it; then, it sums the weighted inputs, and eventually it uses an activation function. Thus, in general, the computations done by a neuron are:

$$Y = f_N(X), \tag{5.2}$$

$$X = \sum_{i=1}^{n} x_i w_i - b, \tag{5.3}$$

where Y is the output.

The most commonly used activation functions are the sign function, sigmoids, and ReLU.

Among the sigmoid functions, the one that will be used in this research is the hyperbolic tangent:

$$f(X) = \frac{e^{2X} - 1}{e^{2X} + 1}, \tag{5.4}$$

The importance of having a monotonic and non-linear function is due to Kolmogorov [1957] theorem, and will be explained later.

A neuron can therefore be characterized by:

- its activation function f_N;
- its threshold, or bias b;
- its weights w_i.

In a neural network, usually the activation functions are the same for all the neurons. Thus, a neuron is characterized only by the so-called augmented vector **w** [Engelbrecht, 2007]:

$$\mathbf{w} = \begin{bmatrix} w_1 & w_2 & \dots & w_{n_{l-1}} & b \end{bmatrix}^T \tag{5.5}$$

where n_{l-1} is the number of neurons in the precedent layer. Consequently, every layer is characterized by a weight matrix of the size $(n_{l-1}+1) \times n_l$.

5.5 FEEDFORWARD NEURAL NETWORKS

As neurons in brains, artificial neurons also need to be organized and interconnected within each other; in an ANN, neurons are usually organized in layers. The most common structure is the FFNN, where every neuron's output is one of the inputs for all the neurons of the following layers.

Although interesting, a discussion on networks topology[2] is not necessary, since it will be shown that FFNNs have all it needs to be used as a neurocontroller, and, being the simplest and most used kind, it would be unnecessary to try to use other topologies.

In an FFNN neurons are arranged into layers. Each neuron receives weighted inputs from all and only the neurons of the previous layer, and then sends their input to all and only the neurons of the following layer.

[2] In mathematics, topology is the study of space under continuous deformation. Network topology refers instead to how a network is arranged. In this report, the word topology will always refer to this second meaning.

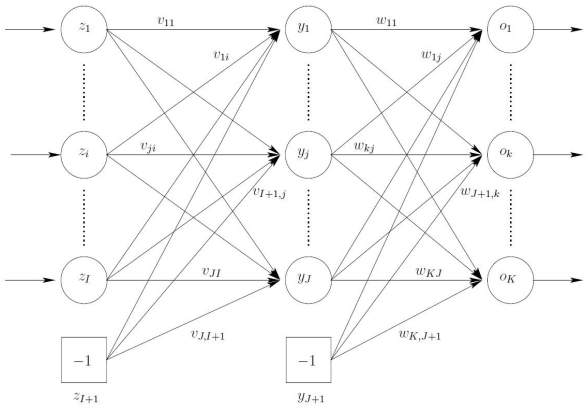


Figure 5.4: Feed Forward Neural Network architecture [Engelbrecht, 2007].

The output of a one-hidden layer FFNN can be computed as follows (assuming that the threshold b is the weight of an additional neuron whose output is -1, as in Figure 5.4):

$$\begin{aligned} o_{k,p} &= X_{ok,p} \\ &= f\left(\sum_{j=1}^{n_h+1} w_{kj}\, f\left(X_{y_{j,p}}\right)\right) \\ &= f\left(\sum_{j=1}^{n_h+1} w_{kj}\, f\left(\sum_{i=1}^{n_i+1} w_{ji} z_{i,p}\right)\right) \end{aligned} \tag{5.6}$$

assuming that the activation function is the same for every neuron in every layer.

Kurková [1992] demonstrates, using Kolmogorov's theorem, that, provided that there is an appropriate number of neurons, a FFNN with one hidden layer that uses a monotonic and non-linear activation function can approximate any continuous function of the kind $\mathbb{R}^{n_{in}} \rightarrow \mathbb{R}^{n_{out}}$; moreover, it is proved that, when using that same activation function, and using two hidden layers, any discontinuous function can be approximated too. Their theorem has only theoretical importance, and it will be seen in the next subsection that in practice many more than two hidden layers can be used.

An FFNN can be considered as a non-linear mapping of $\mathbb{R}^{n_{in}} \rightarrow \mathbb{R}^{n_{out}}$ [Engelbrecht, 2007], and can be used as a means for universal regression, since they can approximate any function with n_{in} inputs and n_{out} outputs.

A major drawback of FFNNs is that of not giving any insight in the problem. It is in fact very hard to understand what happens within a network, and the difficulty increases with increasing the size of the network itself.

5.6 DEEP NETWORKS

A DNN is a network with more than one hidden layer. There exist different kinds of deep networks, but here only FFNN will be treated, being the most commonly used for control problems.

In many problems, DNNs can have millions of parameters and as many as 20 layers; because of this, they can learn much more complex functions than single hidden layer networks can [LeCun et al., 2015]. Their use will be limited to much smaller networks in this research, and mainly for supervised learning. In fact, they are generally not very popular for reinforcement learning yet, due to their high dimensionality.

Their general advantage (in supervised learning) in control problems with respect to single layer FFNNs with the same number of parameters has been shown by Sanchez-Sanchez et al. [2016] for

optimal terminal landing.Deep networks have the an additional advantage over shallow networks: for DNNs, incurring in a local minimum is not relevant. In fact, most local minima in deep networks perform equivalently well, and therefore initialization is not a problem [LeCun et al., 2015]. The training of a deep network does not differ much from that of a shallow network, and most of the recent contests are being won by DNNs trained by pure supervised learning [Schmidhuber, 2015].

Because of their large size, batch methods for supervised learning are usually not advised for this kind of networks. Most often, stochastic gradient descent (SGD) (see Section 5.7.3), or some variations are used. Nevertheless, Le et al. [2011] advice to use batch methods in minibatches (which would be, anyway, much larger than the batches used in stochastic gradient descent, but not as large as the entire dataset). The two methods mentioned are a limited memory version of the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm, which is a method to approximate the Hessian without the use of second order information, and is, as an example, also used in Sparse Nonlinear OPTimizer (SNOPT) [Gill et al., 2007], and the conjugate gradient.

5.7 SUPERVISED LEARNING

In supervised learning a dataset, that shall include both inputs and outputs, should be provided. Thus, a model that provides the output should be needed. In this case, the most natural solution would be using optimal-control theory, but also other possibilities are available, as it will be explained in Section 6.5. The most efficient way of computing the gradient is backpropagation, described in the coming subsection. The gradient can then be used in different ways depending on the algorithms used. However, before describing these algorithms, the issue of overfitting is mentioned.

5.7.1 BACK-PROPAGATION

The principle behind back-propagation is that of analytical derivative rules. Due to the characteristic topology of FFNNs, however, the method makes extensive use of the chain rule. Supervised learning can be seen as an optimization problem, in which the cost function is the sum of the square errors of all the input output pairs [LeCun et al., 1998]:

$$J = \sum_{p=1}^{n_p} \frac{E^p}{n_p} = \sum_{p=1}^{n_p} \frac{1}{2n_p} \left(\mathbf{Y}_p - \mathbf{f}_{NN}\left(\boldsymbol{\theta}, \mathbf{x}_p\right) \right)^2, \tag{5.7}$$

where n_p is the number of input-output pairs. An important difference between this and a normal optimization problem though is that in this case the goal is not that of searching for the global optimum, since that would be an overfitting solution [Hastie et al., 2001]. Overfitting is a common problem in supervised learning, and will be better analyzed in the next subsection.

Since $\mathbf{f}_{NN}$ is continuous in the parameters $\boldsymbol{\theta}$, it is possible to minimize the cost function using gradient descent. Moreover, the analytical derivative can be found using the chain rule (now for only the p^{th} input-output set):

$$\frac{\partial E^p}{\partial \boldsymbol{\theta}} = \frac{\partial E^p}{\partial \mathbf{f}_{NN}\left(\boldsymbol{\theta}, \mathbf{x}_p\right)} \frac{\partial \mathbf{f}_{NN}\left(\boldsymbol{\theta}, \mathbf{x}_p\right)}{\partial \boldsymbol{\theta}} \tag{5.8}$$

It is then possible to use the chain rule in a layer-by-layer fashion. If each layer were considered as a network itself, whose input are the outputs of the previous layer, and whose outputs are fed as inputs for the next layer. As an example, the derivatives of the error with respect to the parameters of the before-last layer can be computed as follows:

$$\frac{\partial E^p}{\partial \boldsymbol{\theta}_{n_l-1}} = \frac{\partial E^p}{\partial \mathbf{f}_{n_l}\left(\boldsymbol{\theta}_{n_l}, \mathbf{f}_{n_l-1}\right)} \frac{\partial \mathbf{f}_{n_l}\left(\boldsymbol{\theta}_{n_l}, \mathbf{f}_{n_l-1}\right)}{\partial \mathbf{f}_{n_l-1}\left(\boldsymbol{\theta}_{n_l-1}, \mathbf{f}_{n_l-2}\right)} \frac{\partial \mathbf{f}_{n_l-1}\left(\boldsymbol{\theta}_{n_l-1}, \mathbf{f}_{n_l-2}\right)}{\partial \boldsymbol{\theta}_{l_n-1}}, \tag{5.9}$$

where $\mathbf{f}_l$ is the output of the l^{th} layer (and, equivalently, the input to the $(l+1)^{\text{th}}$ layer). The parameters of layer l are those that multiply the outputs of the neurons (and therefore, the parameters that multiply the input belong to the layer before). Generalizing, the derivative of the error with respect to the parameters of the l^{th} layer depends on the derivative of the error with respect to the final output, as well as on the derivatives of the output with respect to the input of any of the subsequent layers, and, eventually, on the derivative of the output of the l^{th}:

$$\frac{\partial E^p}{\partial \boldsymbol{\theta}_l} = \frac{\partial E^p}{\partial \mathbf{f}_{n_l}\left(\boldsymbol{\theta}_{n_l}, \mathbf{f}_{n_l-1}\right)} \frac{\partial \mathbf{f}_l\left(\boldsymbol{\theta}_l, \mathbf{f}_{l-1}\right)}{\partial \boldsymbol{\theta}_l} \prod_{j=l+1}^{n_l} \frac{\partial \mathbf{f}_j\left(\boldsymbol{\theta}_j, \mathbf{f}_{j-1}\right)}{\partial \mathbf{f}_{j-1}\left(\boldsymbol{\theta}_{j-1}, \mathbf{f}_{j-2}\right)}. \tag{5.10}$$

In the previous equation, when $l = j = 1$, $\mathbf{f}_0$ appears, which is the input to the network, x_p. It is only now that it might become clear why the algorithm is called backpropagation. Each derivative, in fact, depends on the subsequent layers. It is also important to notice that the dependence on the following layers does not include the derivative of the output of those layers with respect to their parameters, but only with respect to their inputs. Nevertheless, the latter depends on the input of such layers. At last, it should be noticed how the output of the final layer is a function of all the parameters. Therefore the gradient of the error with respect to any parameter is a function of all of the parameters. The gradient is eventually computed by forward propagating first, to obtain E^p and all the $\mathbf{f}_l$, and then backpropagating to obtain the gradient of E_p with respect to the parameters of any of the previous layers. The missing piece is now that of how to compute the derivative of the output of a layer with respect to both its parameters and its inputs:

$$\frac{\partial \mathbf{f}_l\left(\boldsymbol{\theta}_l, \mathbf{f}_{l-1}\right)}{\partial \boldsymbol{\theta}_l} = \mathbf{f_a}\left(\mathbf{f}_{l-1}\right), \tag{5.11}$$

$$\frac{\partial \mathbf{f}_l\left(\boldsymbol{\theta}_l, \mathbf{f}_{l-1}\right)}{\partial \mathbf{f}_{l-1}\left(\boldsymbol{\theta}_{l-1}, \mathbf{f}_{l-2}\right)} = \boldsymbol{\theta}_l \mathbf{f_a}'\left(\mathbf{f}_{l-1}\right), \tag{5.12}$$

where $\mathbf{f_a}$ is the activation function, and can be different in different layers (in particular in the last one, which is often linear). These formulas can easily be derived when considering that $\mathbf{f}_l = \boldsymbol{\theta}\mathbf{f_a}(\mathbf{f}_{l-1})$. The derivative is slightly different for the weights that involve the biases. The derivative of a hyperbolic tangent with respect to its input function can be computed recursively:

$$f_a(x)' = 1 - f_a^2(x). \tag{5.13}$$

For a ReLU:

$$f_a'(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 & \text{if } x > 0. \end{cases} \tag{5.14}$$

Technically, the gradient should be undefined for $x = 0$. However, using the previous equation is not a problem in practice.

In case of sigmoidal activation functions, the gradient can easily vanish for the first layers of a deep network. That is because $\left\|\mathbf{f_a}'\left(\mathbf{f}_{l-1}\right)\right\| \leq 1$, and the parameters are, on average, smaller than one as well. Moreover, the larger the parameters, the smaller would be the absolute value of $\mathbf{f_a}'\left(\mathbf{f}_{l-1}\right)$. Possible ways to mitigate this problem are simply either not using deep networks, or using different methods other than simple stochastic gradient descent (AdaDelta, described in Subsection 5.7.4 is a good candidate, together with AdaGrad, Subsection 5.7.6), or using ReLU instead of sigmoids, whose derivative is either 1 or 0. Another interesting alternative that has gained popularity recently is extreme learning machine (ELM). In this discipline, networks with only one, very large, hidden layer, with sigmoidal activation functions, are used. The particularity is in the fact that the parameters of the hidden layer are randomly initialized, and never updated. The only parameters to be

updated are those of the output layer, and the model is therefore linear in the parameters (but not in the state) [Huang et al., 2015, 2006]. It has been proven that, even for random parameters in the hidden layer, the so trained networks still have the universal capabilities of single hidden layer FFNNs, despite being linear in the parameters.

It is important to efficiently code the algorithm to not compute any of the values more times than needed. In the end, the computational cost of the gradient with respect to all the parameters is in the same order of magnitude as that of computing the output [Schmidhuber, 2015].

Nowadays, almost every supervised learning algorithm makes use of back-propagation, either to compute the gradient $\partial E^p/\partial \boldsymbol{\theta}$ or the Jacobian $\partial \mathbf{Y}_p/\partial \boldsymbol{\theta}$ for a certain input-output pair. In fact, while still used for unsupervised learning, gradient free methods such as evolutionary strategies or simulated annealing are not popular anymore for supervised learning problems, especially when involving deep networks. Exceptions of course still exist, such as the work done by Rere et al. [2015].

VERIFICATION

A code for back-propagation of neural networks with an arbitrary number of layers was developed during for this research. The reason why this was needed is justified by Subsection 5.7.8. The code is specific for hidden layers with hyperbolic tangents as activation functions, and a linear output layer. Nevertheless, it can be easily modified, to adapt to either ReLU or other activation functions.

It was done for a shallow network first, and a deep network later. The first step consisted of comparing the gradient computed by back-propagation with the gradient computed by finite differences. The latter was computed with different sizes of finite differences. The error between the two converged to almost zero for decreasing finite differences, down to 10^{-8}, and then increased, most likely because of numerical errors. Also, in deep networks, for same finite differences, the error was larger in the first layers than in the last ones.

5.7.2 OVERFITTING

Overfitting is the process by which the regression model ends up learning the noise of the training set. Overfitting can occur if training continues for too long and, at the same time, the network has too many nonlinear parameters. Having too many parameters per se is not a problem, since overfitting can be easily avoided with a couple of methods. Having too few parameters can instead result in having not enough flexibility to capture the nonlinearities in the data [Hastie et al., 2001].

Overfitting is caused by an exceeding number of nonlinear parameters. Therefore, a way to avoid it consists of the so called *weight decay*, or *regularization parameter*: a quadratic weight is given to each weight, such that they are mostly kept close to zero. When a sigmoid, or hyperbolic tangent, activation function is used, a small weight shrinks the function to its linear part. When doing backpropagation, this method has the consequence of simply adding a term $\lambda\boldsymbol{\theta}$ when computing the gradient, where $\lambda \geq 0$ is a tuning parameter.

Another way, more direct, consists of using validation data. The main idea consists of checking, every some training epochs, whether the regression has improved also for some validation data that is not being used during training. Once there is no improvement on the validation set, training is interrupted, and the network is tested on a third set, the test set. Each input-output pair can belong to only one of these sets. Typical way of splitting the data set is using 70 % of the data for training, and 15 % for validating and testing [Engelbrecht, 2007]. However, this method does not avoid overfitting, but only causes the training to stop when overfitting occurs: for this reason, the optimal choice is to use also regularization.

5.7.3 STOCHASTIC GRADIENT DESCENT

Stochastic gradient descent (SGD) is the most commonly used method for training large networks [LeCun et al., 2015]. The concept consists of computing the gradient using backpropagation for only a small set of the input-output pairs, and then update. It is based on the concept according

to which it is possible to obtain a noisy gradient of $\sum_{p=1}^{n_p} E^p$ by evaluating only a small subset of the data. It is called stochastic because the samples used to compute the gradient are randomly picked. Such method has many advantages, such as not having large memory requirements, not easily being stuck in local optima (despite being a gradient descent method, the noise due to the stochasticity of the samples contributes in this), and converging much faster than methods using gradient descent on the entire batch [LeCun et al., 1998].

The general equation for stochastic gradient descent is:

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \alpha \nabla_\theta \sum_{p=1}^{n_m} \frac{E^p}{n_m}, \tag{5.15}$$

where n_m is the size of the minibatch being used, and α is a learning rate. The elements in the minibatch are randomly picked. Due to the noise in the gradient estimates, α needs to be zeroed during training, since the variance of the gradient is proportional to it:

$$\alpha = \min(\alpha_0, \alpha_0 * N/k), \tag{5.16}$$

where N is the number of iterations after which the decay begins. Another option consists of, instead, increasing the size of the minibatch with the number of iterations.

The addition of momentum is often used to speed up the learning. Also, momentum helps in reducing the noise as well, by annealing the components of the direction that are caused by the presence of noise. Momentum is simply added as follows, in a recursive way [Rumelhart et al., 1986]:

$$\boldsymbol{v}^{(k+1)} = \mu \boldsymbol{v}^{(k)} - \alpha \nabla_\theta \sum_{p=1}^{n_m} \frac{E^p}{n_m}, \tag{5.17}$$

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)}, \tag{5.18}$$

were $\boldsymbol{v}$ is the momentum, and μ is the momentum rate. Already three tuning parameters have been introduced so far, $alpha_0$, N, and μ, together with n_p. It is already clear now, that tuning a stochastic gradient descent algorithm may become very hard. However, it is possible to introduce some modifications to make some of the parameters self-adapting. Subsections 5.7.4 to 5.7.6 are all cases of self-adapting algorithms that make use fo SGD and that were implemented during this research.

5.7.4 ADADELTA

AdaDelta is an adaptive learning method first proposed by Zeiler [2012]. According to him, the method has various advantages, the most important of which are probably the fact of not needing any tuning for hyperparameters, together with the implementation of learning rates specific to each parameters. Both these advantages are usually characteristic to batch, second-order learning techniques. Moreover, it does so with very small increase in computational requirements. Therefore, it combines the advantages of stochastic gradient descent together with many of the advantages of second order batch methods.

AdaDelta is based on two main ideas. The first consists of the fact that, especially when training deep networks, the gradients of the parameters of the lower layers can be of orders of magnitude smaller than those of the outer layers. Hence, it may be efficient to have different learning rates for different parameters, being larger for those whose gradient is smaller. This is done by computing the discounted mean square of the gradients and dividing the update by that value. At each evaluation, the discounted mean square is updated:

$$E\left[\mathbf{g}^2\right]_k = \rho E\left[\mathbf{g}^2\right]_{k-1} + \left(1-\rho\right)\mathbf{g}_k^2, \tag{5.19}$$

where g_k is the latest gradient estimate and ρ is a decay constant (usually just set equal to 0.95, without need of any tuning). The square operation is done element by element.

The second consideration is a simple idea to make the algorithm resemble more a second order method, despite using first order information only. The idea is that in second-order method the update is proportional to the ratio between the first and the second derivative. Consequently, the inverse of the second derivative is proportional to the ratio between the update of a second order method and the first derivative:

$$\Delta x = \frac{\frac{\partial f}{\partial x}}{\frac{\partial^2 f}{\partial x^2}} \rightarrow \frac{1}{\frac{\partial^2 f}{\partial x^2}} = \frac{\Delta x}{\frac{\partial f}{\partial x}}. \tag{5.20}$$

The last term is therefore approximated:

$$\frac{\Delta \boldsymbol{\theta}}{\frac{\partial f}{\partial \boldsymbol{\theta}}} \approx \frac{\sqrt{E\left[\Delta \boldsymbol{\theta}^2\right]_{k-1} + \epsilon}}{\sqrt{E\left[\mathbf{g}^2\right]_k + \epsilon}}, \tag{5.21}$$

where $E\left[\Delta \mathbf{x}^2\right]_{k-1}$ is computed the same way as Equation (5.19), and ϵ is a very small parameter that has the twofold goal of avoiding to have a division by zero, as well as initializing the first update. The update rule is then:

$$\Delta \boldsymbol{\theta} = -\frac{\sqrt{E\left[\Delta \boldsymbol{\theta}^2\right]_{k-1} + \epsilon}}{\sqrt{E\left[\mathbf{g}^2\right]_k + \epsilon}} \mathbf{g}_k. \tag{5.22}$$

It should be noted that for obvious recursiveness reasons, the root mean square of the updates used is that of the previous iteration.

In theory, this method is supposed to make some sort of the diagonal Hessian approximation. Nevertheless, such approximation is done using the assumption that the previous updates have been obtained using a second order method too. An obvious consequence of this fact is the large dependence on ϵ, which governs the very first update (and is also the only tunable parameter of this technique), as also shown in the test cases proposed by Zeiler [2012]. Nevertheless, the tuning is very easy to do, and in most cases $\epsilon = 10^{-6}$ is a good solution.

5.7.5 NATURAL GRADIENT

The natural gradient as an update method for neural networks was first proposed by [Amari, 1998]. This was done because the space of the neural networks (whose coordinates are its parameters) is Riemannian, and not Euclidean. Without going too much into detail, this means that the gradient does not coincide with the steepest descent direction [Amari, 2016], because the shortest distance in a Riemannian space is not a straight line. Despite not being too complex nor too long, the proof of this will not be reported. The interested reader may refer to Amari and Douglas [1998] who also provide a very simple but efficient two-dimensional example of how the Riemannian metric can speed up convergence if a paremeter space is not Euclidean.

Amari [1998] shows not only that the metric of the parameters of a neural network is Riemannian, but also that such metric coincides with the Fisher Information Matrix (FIM). The proof of both these statements is rather obscure for someone lacking some background in information geometry and statistics, and is therefore not reported. The interested reader may look up to the original paper, or to a more recent version in Amari [2016].

Going directly to the point, the natural gradient $\tilde{\nabla}_\theta$ is obtained as follows:

$$\tilde{\boldsymbol{\nabla}}_\theta = \mathbf{F}(\boldsymbol{\theta})^{-1} \boldsymbol{\nabla}_\theta, \tag{5.23}$$

where $F(\boldsymbol{\theta})$ is the FIM. It is important to notice that the metric is a function of the parameters only, and not of the inputs of the network. However, it is usually obtained by sampling, since its definition (which, also, will not be reported here) involves the expected value operator.

Amari et al. [2000] propose an adaptive method to sequentially approximate the matrix:

$$\mathbf{F}_k(\boldsymbol{\theta}_k) = (1-\epsilon)\mathbf{F}_{k-1}(\boldsymbol{\theta}_k) + \epsilon \nabla_\theta f(\mathbf{x}_{k-1}, \boldsymbol{\theta}_{k-1}) \nabla_\theta f(\mathbf{x}_{k-1}, \boldsymbol{\theta}_{k-1})^T, \tag{5.24}$$

where ϵ is a term vanishing with time. This way no sampling is needed, other than that already used for the computation of the gradient itself. The equation contradicts the fact that the matrix is a function of the parameters; however, assuming that the updates are small at each step, the error is negligible.

Always Amari et al. [2000] then show how to directly estimate the inverse of the FIM:

$$\mathbf{F}_k(\boldsymbol{\theta})^{-1} = (1+\epsilon)\mathbf{F}_{k-1}(\boldsymbol{\theta})^{-1} - \epsilon \mathbf{F}_{k-1}(\boldsymbol{\theta})^{-1} \nabla_\theta f \nabla_\theta^T f \mathbf{F}_{k-1}(\boldsymbol{\theta})^{-1}. \tag{5.25}$$

It should be noted that, according to the previously used definitions, $\nabla_\theta^T f$ is what was previously called the Jacobian, and does not depend on the expected output $\mathbf{y}_p$.

The update step of a natural gradient method is then:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha \mathbf{F}_k(\boldsymbol{\theta})^{-1} \nabla_\theta E\left(\mathbf{x}_p, \mathbf{y}_p, \boldsymbol{\theta}_k\right). \tag{5.26}$$

The natural gradient can then easily implemented in an SGD method, and is usually very robust with respect to the two tuning parameters ϵ and α.

Despite not requiring the inversion of the matrix, this method is still computationally quite demanding, in that it requires the multiplication by the inverse of the FIM, both in the parameters update and in the matrix update. Such multiplications are not demanding per se, but since they occur at each update, and SGD may require millions of updates, they may slow down the algorithm too much, especially when having large networks. Despite the algorithm would convergence in less iterations, the total time needed may happen to be larger than in simple SGD.

More sophisticated and efficient algorithms using the natural gradient are used for deep learning as well [Desjardins et al., 2015; Pascanu and Bengio, 2014; Povey et al., 2015; Roux et al., 2008]. Moreover, the natural gradient is particularly popular in actor-critic and policy gradient reinforcement learning methods [Bhatnagar et al., 2009; Grondman et al., 2012], since the main drawback of it, which is the computational cost of having to multiply the inverse of the FIM, is negligible with respect to the computational cost of the simulation of the episode. Examples of reinforcement learning algorithms making use of the natural gradient are Peters and Schaal [2008] for actor-critic and Kakade [2001] and Miyamae et al. [2010] for actor-only methods.

5.7.6 ADAGRAD

AdaGrad is one of the most popular adaptive methods for stochastic gradient descent, and was first proposed by Duchi et al. [2011]. Their goal is that of obtaining second-order methods by using the square-roots of the gradients. First they define:

$$G_k = \sum_{i=1}^{k} \nabla\theta_i \left(\nabla\theta_i\right)^T \tag{5.27}$$

which plays a fundamental role in the algorithm. The idea behind this algorithm is that of using information about the geometry of the previous updates to exploit the local curvature. It can be seen how the definition of G_k reminds, in some ways, the way that the inverse of the FIM is updated in Equation (5.25).

They then propose a first update rule for the algorithm:

$$\Delta\theta = -\eta \left(G_k\right)^{-1/2} \nabla_\theta. \tag{5.28}$$

However, as for the natural gradient, this update rule can be problematic when using large networks. Therefore, they propose the algorithm that is now known as AdaGrad:

$$\Delta\theta = -\eta \operatorname{diag}\left(G_k\right)^{-1/2} \boldsymbol{\nabla}_\theta . k \tag{5.29}$$

At this point, the matrix G_k can be dropped, since its diagonal is the sum of all the gradients at each epoch, element by element. It is, in some ways, equal to the denominator of AdaDelta, but without discount. This means that the value continuously increases with time, and ends up providing the algorithm with a decaying rate, as in SGD. Here, the decay is specific per each parameter, which is a great advantage when using deep networks. Moreover, the algorithm is designed such that rare, but important features are given the proper importance: in fact, when a parameter that has usually very small gradients, turns out to have a rather large gradient, it ends up having a much larger update than it would have if also the previous gradients were large. Instead, parameters whose updates are often large, tend to have smaller updates. AdaGrad has the advantage with respect to AdaDelta of having a learning rate that automatically annihilates; however, it has the drawback of having the parameter η to be tuned (AdaDelta has ϵ to be tuned instead, but it is usually just let equal to 10^{-6}).

5.7.7 LEVENBERG-MARQUARDT

Levenberg-Marquardt (LM) is a common method used in nonlinear fitting problems, that has very successfully been applied to machine learning. A good overview of the method is given by Yu and Wilamowski [2011]. The main concept is the following: the approximated Hessian of the sum square error (over the entire data-set, or at least a large portion of it) with respect to the parameters, is computed, and then used to iteratively solve the optimization problem, using Newton's method.

The elements of the Hessian can be obtained as follows:

$$h_{i,j} = \frac{\partial^2 E}{\partial\theta_i \partial\theta_j} = \frac{\frac{1}{2}\partial^2 \sum_{p=1}^{n_p}\sum_{m=1}^{n_m} e_{p,m}^2}{\partial\theta_i \partial\theta_j} = \sum_{p=1}^{n_p}\sum_{m=1}^{n_m}\left(\frac{\partial e_{p,m}}{\partial\theta_i}\frac{\partial e_{p,m}}{\partial\theta_j} + \frac{\partial^2 e_{p,m}}{\partial\theta_i\partial\theta_j} e_{p,m}\right), \tag{5.30}$$

where n_m is the length of the output vector. Since Newton's method assumes that $e_{p,m}$ is close to zero, one can neglect the computation of the second term of the sum, which is the most demanding one. Therefore, the entire Hessian can be approximated by using first-order information only:

$$\mathbf{H}_k \approx \mathbf{J}_k^T \mathbf{J}_k \tag{5.31}$$

Therefore, the update $\Delta\boldsymbol{\theta}_k$ can be computed as follows:

$$\Delta\boldsymbol{\theta}_k = \left(\mathbf{J}_k^T \mathbf{J}_k\right)^{-1} \mathbf{J}_k \mathbf{e}_k. \tag{5.32}$$

The latter is the general update rule for a Gauss-Newton method, in which the Hessian has been approximated (otherwise, it would be the standard Newton method). In the LM algorithm, there is a small modification, to ensure that the Hessian is invertible:

$$\mathbf{H} \approx \mathbf{J}^T \mathbf{J} + \mu \mathbf{I}, \tag{5.33}$$

where μ is a dynamic combination coefficient. The goal of the combination coefficient is that of raising all the eigenvalues of the same value (μ, indeed), and therefore diminishing the conditioning number of the Hessian. From a more intuitive point of view, a large μ makes the method resemble more a steepest descent gradient method: this may be particularly useful in case of very small curvatures, where a second-order method tends to induce too large steps. To understand the reason of this, it can be seen that with a very large μ, the method reduces to:

$$\Delta\boldsymbol{\theta}_k = \left(\mu\mathbf{I}\right)^{-1} \mathbf{J}_k \mathbf{e}_k = \frac{1}{\mu}\mathbf{J}_k \mathbf{e}_k. \tag{5.34}$$

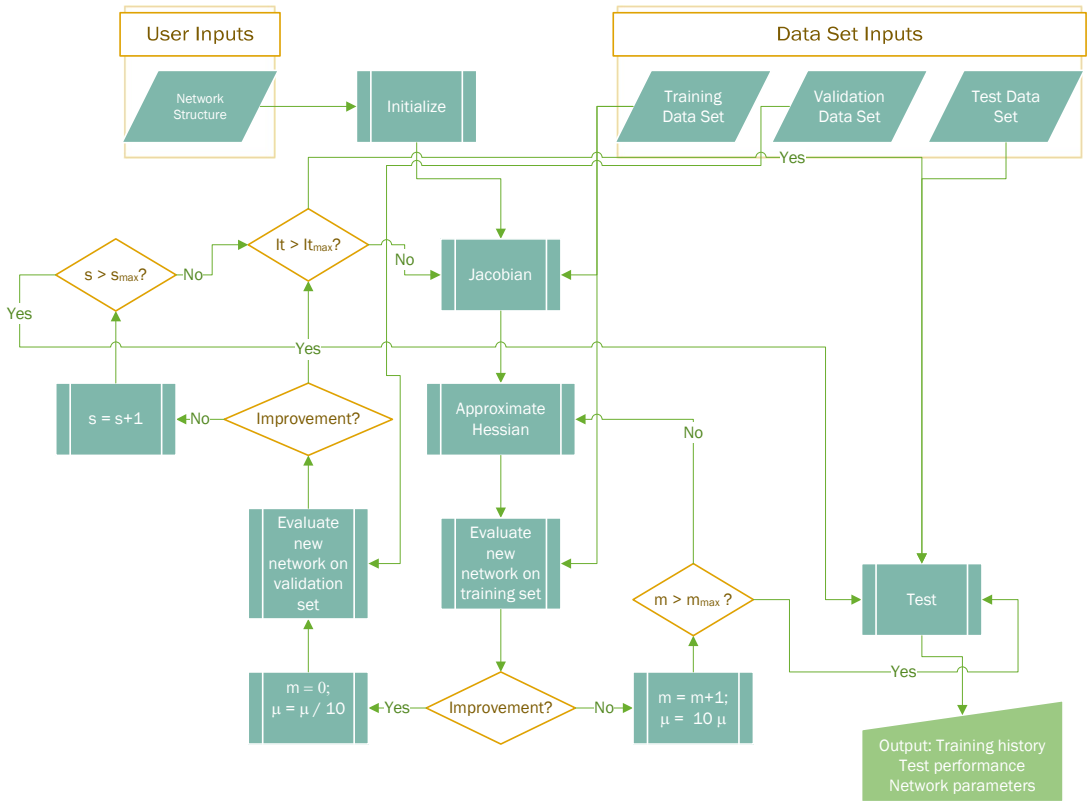


Figure 5.5: Flow-chart for the LM training algorithm.

Therefore, the larger μ, the more the method resembles a first order method, with update rate smaller at every step. In most cases, μ is a dynamic value that is updated at each iteration. The most traditional way of updating μ consists of using a certain value, test the new error after the update, and either accept the result (if the new error is smaller than the previous one) and decrease the value, or increase the value and check again. For these reasons, the LM algorithm is also called "damped least squares" method.

However, other possibilities exist. As an example, Suratgar et al. [2007] propose to use $\mu = 0.01\mathbf{e}^T\mathbf{e}$. This way, μ is larger when the errors are larger (when the assumption done to derive the Gauss-Newton method is stronger), and adapts itself without the need of a line search; however, there is no assurance that an improvement occurs at each update step.

Another modification, which was implemented by Marquardt, is that of using the diagonal of the approximated Hessian rather than the identity matrix to damp the method [Transtrum and Sethna, 2012]:

$$\mathbf{H} \approx \mathbf{J}^T\mathbf{J} + \mu \text{diag}\left(\mathbf{J}^T\mathbf{J}\right). \tag{5.35}$$

With this implementation, for large μ the algorithm resembles more the first step of the AdaDelta algorithm, and should therefore be more efficient than if using the identity matrix.

The flow-chart in Figure 5.5 shows how the algorithm is designed. The correct value of μ is esti-

mated by trial and error, and increased whenever the solution is rejected, meaning that it causes an increase in the sum of the error on the training set. If a solution is accepted, the algorithm moves to the next iteration, and μ is decreased. Usual values are multiplying or dividing by 10. In this case, it was found that having a multiplication by 2 and a division by 3 turned out very efficient. There is no upper no lower bound to the value of μ. If the maximum number of inner iterations is reached, the algorithm starts a new outer iteration with the new value of μ. When the inner iterations are over (either because of having reached the maximum number, or because the solution has been accepted) the algorithms computes the error on the validation set. If there has not been an improvement in the last $s_{\max}$ iterations, the training stops. It is important that the algorithm saves the network that performed best in the validation set. The algorithm may stop also if the maximum number of iterations is reached.

5.7.8 LEARNING ALGORITHMS VERIFICATIONS

All the algorithms mentioned in this section have been developed, verified, and tested during this research. Table 5.1 provides a useful summary of the performance of all the training methods used, and a comparison with some methods included in the MatLab Toolbox. It does not include, however, any results concerning the simple SGD nor the natural gradient. The first is not included because it simply consists of backpropagation; the second because it has eventually not been used in any of the trainings in this research.

All the algorithms have been tested on a network with 3 hidden layers with 10-8-5 neurons, all with hyperbolic tangent as activation function, and linear output functions. The training ratio was 70 %, and the regularization parameter $\lambda = 10^{-2}$. Between brackets, is the number of training dones for a specific version of the algorithm. The MatLab routines are called with their function name. The final peformance is always evaluated on the full set (training, validation and testing set together). As a last note, when testing the algorithms, the network used is the one from the epoch that scored the best in the validation dataset. This is to avoid that an overfitted network is used in the final testing.

The table shows that, in general, the codes here developed achieve better results than those of the Matlab Toolbox. They do so, though, with larger computational requirements. The training with the developed algorithms takes usually 2 to many times longer than the MatLab algorithms. After a further investigation, it was noticed that such difference occurs because of how the networks have been coded. The algorithms implemented in this research need averagely as many, or less, function evaluations than those of the MatLab Toolbox, while achieving better results. In addition, the SGD methods provided by MatLab perform particularly poorly. This is very important, since they are the only methods that can be used to train deep networks. Hence, this subsection justifies the need of implementing algorithms for both backpropagation as well as for supervised learning.

5.8 REINFORCEMENT LEARNING

Reinforcement learning is a branch of artificial intelligence that is directly inspired by how learning occurs in intelligent beings. In a formal way, RL consists of mapping to each state the action that maximizes the expectation of future rewards [Engelbrecht, 2007; Sutton and Barto, 2012]. Informally, it can be defined as learning by interaction, or learning by exploration and exploitation. Exploration refers to the act of attempting new actions, whereas exploitation consists of making use of what was previously learned.

5.8.1 ELEMENTS OF RL

A few general concepts are necessary before discussing the RL framework [Engelbrecht, 2007]:

- **Agent**: is the learner, the decision maker.

Table 5.1: Algorithms' performance on the building energy dataset from MatLab.

Algorithm/ Options	**End conditions**	**Time**	**RMS**	**Standard deviation of RMS**
trainlm (100) MatLab default	15 fails / $\mu = 10^{10}$	8 s	0.0396	0.0045
traingdx (100) MatLab default	15 fails	10 s	0.0652	0.0395
traingda (100) MatLab default	15 fails	4 s	0.1017	0.0287
LM (10) $\lambda_0 = 1$	5 fails / 30 epochs	71 s	0.0368	0.0014
LM (10) $\lambda_0 = 0.01$	5 fails / 30 epochs	76 s	0.0464	0.0204
LM (10) $\lambda_0 = 100$	5 fails / 30 epochs	72 s	0.0370	0.0013
LM (10) $\lambda_0 = 1$	5 fails / 10 epochs	25 s	0.0414	0.0028
LM (10) $\lambda_0 = 1$	5 fails	151 s	0.0361	0.0013
AdaDelta (100) $\epsilon = 10^{-6}$, $N = 10$, $\gamma = .9$	2 000 batches	26 s	0.0433	0.0018
AdaDelta (10) $\epsilon = 10^{-5}$, $N = 10$, $\gamma = .9$	2 000 batches	33 s	0.0462	0.0025
AdaDelta (10) $\epsilon = 10^{-6}$, $N = 10$, $\gamma = .99$	2 000 batches	31 s	0.0440	0.0024
AdaDelta (10) $\epsilon = 10^{-6}$, $N = 100$, $\gamma = .9$	5 000 batches	606 s	0.0358	0.0013
AdaDelta (10) $\epsilon = 10^{-6}$, $N = 1$, $\gamma = .9$	10 000 batches	41 s	0.0481	0.0027
AdaDelta (10) $\epsilon = 10^{-7}$, $N = 10$, $\gamma = .9$	2 000 batches	33 s	0.0501	0.0034
AdaDelta (10) $\epsilon = 10^{-7}$, $N = 10$, $\gamma = .9$	10 000 batches	184 s	0.0400	0.0004
AdaGrad (100) $\eta = 0.1$, $N = 10$	1 000 batches	15 s	0.0441	0.0022
AdaGrad (100) $\eta = 0.1$, $N = 10$	2 000 batches	31 s	0.0414	0.0014
AdaGrad (10) $\eta = 0.1$, $N = 10$	10 000 batches	145 s	0.0384	0.0009
AdaGrad (10) $\eta = 0.1$, $N = 100$	1 000 batches	141 s	0.0395	0.0005
AdaGrad (10) $\eta = 0.1$, $N = 100$	5 000 batches	531 s	0.0356	0.0011

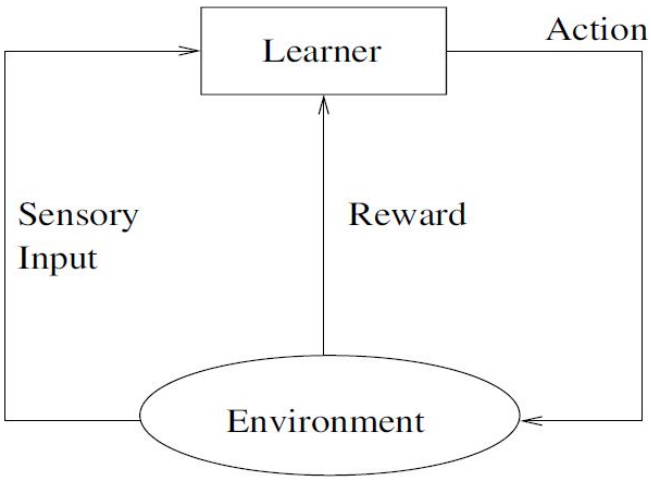


Figure 5.6: Reinforcement learning problem [Engelbrecht, 2007].

- **Environment**: is whatever cannot be modified by the agent. It is generally defined by the transition function (for a discrete or a stochastic system), or the dynamics equations (for a continuous deterministic environment). It is, in the case of this research, equivalent to the model.

- **State** $s \in \mathcal{S}$: is the state in which the agent is. Since a full description of the state is often impossible, the state is limited to the input that the agent receives.

- **Action** $a \in \mathcal{A}$: is the decision made by the agent. According to the dynamics, or the transition function of the problem, the couplet s, a will evolve the state of the agent to the future state, either deterministically or according to the probability distribution $d(s\prime|s,a)$, where $s\prime$ is the state of the agent in the future time-step.

- **Policy** π: is the law, or rule, or function, or probability distribution, according to which an action a is taken depending on the current state s.

- **Immediate reward** $r(s,a,s\prime)$: is the scalar value of how good the latest action has been.

- **Value function** $V_\pi(s)$: is the value of all the future rewards that are expected to be obtained when in state s following policy π.

- **Action-value function** $Q_\pi(s,a)$: is the value of all the future rewards that are expected to be obtained when in state s, taking action a, and then following policy π.

The environment used in RL problems can usually be modeled as an MDP [Sutton and Barto, 2012], in which the state consequent to an action is a probability distribution and not deterministic. This can include aerocapture problems, where the state (as known by the guidance logic, thus, the input for the guidance), together with the action, cannot deterministically define a future state, because of perturbations and lack of information. In a discrete-time Markovian decision process, the future state s' is indeed the result of the probability distribution $d(s\prime|s,a)$ previously mentioned.

5.8.2 VALUE FUNCTIONS AND BOOTSTRAPPING

At this point of the description, discrete states and actions are considered.

The core of reinforcement learning lies in the correct evaluation of the value function. It is clear from its definition that the evaluation of the value function for any state implies having an unbiased expectation of how the state will evolve in the future for an infinite amount of time.In mathematical terms, a value function is defined as (for episodic tasks only, since this is the concern of this research) [Sutton and Barto, 2012]:

$$V(\mathbf{x}) = E\left[\sum_{t=1}^{n_t} \gamma^t r(t)\right] \tag{5.36}$$

where γ is a discount factor, between 0 and 1.

An optimal value function $V^\star$ is the value function referred to the optimal policy $\pi^\star$, the policy for which the corresponding value function is higher than any other at any state. For an optimal value function:

$$V^\star(s) \geq V_\pi(s), \qquad \text{for any } \pi, \text{ and for all } s \in \mathcal{S} \tag{5.37}$$

The same condition holds for the optimal action-value function $Q^\star(s,a)$. These imply, after a short derivation, the Bellman optimality equations [Bellman, 1957]:

$$V^\star(s) = \max_{a\in\mathcal{A}} \sum_{s\prime}^{p} (s\prime|s,a)\left[r(s,a,s') + \gamma V^\star(s\prime)\right] \tag{5.38}$$

$$Q^\star(s,a) = \sum_{s\prime} p(s\prime|s,a)\left[r(s,a,s') + \gamma \max_{a\in\mathcal{A}} Q^\star(s\prime,a)\right] \tag{5.39}$$

Both equations express the concept of bootstrapping, which consists of evaluating a value function as the sum of its immediate reward plus the discounted value of the value function of the following state [Sutton and Barto, 2012].

From this the policy improvement theorem [Sutton and Barto, 2012] can be derived, which is the foundation of dynamic programming. It states that, if, for all $s \in \mathcal{S}$:

$$Q_\pi(s,\pi\prime(s)) \geq V_\pi(s) \tag{5.40}$$

then $\pi\prime$ is as good, or better, than π. By applying this criterion, it is possible to iteratively optimize the policy and its corresponding value function. This should be done for any state, and for any action possible, iteratively, both because of the randomness of the state transition, and because every update is based on a previously non-optimal value function of the following state. After virtually an infinite number of iteration, convergence occurs. It is clear that this method becomes infeasible for problems with large state-space and action-space; moreover, it is proved that its convergence time increases polynomially with the size of the state-space, and exponentially with the number of dimensions. The latter is the so-called "*curse of dimensionality*" [Bellman, 1957].

5.8.3 Q-LEARNING

Q-learning partly solves the previous problem [Watkins, 1989]. It consists of learning from a real situation (or simulation). Learning starts from a certain state, and then goes on. By so doing, the only states to be updated would be those that occur during a real situation, largely saving computational time with respect to dynamic programming. The update rule for this kind of learning is [Sutton and Barto, 2012]:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r(s,a) + \gamma \max_{a\prime} Q(s\prime,a\prime) - Q(s,a)\right] \tag{5.41}$$

where α is a constant smaller than 1, which is needed because of the non-deterministic properties of the environment and because of the non optimal value of $Q(s\prime,a\prime)$, and the expression in the parenthesis is referred to as the δ:

$$\delta = \left[r(s,a) + \gamma \max_{a\prime} Q(s\prime,a\prime) - Q(s,a)\right] \tag{5.42}$$

The value of δ is obtained by substituting the values obtained in Bellman's optimality equation.

5.8.4 ELIGIBILITY TRACES

When optimizing, the ΔV of an aerocapture, all of the rewards are given at the final step, once the ΔV can be determined. Thus, an update rule for the only previous state may slow down the convergence process indefinitely. This is solved using Q(λ)-learning, which makes use of eligibility traces.

An eligibility trace Z is initialized at the beginning of each episode by setting $Z(s) = 0$ for all $s \in \mathcal{S}$. Then, at each time-step:

$$Z_t(s) = \begin{cases} \gamma\lambda Z_{t-1}(s) & \text{if} s \neq s_t \\ 1 & \text{if} s = s_t \end{cases} \tag{5.43}$$

The update is then done for all $s \in \mathcal{S}$, using the δ as computed for the Q-learning:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\delta Z(s) \tag{5.44}$$

By so doing, and properly tuning the value for λ (its optimal value depends on the problem, but in most examples shown in Sutton and Barto [2012] the best value is between 0.8 and 0.9), one can be certain that part of the value function at the final state is propagated backwards to the initial state as well.

However, if one is looking for the optimal action-value function, a problem occurs. It is in fact possible that a certain action a appears to be less optimal than other actions just because the corresponding state-action function considers the not yet optimal policy π. However, it is possible that that very action a would be optimal if the action-value function were the optimal one. This can leave the state consequent to that action to never be explored, and thus the optimal action-value function found would not be a global optimum. To solve this, an element of randomness in the decision process is added. A policy is called greedy if it always tries to maximize the value function:

$$\pi\prime(s) = \arg\max_{a \in \mathcal{A}} Q_\pi(s,a) \tag{5.45}$$

It is called ϵ-greedy if instead it takes a non-optimal decision with probability ϵ. The choice of ϵ is fundamental to the convergence of the method, and is usually decreased in time during learning [Sutton and Barto, 2012].

When taking a non-optimal action at time t_k in a Q(λ) learning method, the effects of such choice propagate backwards because of the eligibility trace. This is something undesired, since in this case the algorithm is just exploring, and thus the consequences of taking a non-optimal action at t_k would negatively affect the states previous to that non-optimal decision. At this point, two different philosophies are possible: using Watkins' Q(λ) [Watkins, 1989], one deletes the trace once the non-optimal action is taken; using Peng's Q(λ) [Peng, 1993; Peng and Williams, 1996] instead, a new simulation starts where the non-optimal action is taken, but the previous one is also continued, such that the update of the final state can reach the initial state.

Watkins' Q(λ) is clearly less efficient than Peng's Q(λ), since the propagation backwards in time is interrupted, making the eligibility traces less efficient. In the extreme case, assuming $\epsilon = 1$, Watkins' Q(λ) would be reduce to a simple Q-learning method, which, as it was mentioned, would be very inefficient for the problem of aerocapture. Peng's Q(λ), on the other hand, although more efficient, highly increases the complexity of the structure of the algorithm: if, as an example, one had a simulation lasting 100 time-steps, and an $\epsilon = 0.1$, the final number of simulations consequent to that single one would easily be about 2^{10}, since, on average, every simulation would split into two every ten time-steps. This would not be a problem from a computational point of view, since each of those simulations would contribute to optimizing the action-value function, but from the point of view of the architecture of the software.

5.9 CONTINUOUS RL

A final complication occurs because of the continuity of the state-action space. So far, only discrete, or tabular, states were considered. To make RL possible in continuous state-action spaces, parametric function approximators, such as the previously discussed FFNNs, have to be used, for both the value and the policy functions.

An algorithm such as Q(λ), when made continuous with the aid of a parametrized function, is a critic-only algorithm. The policy does not play a role in the optimization process, since all the decisions made are either optimal with respect to the action-value function or, if using an ϵ-greedy policy, random.

These kind of algorithms suffer of the fact that either the action spectrum has to be discrete, either an optimization problem has to be used at every step. Moreover, the approximate action-value function would be biased for most of the cases, and no guarantee of convergence is available [Grondman et al., 2012]. However, they have the advantage that the update occurs at each time-step.

The opposite to those are actor-only methods (where the name actor stands for the agent), in which the agent is a parametrized function, and no explicit value function is used. They are divided into two categories: those searching in action space, and those searching in parameter space. Actor-only methods that search in action space are particularly popular for tasks with no delayed rewards. That is because they do not make use of a value function, which is the only means of keeping delayed rewards into account, without running entire episodes. Examples of these are the REINFORCE algorithms [Williams, 1992], in which a policy is reinforced whenever the reward is larger than a certain baseline. Without going into details, the baseline is the average reward encountered up to that moment. Always Williams [1992] gives an example of how the same method can be used for episodic tasks with delayed rewards, using "unfolding-in-time" mapping.

Moreover, REINFORCE algorithms converge particularly slowly because of the large noise in the gradients, due to the use of stochastic policies. Randomness is extremely important in that it provides exploration, and should not be removed. The way the algorithm works is indeed by taking a stochastic action whose probability density distribution is defined by the current policy parameters: if the action gives a larger reward than the baseline, the action is reinforced.

The main advantage of these algorithms is indeed that they work in the action space, instead of in the parameters space: this means that the reinforcement can be provided by back-propagation, which is extremely efficient.

However, the presence of a baseline cannot work with the problem of this research, since the maximum obtainable reward depends on the initial conditions and perturbations, and differs very much one case from the other. Therefore, a policy tested with unfavorable initial conditions will always be negatively reinforced, even if it obtained the theoretically maximum achievable rewards for those initial conditions.

Policy gradient methods that work in the policy parameter space can be used for episodic tasks instead, if they are updated only at the end of an episode, or of a Monte Carlo run of episodes, in a way that maximizes the total rewards of the episode(s). In some ways, the evolutionary strategies used by Gelly and Vernis [2009] could be included into this category. Policy gradient methods that search in parameter space have the disadvantage that they have to deal with a much higher dimensionality, since the parameter space can easily have 100, or 1,000 dimensions. However, they can make use of global optimizers, and get rid of many problems that are specific to RL. A particularly efficient method is PGPE, developed by Sehnke et al. [2008]. The super-symmetric version of it is the algorithm that will be used in the RL part of this research, for reasons that will be explained in the next subsection.

The problem of the baseline could have also been solved by actor-critic methods. With these, it would be possible to update the policy (and, in this case, also the value) networks at each time-step, while still keeping delayed reward into account. However, they require linear approximators to be

able to reach global optima. Moreover, they are very difficult to tune. An example of this application in the aerospace field is provided by van Kampen et al. [2006], who showed that it can be used to control the longitudinal dynamics of an F16.

In the case of this research, an additional drawback of using an actor-critic method is the fact that the rewards are a function of the final conditions only: thus, the method would have largely relied on bootstrapping, and this could have made the learning less robust.

5.10 POLICY GRADIENT WITH PARAMETERS EXPLORATION

Sehnke et al. [2008] developed a method called PGPE for episodic tasks, with delayed rewards, to tackle the first problem of actor-only methods, concerning the variance of the gradient. They propose to move the exploration to the parameters, instead of to the action. By so doing, a policy is perturbed, and then is deterministic during the entire episode. Then, by sampling, the gradient with respect to the policy parameters is estimated. The method still uses a baseline, and is therefore not suitable for this research. However, the problem is first mitigated by Sehnke et al. [2010], with the SyS PGPE, and then completely removed by Sehnke [2013], who introduced the Super SyS PGPE, which is entirely baseline-free.

The main reason why Super SySPGPE is used is indeed the fact that it is baseline-free, it is a global optimizer, and it has proven to be working very well with high-dimensional policies such as an FFNN.

The method is here reported. Any derivation is skipped, and can be found in the original papers. The main concept behind SySPGPE is that of having a random perturbation $\boldsymbol{\epsilon}$ in the parameters, and evaluate the episode for that perturbed policy. Same is done for a policy that is perturbed in the exactly opposite direction ($-\boldsymbol{\epsilon}$). Then, the mean value of the parameters is shifted in the direction of the one that scored best, in a way proportional to how much better the perturbation was. It is important to notice that the current policy is never evaluated. Moreover, what is estimated this way, is more of a global trend rather than a gradient, since the perturbation is always very far from being infinitesimal. For this reason, the method is also very suitable for global optimization of high-dimensional problems, as it will be shown later.

The perturbation is drawn from a Gaussian distribution, whose mean $\boldsymbol{\mu}$ and standard deviation $\boldsymbol{\sigma}$ are different for each parameter. The update rule is the following, normalized according to Sehnke and Zhao [2015]:

$$\Delta\boldsymbol{\mu} = \alpha_\mu \frac{\boldsymbol{\epsilon}\left(r^+ - r^-\right)}{2m - r^+ - r^-}, \tag{5.46}$$

where r^+ is the reward obtained by the policy perturbed by $\boldsymbol{\epsilon}$, r^- is the reward obtained the policy perturbed by $-\boldsymbol{\epsilon}$, and m is the maximum obtainable reward Alternatively, without normalization:

$$\Delta\boldsymbol{\mu} = \alpha_\mu \frac{\boldsymbol{\epsilon}\left(r^+ - r^-\right)}{2}. \tag{5.47}$$

The variance with which the perturbations are drawn is also updated. To do so, it is necessary, for each parameter μ_i, to obtain a perturbation ϵ_i^* that is the mirror of the original perturbation ϵ_i. In other words, it is necessary to generate a function that, if ϵ_i belongs to the central 5% of a Gaussian distributions, finds the corresponding ϵ_i^* belonging to the outer 5%. A closed-form function that does this exactly is unknown, and Sehnke [2013] proposes a function that is a very good approxima-

tion of it. Without giving any derivation:

$$\phi_i = 0.67449\,\sigma_i, \tag{5.48}$$

$$a_i = \frac{\phi_i - \|\epsilon_i\|}{\phi_i}, \tag{5.49}$$

$$\epsilon_i^* = \frac{\epsilon_i}{\|\epsilon_i\|} \begin{cases} e^{c_1 \frac{\|a_i\|^3 - \|a_i\|}{\log(\|a_i\|)} + c_2 \|a_i\|} & \text{if} \quad a_i \leq 0 \\ \frac{e^{a_i}}{\left(1-a_i^3\right)^{c_3 a_i}} & \text{otherwise,} \end{cases} \tag{5.50}$$

where $c_1 = -0.06655$, $c_2 = -0.9706$, and $c_3 = 0.124$. Such distribution has a standard deviation of 1.002 times that of the original one, and a slightly different shape. With this, another pair of samples is generated, using $\boldsymbol{\epsilon}^*$ and $-\boldsymbol{\epsilon}^*$. The mean reward of these two is r^{--}, whereas the mean reward of the previous two is r^{++}. Also, these two samples can participate in Equation (5.47) if one substitutes $\boldsymbol{\epsilon}$ with $\left(\boldsymbol{\epsilon}^* + \boldsymbol{\epsilon}\right)/2$, and computes r^+ as the mean of the rewards obtained with $\boldsymbol{\epsilon}$ and $\boldsymbol{\epsilon}^*$ (and r^- as the mean of the rewards obtained with $-\boldsymbol{\epsilon}$ and $-\boldsymbol{\epsilon}^*$). The standard deviation can then be updated as:

$$\epsilon^\tau = \begin{cases} \epsilon & \text{if} \quad r^{++} - r^{--} \leq 0 \\ \epsilon^* & \text{otherwise} \end{cases} \tag{5.51}$$

$$\Delta\sigma_i = \alpha_\sigma \frac{\frac{\epsilon_i^\tau 2 - \sigma_i^2}{\sigma_i} \left\| r^{++} - r^{--} \right\|}{2}. \tag{5.52}$$

This way, the standard deviation of a certain parameter decreases if the highest rewards are obtained when such parameters is perturbed by more than one median deviation (ϕ_i), and increases otherwise. Also in this case, normalization is possible, dividing by $\left(2m - r^{++} - r^{--}\right)$.

5.10.1 VERIFICATION

The algorithm has been reproduced for this research. The verification consisted of two parts: first, checking whether the mirrored function was properly written. This was done by generating one million of random samples, and mirroring them, and then checking the distribution of the samples. Then, the software was validated by optimizing the Rastrigin function, which is provided by Matlab.

The method turned out to reach the global optimum almost always for a 10-dimensional Rastrigin, and achieved local optima extremely close to the global one for both the 100- and the 1,000-dimensional Rastrigin. A convergence plot of the latter case for an average of 10 attempts is shown in Figure 5.7. The convergence curve is extremely similar to the one obtained by Sehnke [2013] in their validation. The same metaparameters were used (and the search space was bounded between -10 and 10 in every dimension). To give an idea of the complexity of the problem, an sequential quadratic programming (SQP) method stops in the closes local minimum, and also many global optimizers such as simulated annealing or evolutionary algorithms often have a hard time in solving the Rastrigin problem for high dimensionalities.

Figure 5.8 shows instead the convergence for the 100-dimensional Rastrigin, with an initial variance very large ($\sigma_0 = 20$). The convergence curve is quite curious, and seems to be followed by every one of the different 100 optimization attempts. This sort of curve will be encountered again in this research, and will be interpreted as an indicator of a too large σ_0. However, it does not cause excessive problems, since the convergence curve for the same problem (not reported here) with a smaller initial variance ($\sigma_0 = 2$), converges about 55,000 iterations, instead of 65,000. It would be certainly of interest to investigate whether this feature can be exploited in to make the optimization faster.

In conclusion, the algorithm, meant as a global optimizer, is verified. However, the advantage of this method is that it also uses some sort of gradient information. For this reason, it can be used in a sequential way in RL problems. More about this will be shown in Section 6.6.

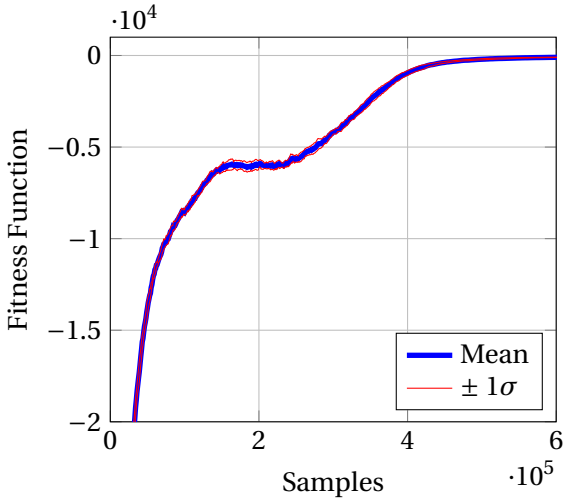


Figure 5.7: Average and standard deviation of the convergence curve of the PGPE for the 1,000-dimensions Rastrigin. Statistics from 10 different optimizations. $\sigma_0 = 2$, $\alpha_\mu = 1$, $\alpha_\sigma = 0.01$.

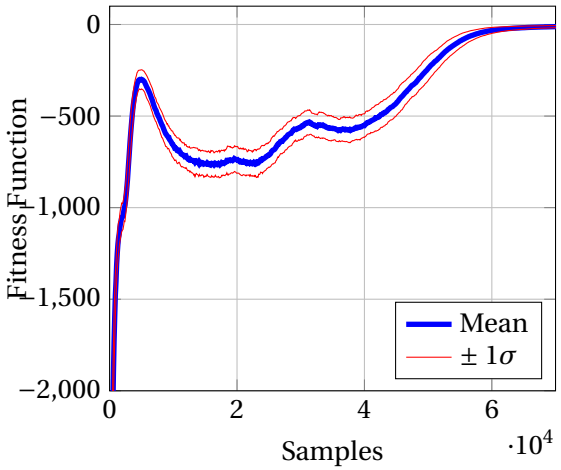


Figure 5.8: Average and standard deviation of the convergence curve of the PGPE for the 100-dimensions Rastrigin, with high initial variance. Statistics from 100 different optimizations. $\sigma_0 = 20$, $\alpha_\mu = 1$, $\alpha_\sigma = 0.01$.

5.11 CONCLUSIONS

In this chapter the the basics of machine learning and reinforcement learning have been described. It was explained in Section 5.1 that both are very attractive as a guidance solution for aerocapture. Specifically, through the first one, it is theoretically possible to obtain a neural network that should be able to give the approximation of the solution of an optimal control problem in real time.

However, optimal control problems are suited for deterministic cases. In aerocapture, the optimal solution for a deterministic environment is very different from the optimal solution for a stochastic environment, for reasons that were shown in Subsection 4.3.1. Hence, RL seems like an even better solution. RL has many branches, and each branch is divided in a variety of algorithms and methods. Among all, Super SySPGPE was chosen, given the fact that it can update the network after each episode (instead of after a Monte Carlo run of episodes, like evolutionary algorithms), has global optimization properties, and does not need a baseline.

6

SOFTWARE DESIGN, VERIFICATION AND VALIDATION

All the computational tools needed for this research have been developed, and used, on MatLab.

The software developed includes many different tools, listed below:

1. a trajectory simulator;
2. an NPC guidance;
3. a trajectory solver;
4. a machine learning architecture, including all the algorithms that were tested in Chapter 5;
5. a reinforcement learning architecture.

The design, verification and validation of all the tools that have been developed throughout this research is reported in this chapter. Before that, a short discussion about the already available software is carried out.

6.1 AVAILABLE SOFTWARE

There are a few software already available that will be used during the research. Those only concern neural network simulation and training, numerical integration and the US76 Atmosphere model. The packages concerning neural network training have only been used for verification in Subsection 5.7.8. Also there, motivation for not using the available MatLab Neural Network Toolbox was given. Thus, it is possible to start with the integrators.

Integrators. MATLAB has already implemented many routines that can numerically integrate using variable time-step methods, like **ode23**, **ode45** and **ode113**, which are respectively, Runge-Kutta (RK)23, RK45 (Dormand-Prince), and variable time-step, variable order Adams-Bashforth-Moulton of orders 1 to 12. The latter is a multi-step solver, and therefore needs to be initialized using a different method, which is not specified. Fixed steps solvers are also available, all of the RK family, from the first order to the fifth.

US76 Atmosphere. Rather than a tool, this is just a function. The one use is not the MatLab built-in function, which is only valid up to little higher than 84 km altitude, but the "Complete 1976 Standard Atmosphere" uploaded on the MatLab File Exchange by Brent Lewis in 2007. Some of the output of the function were compared with the table in [NASA, 1976], and found that the software was correct.

6.2 SIMULATOR ARCHITECTURE

In this section, the high level architecture of the simulator is described, followed by the operations, and the external forces.

6.2.1 HIGH-LEVEL ARCHITECTURE

The simulator is responsible to numerically reproduce the flown trajectories. The inputs it receives (not including the various databases being used), are the initial conditions, two different arrays and a scalar number specifying the perturbations, and a guidance scheme. The simulator also includes a simple attitude controller/planner, but since this research is not focused on attitude control, it is not changeable, and therefore it is not considered as an input. The outputs are the control and state histories of the vehicle, as well as the encountered density time series. The reason why the latter is also included as an output is because the atmospheric density perturbation model can be quite complex, and therefore the density cannot be obtained easily from the histories and the inputs. An additional vector of data about the final state is outputted, which gives information about the final planar ΔV, the ΔV including the needed inclination change, and the final apoapsis. The state history is given in spherical, relative coordinates. If desired, it is possible to transform that set of data into any other needed. Moreover, using the density history and the spherical relative state, it is possible to directly obtain the load factor history (together with the perturbations data), as well as the heat-flux history (both cold and hot wall), from which the heat-load history can be integrated. The choice of saving only one set of variable, and of not including heat-flux nor heat-load history, is done to reduce the required memory.

6.2.2 OPERATIONS

The simulator starts propagating the trajectory according to all the pre-set perturbations (in this phase, with the exception of density small-scale perturbations) until the minimum drag is reached, always set to 0.5 g. After that, the guided phase begins. The only different thing is that now, every one second, the commanded bank-angle is computed, and the consequent bank-angle trajectory is planned. Such a phase ends if either the drag goes back to be below 0.5 g, or if the altitude goes below the sea level, or, even, if the descent becomes steeper than γ =-70°. The latter condition is implemented for two reasons: first, the vehicle would never recover from such a steep descent, and therefore the aerocapture can be considered not successful, and second, a flight-path angle of -90° would cause the heading angle to be not defined. Alternatively, in a few cases, the drag condition for the guidance is substituted by the 100 km altitude line, above which the guidance is shut down (in any case, the trigger condition stays 0.5 g). The last phase of the simulation is then concluded either if apoapsis is reached (true anomaly θ = 180°), or if the flight is almost vertical, as well as if the velocity goes below 0.3 V_c, or the altitude goes below 0 km. In case the orbit is still hyperbolic, integration would end 3000 s after the end of the closed-loop phase. Such condition holds also in the exoatmospheric phase, even though one should carefully select proper initial conditions before launching the software. In case the target apoapsis is such that half orbital period at that altitude is much larger than 3000 s, the duration can be changed. This process can be seen in Figure 6.1.

These three phases are labeled, respectively, "exoatmospheric phase", "closed-loop phase", and "exit" phase. The labels' names are not completely truthful to the essence of the phases themselves, since in the exoatmospheric phase the aerodynamic force is still included, and the exit phase might happen to be the "before-crash" phase. Table 6.1 summarizes all the characteristics of each phase.

All the forces and perturbations (if active) are included in the closed-loop phase. In the other phases, the only perturbations not included are the small-scale density perturbations. As a final remark, one might debate the need of including all the forces in both the exoatmospheric and exit phase. The reason is twofold: first, Keplerian orbits could not be used, because of the presence of the J_2 gravity term, which becomes important especially in those phases. Second, during those

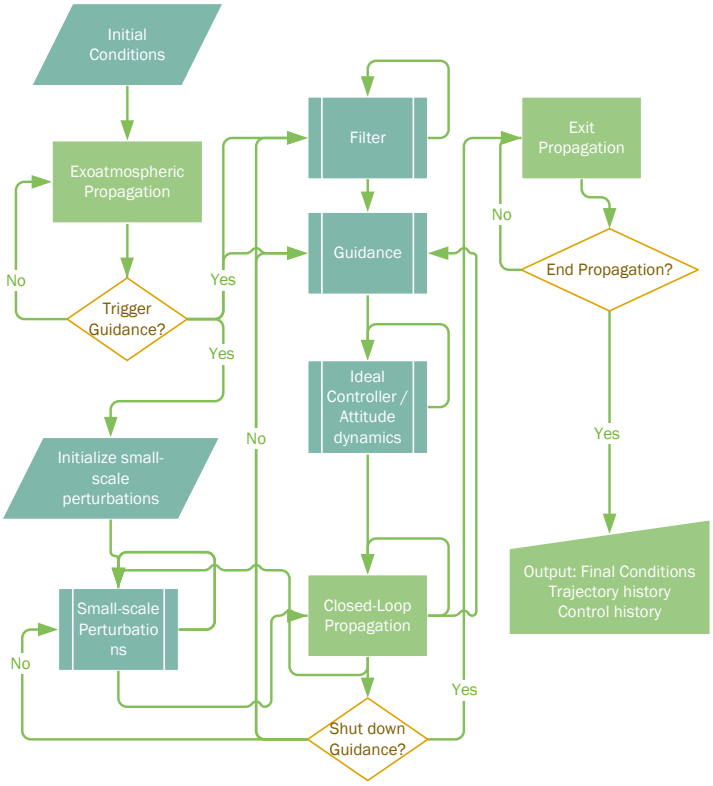


Figure 6.1: Flow chart for the high-level architecture.

phases the dynamics of the vehicle are very slow, and also, since no closed-loop is active, adaptive time-steps integrators can be used: these two factors, combined, cause the two phases to be negligible from a point of view of computational demands with respect to the closed-loop phase. The small-scale perturbations cannot be included in the exit phase nor in the exoatmospheric, because not compatible with adaptive step-size integrators. However, the loss is negligible, as previously explained.

6.2.3 EXTERNAL FORCES

The external forces play a major role in the motion of the vehicle in the Propagation boxes of Figure 6.1. One could refer to Figure 6.5 on page 116 to see the full flow chart of the external forces.

The external forces include the aerodynamic force, and the oblate gravity field. The computation of the gravitational force is rather simple, since it makes use of the inertial Cartesian coordinates, and only needs the constants of J_2 and μ.

A little more complicated is the case of the aerodynamic force. To compute it, it is necessary to obtain the density, and thus the altitude, which is a function of the position and velocity in the spherical coordinates, as well as the relative velocity. Moreover, the direction of the aerodynamic force in the inertial frame is a function of all the angles in the spherical, relative coordinates, as well

Table 6.1: Integration phases characteristics.

Phase name	End Conditions (OR)	Forces	Perturbations	Integrator
Exoatmospheric	$\Delta t > 3000$ s, $D > 0.5$ g	All	All but small-scale	Adaptive
Closed-loop	$D < 0.5$ g (or $h > 100$ km), $h < 0$ km, $\gamma_{\mathcal{R}} <$ -70°, $V_{\mathcal{R}} < 0.5 V_c$	All	All	Fixed-step
Exit	$\theta = 180°$, $\Delta t \geq 3000$ s, $h < -1$ km, $\gamma_{\mathcal{R}} <$ -89°, $V_{\mathcal{R}} < 0.3 V_c$	All	All but small-scale	Adaptive

Table 6.2: Constants used.

Name	Symbol	Value	Units
Earth rotation	ω_{cb}	7.2921159×10^{-5}	$\mathrm{rad\,s^{-1}}$
Earth gravitational parameter	μ	$3.986004418 \times 10^{14}$	$\mathrm{m^3\,s^{-2}}$
Earth J_2	J_2	$1.08262668 \times 10^{-3}$	-
Earth Equatorial radius	R_e	6.378137×10^{6}	m
Earth flattening parameter	f	$3.352810664747 \times 10^{-3}$	-
Air gas constant	R_0	2.8699×10^{2}	$\mathrm{J\,kg^{-1}\,K^{-1}}$
Stefan-Boltzmann constant	σ	5.670367×10^{-8}	$\mathrm{W\,m^{-2}\,K^{-4}}$

as of the bank-angle. Density can also be depend on the large-scale and small-scale perturbations. Because the small-scale perturbations are rather complex, they are treated in Subsection 6.2.5. The forces are also functions of the aerodynamic coefficients, which are, in turn, functions of the local speed of sound (and, therefore, altitude), and relative velocity. Therefore, a few rotations and coordinates transformations are needed to compute the aerodynamic force. Also, many constants are needed. Those are summarized in Table 6.2. For simplicity, not only the constants used for the external forces are included in the table, but also those used in Subsection 6.2.8 (except for those that are specific to some equations).

VERIFICATION

Every block of Figure 6.5 has been verified manually. A few blocks, however, deserved more attention than others. The transformation blocks have been verified for special cases. The altitude routine has been verified by computing the altitudes at the Equator and at the Poles. The transformation to rotating coordinates has been verified with its inverse; in addition, it has been tested that it would not cause singularities at specific states, such as zero or vertical velocity, or position over the Poles. Eventually, an additional verification can be seen in Subsection 6.3.1, where the propagator is compared to propagators that use different variables. Below are reported the verification for the tool outputting the external forces. However, this is not to be confused with the full validation, that includes validation of the vehicle and atmospheric models, and which is reported in Subsection 6.2.7.

Central Gravity. The gravity force has been verified in two steps. First step consisted in reproducing a Keplerian orbit, neglecting any perturbations due to either atmosphere or J_2. The simulation consisted of reproducing an orbit with a semi-major axis of 7000 km, inclination of 30° and eccentricity 0.1 for the duration of a full civil day. Using a very high accuracy propagator, the change

in semi-major axis at the end of the simulation was 3×10^{-3} m. The eccentricity has a variation of 3.5×10^{-11}. Other parameters, such as the inclination or the RAAN, have variations that are on the order of magnitude of the machine accuracy. Considering that a full aerocapture would last no longer than one hour, this makes the verification successful.

J_2 Gravity. The second step consists of verifying the effect due to J_2. This is done for the same conditions as before, except that the eccentricity is now set to 0. The simplest way to verify this block consists of using the energy equation including the potential energy due to J_2:

$$E = \frac{1}{2}V_{\mathcal{I}}^2 - \frac{\mu}{r} + \frac{1}{2}\mu J_2 \frac{R_e^2}{r^3}\left(3\sin^2\delta - 1\right). \tag{6.1}$$

This energy is constant during the entire orbit, with variations due to numerical errors in the order of $10^{-3}\,\mathrm{J\,kg^{-1}}$. Hence, the acceleration due to the oblate gravity field can be considered verified.

Drag Force. The drag force test is the first of two tests involving the aerodynamic force. The test consists of letting an object fall from high altitude, and compare its speed to the local terminal velocity, which changes along the altitude due to changes in gravity and, mainly, density. The vehicle is supposed to accelerate as long as its velocity is smaller than the local terminal velocity. Once it is larger, it should start decelerate, and asymptotically reach terminal velocity. This behavior is also seen in any re-entry problem: the vehicle usually continues accelerating for the first part of the flight, until $D > -g\sin\gamma$. In re-entry, since the flight is not vertical, deceleration begins before the speed becomes larger than terminal velocity. The terminal velocity is computed, at any point, setting $D = g$, which implies:

$$V_t = \sqrt{\frac{2gm}{\rho S C_D}}, \tag{6.2}$$

with g and ρ both functions of altitude. In this case, an atmosphere with constant scale height has been used.

To avoid any effects due to Earth's rotation, the freefall starts above the North Pole. The object has a mass of 50 t, and a ballistic coefficient of $176.8\,\mathrm{kg\,m^{-2}}$. In the first attempt, the flight is set to be vertical, and any lift that would cause the flight to change direction is set to zero. Figure 6.2 shows the velocity of the falling object and the local terminal velocity as functions of altitude. It can be seen that the object continues accelerating until it crosses the terminal velocity. The sign of the acceleration is positive as long as the velocity is smaller than the terminal velocity, and is then negative once the velocity overshoots terminal velocity. Afterwards, the two asymptotically approach each other. Interestingly, if the freefall begins from some lower altitude (e.g., 20 km), the overshoot does not occur, and the velocity approaches the terminal velocity from below.

The second case concerns a ballistic descent. Here, the equilibrium velocity involves the sine of the flight-path angle, since, in a non-rotating planet, deceleration only begins when $D > -g\sin\gamma$. Therefore:

$$V_{eq} = \sqrt{\frac{-2gm\sin\gamma}{\rho S C_D}} \qquad \text{if} \quad \gamma < 0, \tag{6.3}$$

whereas if $\gamma > 0$ the vehicle decelerates in any case. In this second test, the flight begins horizontally at an altitude of 100 km with circular velocity. Deceleration starts immediately, even though the velocity is smaller than the terminal velocity, because the flight-path angle is zero, and thus the equilibrium velocity is also zero. Because of the drag, the horizontal velocity decreases continuously: hence, the terminal and equilibrium velocities become asymptotically the same, and so does, even though more slowly, the vehicle's velocity. This process can be seen in Figure 6.3.

Consequently, the drag force can also be considered verified.

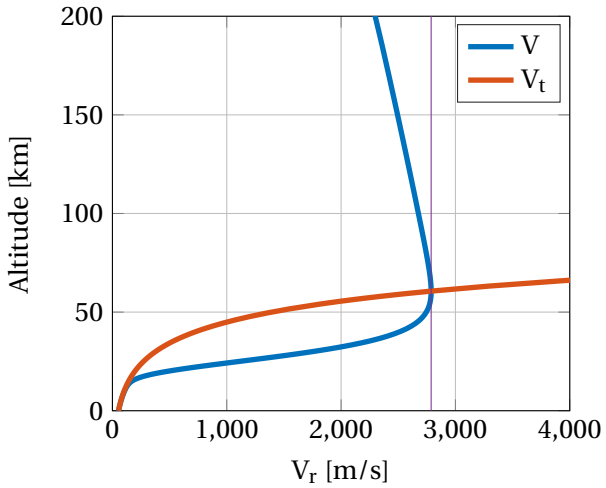


Figure 6.2: Velocity and terminal velocity during freefall from 500 km altitude.

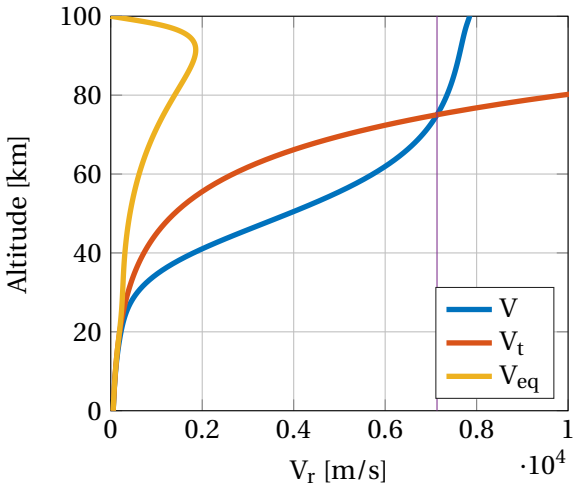


Figure 6.3: Velocity, terminal velocity and equilibrium velocity during ballistic entry.

Lift Force. The lift force is verified using the analytical solution of skipping entry. Analytical solutions make use of a variety of assumptions. It will be shown that for conditions that tend to those assumptions, the simulator gives solutions that tend to the analytical ones. Two of the predictions of the theory are [Mooij, 2014]:

$$\gamma_F = -\gamma_E; \tag{6.4}$$

$$V_F = V_E e^{\frac{2\gamma_E}{L/D}}. \tag{6.5}$$

Two of the main assumptions of the analytical solutions are flat Earth and aerodynamic force much larger than gravity. Both of these become closer to be true if the radius of the plane tends to infinity. Therefore, the two equations above mentioned will be tested for increasing planet radius. Additional assumptions include constant aerodynamic coefficients and exponential atmospheric density. The entry interface occurs at 100 km, with local escape velocity and a relative entry flight-path angle of $-10°$: this way, the expected ratio between entry and final velocity is 2.0100 for $L/D = 0.5$ and 1.4177 for $L/D = 1$.

Figure 6.4 shows the trajectories for increasing Earth radius. The vertical lines correspond to the theoretical, analytical values. The trajectories have been computed for nine exponentially increasing radii of the celestial body, from $R = R_E$ to $R = 256\,R_E$. It can be seen that there is an asymptotic behaviour, and the final velocity tends to the analytical one for the radius tending to infinity. Specifically, when $L/D = 0.5$, the difference is larger than $200\,\mathrm{m\,s^{-1}}$ for $R = R_E$, whereas for the largest radius the difference is about $3\,\mathrm{m\,s^{-1}}$. When $L/D = 1$, the difference decreases from around $400\,\mathrm{m\,s^{-1}}$ to $3.5\,\mathrm{m\,s^{-1}}$. The same convergence occurs for the flight-path angle. Since the plots are very similar, they are not reported here. For $L/D = 0.5$ the final flight-path angle when $R = R_E$ is 7.0268°, whereas for the largest radius it is 9.9999°.

At this point, all the forces have been verified to be working correctly. An additional verification will be done in Subsection 6.3.1. However, these verifications do not include verification of the models used, such as the vehicle, the atmosphere, the planet's shape and so on. For those, the validation of (almost) the full model is reported in Subsection 6.2.7.

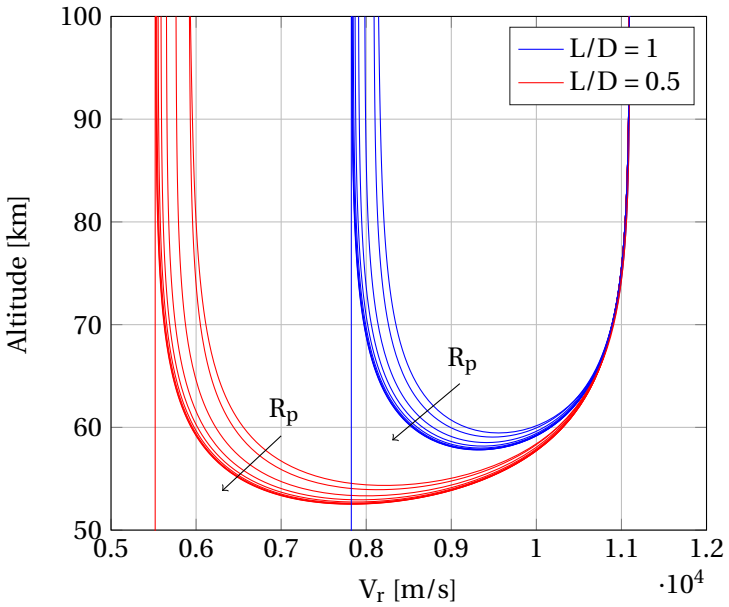


Figure 6.4: Convergence of numerical solutions to analytical solution for increasing planet radius; $BC = 159.15\,\mathrm{kg\,m^{-1}}$.

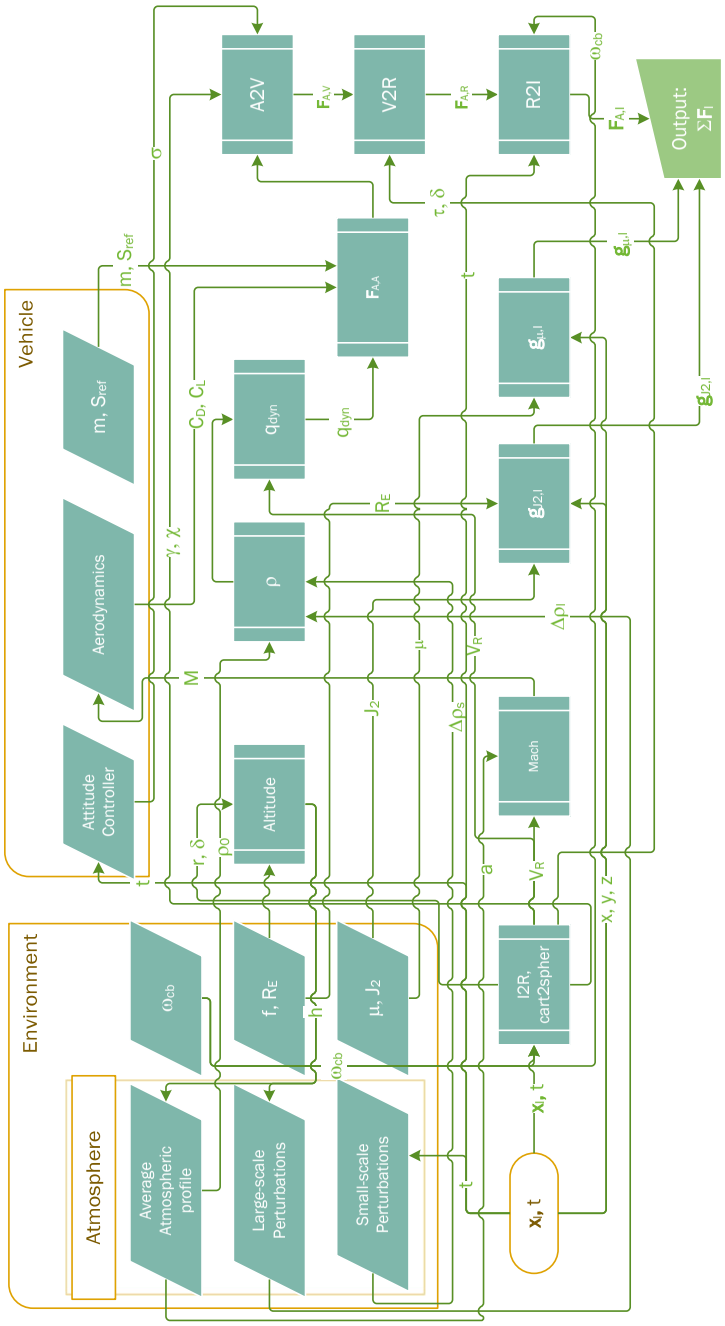


Figure 6.5: Flow chart for the external forces.

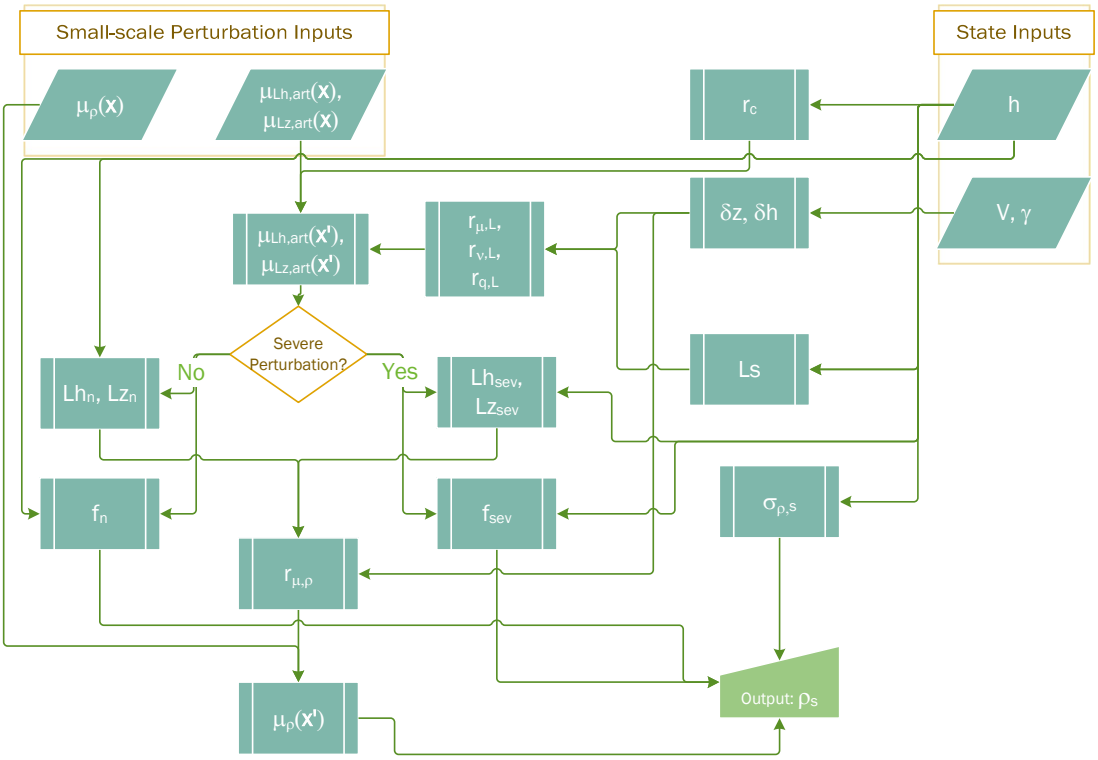


Figure 6.6: Flow chart for the atmospheric small-scale density perturbations.

6.2.4 VEHICLE

The vehicle model does not require much discussion. In fact, all that is used is an interpolation of lift and drag coefficients as a function of Mach number. The angle-of-attack is not even play a role in the simulator, despite it is indirectly needed to compute the corresponding aerodynamic coefficients at each Mach number.

In addition to that, some perturbations may be included. Those are kept constant during the entirety of the trajectory, and are a variation in ±10 % in both lift and drag coefficients (independently), as well as a ±1.5 % variation in mass, unless otherwise specified.

6.2.5 ATMOSPHERIC DENSITY PERTURBATIONS

Three kinds of atmospheric density perturbations are designed to occur in this simulator. The first and simplest one, is that of multiplying the density profile by a constant random number larger than 0. In this research, such number is in the interval [0.5, 1.5]. However, this kind of perturbation will not be the subject of this subsection.

The other two kinds of perturbations are the small-scale and the large-scale perturbations, both according to the Earth GRAM-99 database, as described in Subsection 3.6.3.

The large-scale perturbation subroutine takes as inputs altitude, latitude and longitude of the

spacecraft, together with an array with the 4 random variables described in Subsection 3.6.2 (ϕ_Q, Q, Q_{nm}, and a_v), and uses all this data to compute ρ_l by Equation 3.43. The altitude is then also used to interpolate the value of $\sigma_{\rho,l}$. The product of these is then multiplied by the unperturbed density at the given altitude, and added to it. This is done each time the propagator needs to compute the external forces.

The small-scale perturbation subroutine is instead a little more complicated. In addition, it requires a particular initialization. The initialization takes as input a seed to initiate its random sequence. Since the small-scale perturbations require a random number at each step, and since the duration of the trajectory is unknown before-hand, it would not make much sense to provide an entire random sequence before. Therefore, for each trajectory a random seed is provided to initiate the sequence. Moreover, the model includes two cross-correlated variables, which are used to compute the artificial horizons of the horizontal and vertical scale sizes. To be sure that the cross-correlation is not affected by the two initial values randomly given to those, a sequence of 100 of those is generated.

The small-scale main subroutine is activated at each guidance call, and it takes as inputs altitude, velocity, and flight-path angle of the vehicle, together with the last values of the sequences described in Subsection 3.6.2. It then estimates the expected displacements by linearization, and uses them to compute the various auto-correlation terms. After that, the random variables are generated, the severe perturbations condition is checked, and ρ_l at the next guidance call is generated, using the $\sigma_{\rho,l}$ for the current altitude. This last detail makes it obvious that a discontinuity in density occurs at each guidance call. However, such discontinuity is rather small, since $\sigma_{\rho,l}$ does not change very fast with altitude. Also, the discontinuity is always outside of the integration boundaries. At last, a linear, time-dependent profile connecting the current $\mu_\rho(\mathbf{x})$ to the next one is generated, and then multiplied by the current $\sigma_{\rho,l}$. At each function evaluation, such value is then multiplied by the local density, and added to it. The architecture for this piece of software can be seen in Figure 6.6 (only for the small-scale perturbations, since the large-scale perturbations are much simpler).

VERIFICATION

Each block of the large-scale subroutine has been verified by simply checking, by hand each of the single operations. Concerning the small-scale software, the same thing has been done for most of the blocks. However, the most complicated ones, which are those generating the auto-correlated and the cross-correlated variables, have been subject to a more intense statistical campaign.

Specifically, the block generating $\mu(\mathbf{x}')$ has been verified by generating one million samples with two different values of $r_{\mu,\rho}$. The verification has been carried out by using the property of Equation 3.48.

The same thing has been done for the artificial scale sizes. In that case, the cross-correlation has also been checked, using the property of Equation 3.50. It turned out that, in every case, the average product between the two variables came much closer to the value of r_c when removing the first 50 variables. That is due to the fact that the first part of the sequence is rather largely dependent on the initial choices for ν_L and μ_L. For this reason, the sequence was always initialized by generated 50 cross-correlated variables before using them in the trajectory.

At last, the frequency of the severe perturbations was also checked. However, after a statistical evaluation, it ended up being almost twice as likely than predicted according to P_{sev}. This somewhat makes sense. In fact, severe perturbations occur when either of the artificial scale sizes go beyond the P_{sev} value. The two scale sizes are cross-correlated, but their cross-correlation is relatively small (in the tested case, at $h = 70$ km, it is 0.64) and $P_{sev} = 0.01$ on average. Therefore, it is very unlikely that both scale sizes cause turbulence at the same time. However, the software is kept like this: in fact, it is preferred to have a more turbulent than usual model.

As desired by the authors of the model, the severe perturbations turn out to be "patchy", in the

Table 6.3: Statistical verification of the small-scale perturbation model. Cases 1 and 2 refer to a simulation lasting 100 hours, whereas the simulation in Case 3 lasts 1 000 hours. r_ρ expected is computed using the value of the occurred P_{sev}.

	Conditions				Expected values			
	V[km s^{-1}]	h[km]	γ [deg]	dt [second]	r_ρ	$r_{c,L}$	r_{Lz}	r_{Lh}
Case 1	10	70	0	0.1	0.9964	0.64	0.9995	0.9995
Case 2	10	70	-3	0.1	0.9896	0.64	0.9967	0.9967
Case 3	10	70	-3	1	0.9092	0.64	0.9675	0.9675
Case 4	10	70	-3	0.1	0.9895	0.64	0.9967	0.9967
	Standard deviations				Occurred values			
	μ_ρ	ν_{Lz}	μ_{Lh}		r_ρ	$r_{c,L}$	r_{Lz}	r_{Lh}
Case 1	0.9931	0.9867	0.9850		0.9829	0.6039	0.9733	0.9699
Case 2	0.9927	0.9933	0.9963		0.9760	0.6280	0.9833	0.9894
Case 3	0.9996	0.9975	1.0001		0.9084	0.6379	0.9626	0.9677
Case 4	1.0005	1.0005	1.0001		0.9915	0.6421	0.9977	0.9970
	Severe Turbulence							
	P_{sev}		P_{sev} occurred		Patchiness		Turbulence size [km]	
Case 1	0.0081		0.0124		0.9633		27.2	
Case 2	0.0081		0.0134		0.9066		10.7	
Case 3	0.0081		0.0142		0.7356		37.8	
Case 4	0.0081		0.0145		0.9133		11.5	

sense that once the severe perturbation is triggered, it is likely to last more than only one iteration. Patchiness was measured by computing the average product of the severe turbulence index (equal to 0 if non severe, equal to 1 if severe) at iterations i and $i-1$, and dividing it by the average value of the severe turbulence (which is the occurred probability of having severe perturbations). From that, it is possible to derive the average duration of severe turbulence, $1/(1-\text{patchiness})$, and, consequently, the physical size of the turbulence.

Table 6.3 shows the expected values and the obtained one for the statistics of three simulations. All three cases refer to a hypothetical spacecraft that flies at the same speed, altitude and flight-path angle for 100 hours in Cases 1 and 2, and 1 000 hours in Cases 3 and 4.

There are some inaccuracies in the first two cases. The standard deviations are instead all very close to one (as expected), even though they are much closer to it in Case 3. The reason why Case 3 is more similar to expectations is because it has smaller autocorrelation values. Therefore, less samples are needed to have a statistically representative case. Case 3 is also more accurate in all the autocorrelation and cross-correlation values.

Eventually, Case 4, which is exactly the same as Case 2, but run for 10 times the time (and, thus number of samples) shows how the model tends to the expected value with an increasing number of samples.

Therefore, this tool can be considered verified. The only unexpected result is the fact that the turbulence has a much larger average size in Case 3 than in Cases 2 and 4, whereas they should, in theory, be the same. However, this value is even more sensitive to the sample size than the previously discussed values, since turbulence occurs only about 1 % of the times.

6.2.6 IDEAL CONTROLLER

Applying an ideal controller to the simulator is a simple task. To include an ideal controller that satisfies the constraints on angular rate and acceleration, while, at the same time, achieving the target command in minimum time, and without overshoot, is slightly more elaborate.

In this subsection, a way for the guidance commands to satisfy the constraints on the attitude

dynamics is proposed. Alternatively, this can be seen as a way to reproduce attitude dynamics as controlled by an ideal controller that respects the physical constraints of the vehicle. In addition, such an ideal controller achieves the target command in minimum time. It is stressed that the attitude dynamics are not simulated, and bank-angle and bank-angle rate behave as state variables that are only functions of the bank-angle acceleration. There are, thus, no actuators involved in the simulation.

To not complicate matters further, the attitude variables are expressed in the aerodynamic frame. Moreover, as already mentioned, the angle-of-attack is always in trim position, and is assumed to be subject to no dynamic constraints, whereas the sideslip angle is always set to zero. Any perturbations in those angles are not considered.

The only attitude variable controlled is the bank-angle, subject to angular rate and acceleration constraints. It is assumed that the only torque acting is the one due to the RCS.

The fact that there are no perturbations makes the problem in part simpler. However, a simple proportional controller would overshoot, whereas a proportional-integral-derivative (PID) would at least need some tuning, and it is not sure it could achieve optimality.

For the controller to be ideal, it should reach the commanded bank-angle in the shortest time possible, while not exceeding the constraints. This could be set up as a minimum time and constrained optimal control problem. However, due to the simplicity of the problem, another option is possible.

To achieve minimum time, the RCS always operates in a bang-bang fashion, which implies maximum acceleration and deceleration, unless already at maximum rotational rate. The ideal controller plans a trajectory (here, the word trajectory is used in the sense it has in control theory, which is that of a control- and/or state-history) in the $t-\dot{\sigma}$ plane. It designs a trajectory in that plane which has the following characteristics:

1. Begins with $\dot{\sigma} = \dot{\sigma}_0$, and always finishes with $\dot{\sigma} = 0$.
2. The inclination of the line is always $\|\ddot{\sigma}\| = \ddot{\sigma}_{\max}$, or $\ddot{\sigma} = 0$, only if $\|\dot{\sigma}\| \equiv \dot{\sigma}_{\max}$.
3. The integral of the trajectory is such that $\sigma_0 + \int_{t_0}^{t_f} \dot{\sigma}\, dt = \sigma_{cmd}$.
4. The trajectory has maximum two inclined segments, and a total of no more than three segments.

These characteristics together imply that the trajectory leads from the current state to $\sigma_f = \sigma_{cmd}$ and $\dot{\sigma}_f = 0$ in the shortest time possible. The initial and final conditions are ensured by the Characteristics 1 and 3. Characteristic 2 implies that the planned trajectory has the maximum integral, in absolute value, in the shortest time. Characteristic 4 implies that there cannot be any overshoot. Moreover, the integral of any trajectory planned following Characteristic 2 can be analytically computed using areas of rectangles and triangles only.

If the difference between current bank-angle and commanded bank-angle is larger than 180°, the controller also decides which direction to turn. This is done, however, without considering the current initial bank rate. How a trajectory respecting these characteristics is designed is the subject of the remaining part of this subsection.

At first, the ideal controller tries to plan a trajectory in which the final bank-angle rate is set to zero, as fast as possible. If that leads to a final error $\Delta\sigma_0^* = \left\|\sigma_f^* - \sigma_{cmd}\right\| < \Delta\sigma_{\max}$, then that will be the trajectory. If that is not the case, the controller computes the maximum rotational velocity $\dot{\sigma}_{\max,aux}$ it has to reach to achieve σ_{cmd} in the shortest time possible, while subject to acceleration constraints, but no velocity constraints. Using basic triangle geometry:

$$\dot{\sigma}_{\max,aux} = \operatorname{sign}\left(\Delta\sigma_0^*\right)\sqrt{\left\|\Delta\sigma_0\ddot{\sigma}_{\max} + \frac{1}{2}\dot{\sigma}_0^2\right\|}. \tag{6.6}$$

The reason of this equation can be deduced by Figure 6.7. Because the acceleration is always maximum, $\Delta T_a = 2\dot{\sigma}_{\max,aux}/\ddot{\sigma}_{\max}$, and also, $\Delta T_{\dot{\sigma},0} = \dot{\sigma}_0/\ddot{\sigma}_{\max}$, no matter the sign of $\dot{\sigma}_0$. Therefore, $\Delta\sigma_0$, which is the integral of the trajectory, is always equal to the area of the blue triangle, minus the area of the red dashed triangle. That is because, if $\dot{\sigma}_0$ and $\dot{\sigma}_{\max,aux}$ have the same sign, then that's equal to just shifting the t_0 line to the right. In case they are not, the t_0 line is shifted to the left. Then the integral includes the entire blue triangle, and the red triangle. But the latter is in the negative semi-plane, and it is therefore subtracted. Consequently:

$$\Delta\sigma = \frac{\dot{\sigma}^2_{\max,aux}}{\ddot{\sigma}_{\max}} - \frac{1}{2}\frac{\dot{\sigma}_0^2}{\ddot{\sigma}_{\max}}, \tag{6.7}$$

which, when inverted, gives Equation 6.6. The reason why $\dot{\sigma}_{\max,aux}$ takes the sign of $\Delta\sigma_0^*$ instead of the sign of $\Delta\sigma_0$ is that, as previously mentioned, the controller first checks what the final bank-angle error is if simply the rotational rate were zeroed. Since that part is necessary, because one of the conditions is the zero of the bank-angle rate at the end of the trajectory, it is only depending on that sign, that the controller decides whether the integral that should be added is either positive or negative.

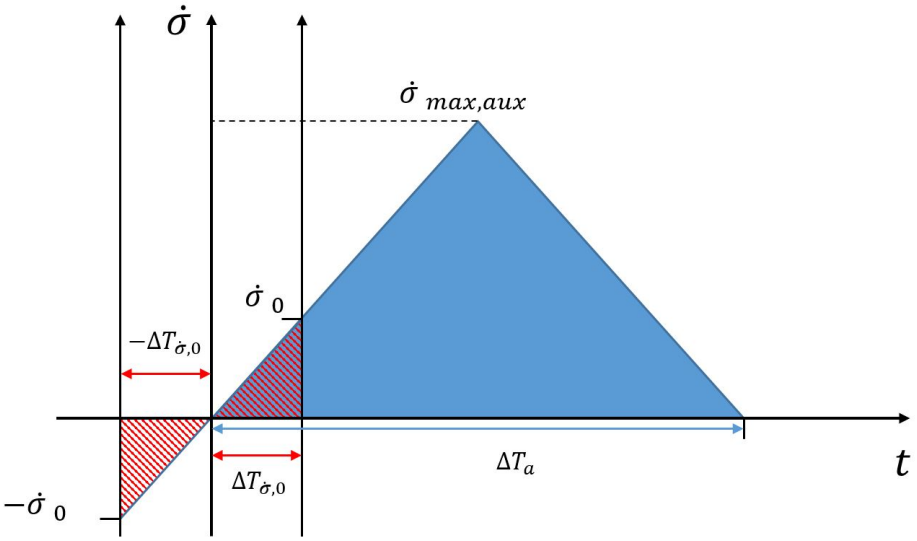


Figure 6.7: Independence of $\dot{\sigma}_{\max,aux}$ with respect to the sign of $\dot{\sigma}_0$.

At this point, it is necessary to check whether the rotational rate saturates, which is, whether $\left\|\dot{\sigma}_{\max,aux}\right\| > \dot{\sigma}_{\max}$. If not, the planning is complete. If yes, an additional correction is needed. Looking at Figure 6.8:

$$\Delta T_{sat,1} = \frac{2\left(\left\|\dot{\sigma}_{\max,aux}\right\| - \dot{\sigma}_{\max}\right)}{\ddot{\sigma}_{\max}}. \tag{6.8}$$

Moreover, the red dashed triangle gives an additional integral that should be added to the trajectory, by lengthening it in time. The addition in time is equal to the area of the triangle, dividing by the maximum angular rate:

$$\Delta T_{sat,2} = \frac{\frac{1}{2}\Delta T_{sat,1}\left(\left\|\dot{\sigma}_{\max,aux}\right\| - \dot{\sigma}_{\max}\right)}{\dot{\sigma}_{\max}}. \tag{6.9}$$

The trajectory would then be the triangle cut as in Figure 6.8 with added, during the saturated part, an additional saturated part of duration $\Delta T_{sat,2}$, and is shown in Figure 6.9.

As an addition, the piece of software has the option of not commanding any change if the difference between the current bank-angle (if the rotational rate is zero) and the commanded bank-angle is smaller than a certain threshold. With this method, it is also easy to implement additional possible random error margins in the final bank-angle and bank-angle rate.

When seeing the procedure in the $\dot{\sigma} - t$ plane, it can be possible to prove that this trajectory minimizes Δt. In fact, to do so, it is necessary to have the largest possible integral below the trajectory, which is obtained by having a trajectory that is always as steep as possible, until it reaches $\dot{\sigma}_{\max}$.

Eventually, the bank-angle trajectory is implemented as a time-dependent parameter in the equations of motion. The trajectory is renewed at the following guidance command.

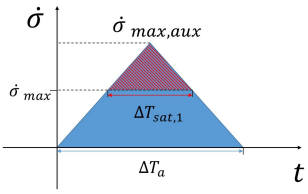


Figure 6.8: Saturation of bank-angle rate.

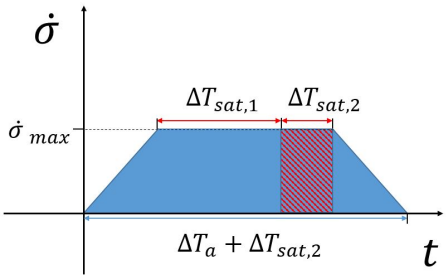


Figure 6.9: Final bank-angle trajectory.

VERIFICATION

The ideal controller has been verified in two steps. At first, it has been checked whether the generated trajectory followed all the characteristics mentioned above. This was done visually, by accurately selecting a variety of cases. In these cases, it is attempted to have different situations, such as $\Delta\sigma_0$ and $\dot{\sigma}_0$ with same and opposite sign, as well as cases with and without saturation, and a case with $\Delta\sigma_0$ and $\dot{\sigma}_0$ with same sign, but with $\Delta\sigma_0^*$ with opposite sign (in the legend, it is the case where σ_f =105°). In every case checked, the ideal controller behaved as expected.

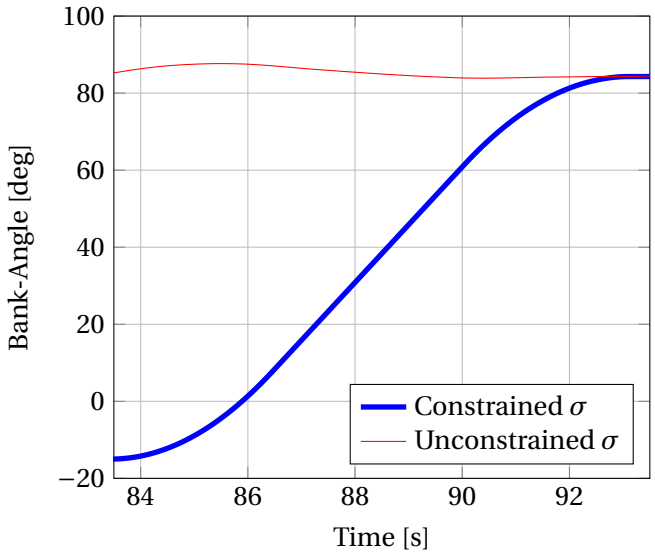


Figure 6.10: Simulation of the constrained bank-angle immediately after Phase 2 has been triggered.

The controller can eventually be validated by visually checking its performance when in action. In Figure 6.10, it is shown how it performs starting from the very moment in which the *Lu* guidance

described in Chapter 4 triggers Phase 2 (with $\sigma_d = 85°$ in this case), at $t = 83.5$ s. In the case reported here, both the controller and the guidance update with a frequency of 10 Hz. Being the commanded angle very far from the current angle, the controller plans saturation. It takes indeed about 3 seconds, as expected, to reach full speed, which occurs at $t = 86.5$ s. The same thing happens when decelerating. The inclination of the line after saturation is, also as expected, equal to $15\,°\,s^{-1}$.

6.2.7 SIMULATOR VALIDATION

In addition to all the verifications reported in the previous subsections, the simulator can be validated by comparison with Robinson et al. [2009]. In that paper, many details are given about how the simulations were simulated. Moreover, those simulations were carried out using Program to Optimize Simulated Trajectories (POST), a NASA software that has been validated using real flight data. Of particular interest are the results reported in Table 5 of Robinson et al. [2009], which concern Earth aerocapture. The simulated flight is for the same capsule used in this research; the atmospheric model used is the US76, and the initial velocities and flight-path angles, together with the final conditions (the apoapsis is always 500 km) are given. Moreover, the gravity model includes J_2, (despite the surface model is spherical). The initial heading and latitudes are unknown. Therefore, in this validation, the flight begins at the Equator, and is headed northwards, with $\chi_{\mathcal{R},0} = 0°$.

The flights are for constant bank-angles, which means that no guidance is required for the validation. With this comparison, the software will be validated in its rather basic version: unguided, uncontrolled, for unperturbed US76 Atmosphere, and spherical Earth, with oblate gravity field. In addition, in some cases the mass of the spacecraft has been modified, to match the various ballistic coefficients of the paper.

The first case to be tested is Case 1 of Table 5 of Robinson et al. [2009]. It is an Earth aerocapture at Mars entry conditions, with full lift down. It has been chosen because with that trajectory the heat-flux could also be validated. The entry flight-path angle that was found to achieve a 500 km altitude aerocapture is $-5.101\,505°$, differing by at least 0.005° from the entry angle in the paper. The amount of significant digits is important in this case: in fact, already an entry angle of $--5.101\,506°$ causes a miss of about 40 km. The other cases tested are Case 2, which is equivalent to Case 1, but with full lift-up, and Cases 11 and 12, which are more challenging since the entry velocity is much larger. Also, a different ballistic coefficient is used in the last 2 cases.

In Case 2, the mismatch is a little larger. This is due to the fact that the trajectory is full-lift up, and therefore there is a much smaller sensitivity to entry conditions. As an example, in this case, a difference in entry conditions of 0.01° causes an miss of 20 km, whereas in Case 1, the miss was 40 km with a difference in entry angle 10^4 times smaller. The mismatch in Case 12 is the largest, amounting to about 0.04°, whereas the mismatch in Case 11 is the smallest, being at least 0.003° (and also the most sensitive to initial conditions). Table 6.4 shows a summary of the 4 validation cases that were tested. Figure 6.11 shows instead the trajectories corresponding to the various cases.

The theory that the small mismatch in entry conditions is due to the unknown initial latitudes and headings, and not to error in the simulator, is supported by the propagators analysis in Subsection 6.3.1, in which it is shown that 3 different propagators, all using diffent equations of motion (Cartesian inertial, spherical relative dimensional, and spherical relative dimensionless), lead to solutions that differ only by negligible values (in the order of magnitude of meters in apoapsis altitude). However, this is not entirey true, since all the propagators use the same models for atmosphere and vehicle aerodynamics, and therefore the mismatch might be in those (despite the output of all those subroutines have been verified independently).

6.2.8 HEAT-FLUX ESTIMATOR

The heat-flux has been estimated off-line using the theory of Subsection 3.5.5. The heat-flux is computed using the Detra-Hidalgo relation for the convective heat-flux, with hot wall correction, and

Table 6.4: Comparison of some cases from Table 5 of Robinson et al. [2009] with the results obtained with the simulator developed in this research.

	Original Data				Validation
Case no.	$V_{\mathcal{R},0}$ [$\mathrm{m\,s^{-1}}$]	$\gamma_{\mathcal{R},0}$ [°]	BC ($M=30$) [$\mathrm{kg\,m^{-2}}$]	σ [°]	$\gamma_{\mathcal{R},0}$ [°]
1	12,201	-5.09	122	180	-5.101505
2	12,201	-7.29	122	0	-7.326
11	16,007	-6.39	356	180	-6.39836365
12	16,007	-10.18	356	0	-10.221

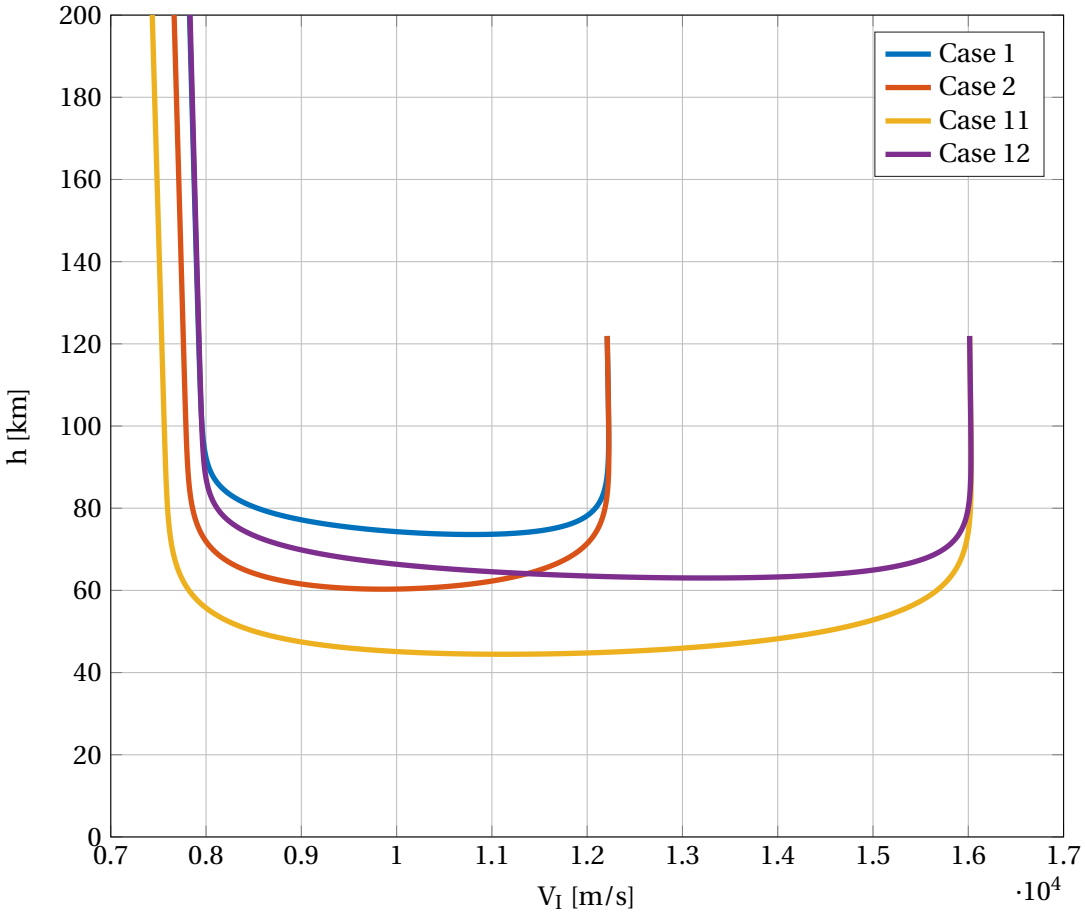


Figure 6.11: Trajectories corresponding to 4 different cases of Table 5 of Robinson et al. [2009], which are summarized in Table 6.4.

the Tauber-Sutton relation for the radiative heat-flux. The adiabatic wall temperature is computed from the total enthalpy interpolating data from Menart and Henderson [2008]. The wall temperature is computed assuming an emissivity coefficient $\epsilon = 0.8$, equal at every wavelength[1]. Two different effective nose radii are used: an effective nose radius of 4 m is used for the convective flux, and an effective nose radius of 2.9 m is for the radiative flux. Both have been chosen according to Robinson et al. [2009], and are the effective radii corresponding to an angle of attack of about -20°, which is the one occurring at superorbital velocities.

[1] The emissivity/absorptivity for the radiative heat-flux is instead set to 1. This is not a contradiction, since the emissivity/absorptivity of a real material is a function of the wavelength. It is unknown what value is used by Robinson et al. [2009].

The wall temperature requires an iterative procedure to be accurately estimated.however, it is not a problem to estimate the wall temperature using the heat-flux of the previous time-step, as it is also done by Robinson et al. [2009]. The wall temperature is initialized setting equilibrium at the first time-step, in which the convective heat-flux is not corrected for hot wall. When computing the wall temperature, it is assumed that it has zero conductivity towards the inside of the spacecraft, and no ablation heat is considered either.

For the same reasons, it is not a problem to integrate the total heat-load along the trajectory using very low accuracy quadrature methods: the integral is computed by summing the heat-flux history, which is sampled at 1 s frequency. Figure 6.12 shows the process described above.

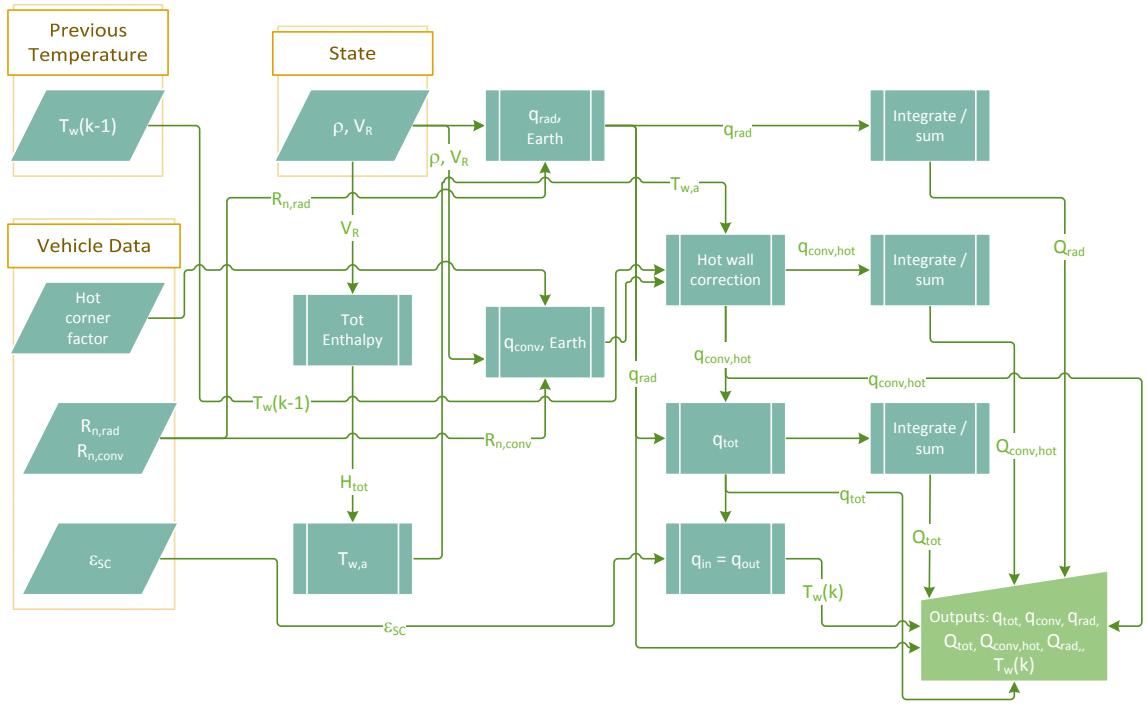


Figure 6.12: Flow chart for the heat-flux and heat-load estimator.

VALIDATION

The equations for radiative and convective heat have both been manually verified. Specifically, the radiative heat equation has been verified in all its components, which are the linear interpolation, the computation of the exponent for the nose radius, and the final equation.

The computation of the temperature by energy balance has been verified the same way. In addition, the heat-flux for the trajectory of Case 1, Table 5 of Robinson et al. [2009] was evaluated, and compared to the heat-flux computed by then. The comparison was done for stagnation point only.

Also the interpolation of the data from Menart and Henderson [2008] has been visually verified, by checking that the shape and magnitude of the data was the same as in their plot.

For a validation of the tool, the reader may refer to Figures 6.13 and 6.14, keeping in mind that the heat-rate history computed here is based on a trajectory that is slightly different from the original one.

It was found that the peak convective heat-flux is larger than expected, by about 25 %. However, this should not surprise, since a different (less accurate, and more conservative) formula was chosen. As an example, using the Sutton-Graves equation instead, one would obtain (for the same trajectory), a peak convective heat-flux that is much closer to that of the compared paper, being larger by less than 10 %. Once more, this proves the limited applicability of these approximations, and how these should be used as an indicator of the order of magnitude rather than as an accurate estimate.

The radiative heat-flux is also a little off, with an error of around 20 %, being the peak 1.01 $\mathrm{MW\,m^{-2}}$. This is less expected, since the formulation used is exactly the same. However, when seeing these differences, one should also keep in mind that the trajectory used for validation is not exactly the same as the one of the paper, as shown in Subsection 6.2.7. This is something not to be underestimated, since the heat-flux (especially the radiative) is extremely sensitive to the density. It is estimated that if the heat-flux peak were off by 1 km in altitude (which is not so unlikely, given the different initial conditions), and therefore by 0.85 times in density, the radiative heat-flux peak would reduce to 0.83 $\mathrm{MW\,m^{-2}}$. Convective heat-flux is instead much less sensitive to the density, and therefore such variation would not affect it much (about 7 % only: this way, the Sutton-Graves equation would match very well with the validation data). Wall temperature profile matches instead very closely the target one, with a peak of about 2500 K, but this is no surprise, since the temperature is much less sensitive to any heat-flux error.

In both the simulated case and the one used as comparison, the radiative heat-flux peak slightly anticipates the convective peak, and occurs a little before 100 s after entry interface.

Eventually, the hot corner has not been estimated. The reason is that in Robinson et al. [2009], for the same trajectory, the ratio between stagnation and hot corner heat-flux is far from being 1.6 (and the temperature is almost the same, meaning that the difference is not due to a change in wall temperature). Moreover, at the hot corner the radiative heat-flux is different than at stagnation point. All these factors make it such that it is likely that there are more factors to be held into account when estimating the fluxes at the hot corner.

As a conclusion, the validation can be said to be successful. All these equations are in fact extremely sensitive to the trajectory, and the use of different equations can matter a lot. As previously mentioned, if the peak occurred in Robinson et al. [2009] at 1 km altitude higher than in the here simulated trajectory, which is very possible, considering the fact their entries are slightly shallower, the radiative heat-flux peak would be almost the same. In the same conditions, using the Tauber-Sutton relation instead of the Detra-Hidalgo, the convective heat-flux would also be almost the same. Nevertheless, the Detra-Hidalgo relation is chosen, because better suited to the problem, and more conservative. Moreover, the difference between the two relations is in the range found in Carandente et al. [2013].

6.3 PROPAGATORS AND INTEGRATORS ANALYSIS

Four different kinds of propagators have been analysed during this research. It was decided, in Chapter 3, that a propagator in Cartesian and inertial coordinates will be used for the simulator. The NPC guidance uses instead spherical relative coordinates.

A guidance logic needs to integrate the trajectory on-board and in real-time. Therefore, a trade-off between accuracy and computational time is needed. Commonly used implementations for this goal are described in Chapter 3, and are:

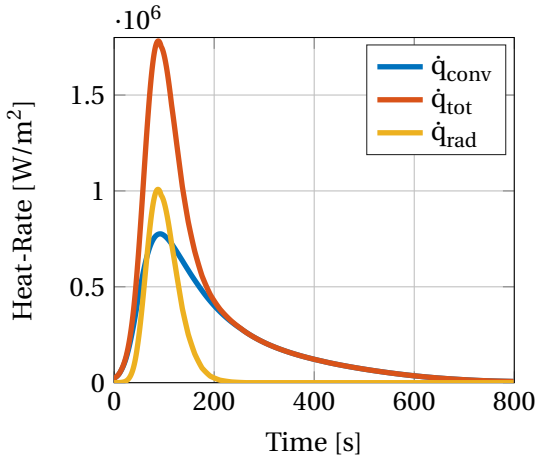


Figure 6.13: Computed heat-flux for Case 1, Table 5, of Robinson et al. [2009], at stagnation point.

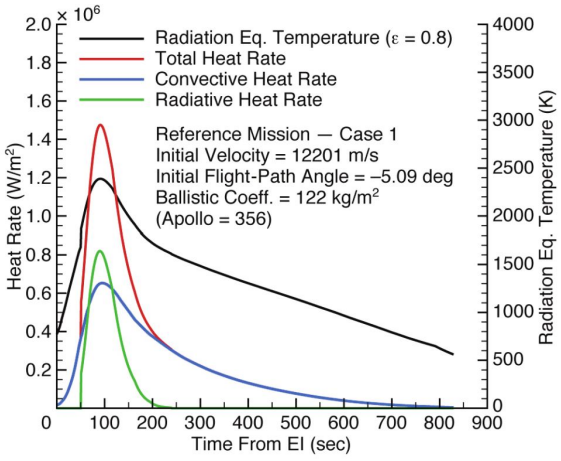


Figure 6.14: Heat-flux for Case 1, Table 5, at stagnation point. From Robinson et al. [2009].

- Dimensional equations, time used as independent variable
- Dimensionless equations, time used as independent variable
- Dimensionless equations, energy used as independent variable

For simplicity, throughout this section, when talking about spherical coordinates, it is implied that a rotating frame is used.

The goal of this section is that of analyzing the effect of the choice of a different integrator on speed and accuracy of the solution. Moreover, as mentioned in Subsection 3.9.3, the integration using energy as independent variable introduces some errors, which shall be quantified.

The error in final velocity (as function of either energy or altitude) is the only performance parameter that will be used. Some importance to the error history will also be given.

Cartesian inertial equations will also be used for verification. A match between spherical and Cartesian propagators would be a final verification of both the equations of motion.

For simplicity, the mission in the first part of this section is a normal entry case, and not an aerocapture. The baseline mission used here consists of an entry on a rotating, oblate Earth beginning at an altitude of 100 km, with a relative velocity of 11 km s^{-1} and a relative flight path angle of -5°, at a latitude of 45° North. The initial heading is 45°, and the bank-angle is equal to 90° during the entire flight: from a planar point of view, this entry is ballistic; the flight is therefore unguided, and is set to last 180 s, unless otherwise stated. The vehicle used is the Apollo Command Module in trimmed condition. The mass of the Apollo Command Module is reduced to 5,115 kg, and its lift coefficient is reduced by 10 %: by so doing, at hypersonic speed, this vehicle has the same ballistic coefficient and lift-to-drag coefficient as the Orion MPCV, used by Lu et al. [2015].

6.3.1 PROPAGATORS VERIFICATION

In this subsection the equations of motion are verified by comparison. First, Cartesian equations of motion will be used as benchmark for the spherical rotating. This does not mean that the Cartesian equations of motion are more correct. Simply, to plot the difference, one of the two has to be set as reference. If, then, the difference between the two propagations is negligible, both can be considered verified.

Figure 6.15 shows that the propagators all agree with their solutions, when integrated with an RK5 and a time-step of 0.1 s. The final difference between Cartesian, inertial, and spherical, dimen-

sional propagators is in the order of 10^{-10} m s^{-1}, with a maximum error on the order of 10^{-9} m s^{-1} during the integration. The maximum difference between Cartesian, inertial, and spherical, dimensionless coordinates is instead one order of magnitude larger.

It is impossible to state which of these solutions is more accurate, but it does not matter, considering the very small difference.

For the above mentioned reasons, when referring to the "exact solution" it will be referred to one of these solutions; moreover, when talking about error, it will be meant a displacement from exact solutions obtained with RK5 and a step-size of 0.1 s.

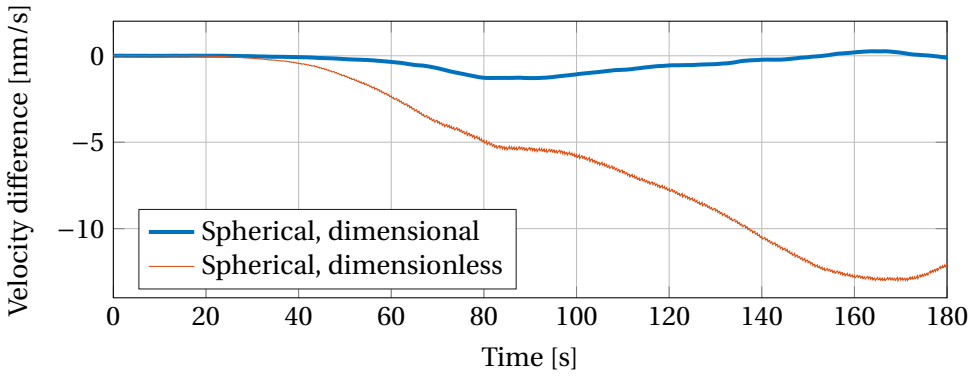


Figure 6.15: Displacement in velocity of spherical, dimensional, and spherical, dimensionless, coordinates (both with time as independent variable) from propagation in Cartesian, inertial, coordinates. Integration step of 0.1 s.

6.3.2 ENERGY-INDEPENDENT PROPAGATOR

The investigation and comparison of the propagator that uses energy as independent variable with the other propagators is a little more complicated. In fact, it is at first necessary to know what the final energy is. This is done using the previously mentioned "exact solution". Moreover, the integration will occur with uniform energy steps, which will end up to be very different from the uniform time-step, it is necessary to interpolate the values of one solution to find the difference between the other. In this case, it is decided to compute the dimensionless energy at each time-step of the "exact solution"; then, the velocity is interpolated with a spline[2] function with respect to dimensionless energy, and the difference is computed at each energy-step of the solution that uses energy as independent variable. The result can be seen in Figure 6.16.

It should be stressed that at any point in that plot the energy is the same for the two compared solutions: therefore, for an error in velocity there is a corresponding error in altitude. The very first integration step causes an error in velocity of 1 m s^{-1}. That error is then reduced to about 2 cm s^{-1} by the end of the integration. It cannot be said how an initial value problem such as this one initially diverges, and later converges. However, it is interesting to investigate the cause of the initial peak in error. As stated in 3.9.3, the equations using energy as independent variable are not completely compatible with a rotating planet. Thus, at first, it is investigated whether the exclusion of these two elements reduces the error of the equations. The resulting plot is almost completely equal, and is therefore not reported here. The cause of such large error is therefore to be found elsewhere.

A hypothesis consists of the fact that the initial energy step might be very large. The error for an RK method of order p, if all the first $p+1$ derivatives of $f(x,y)$ exist [Hairer et al., 1993], is bounded

[2] A spline interpolation cannot be very accurate. Nevertheless, it will be seen that the error in the propagation is much larger than the error caused by the use of interpolation.

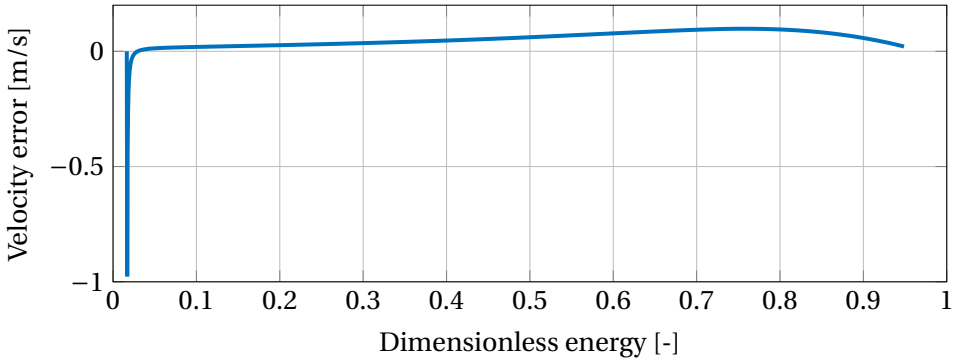


Figure 6.16: Error in velocity of spherical, dimensionless coordinates (with energy as independent variable). Integration step of 5.1793×10^{-4} dimensionless energy units.

by the following equation:

$$\left| y(x_0 + h) - y_1 \right| \leq K h^{(p+1)} \max_{0 \leq t \leq 1} y^{p+1}(x_0 + th), \tag{6.10}$$

where y^{p+1} is the p+1$^{\text{th}}$ derivative of the integrated function.

When using energy as independent variable, all the derivatives are divided by the drag, which is extremely low in the beginning of the simulation. This causes very large errors during the first steps. To prove this, two tests have been done: Figure 6.17 shows the error for a simulation with the same number of steps, in a span of 18 s only, neglecting rotation of the Earth and gravity due to J_2. It should be noted that the integration step in the energy domain has now become much smaller than ten times only: in fact, despite the time span has decreased by ten times, the energy span has decreased by much more than that, and the energy step is now about 500 times smaller. According to the theory of integrators, this should mean an improvement in the accuracy of about 10^{16} times. By inspection of the plots, one can notice that also in this case there is an initial tendency to an error, but with a much smaller order of magnitude. To properly understand how small that initial peak is, the reader should think that the oscillations in this plot are as large as the relative tolerance of the machine, which, using Matlab, is 2.2×10^{-16}. The peak of the error has been reduced of, instead of the theoretically predicted 10^{16} times, 10^{11} times, which is, however, satisfactory.

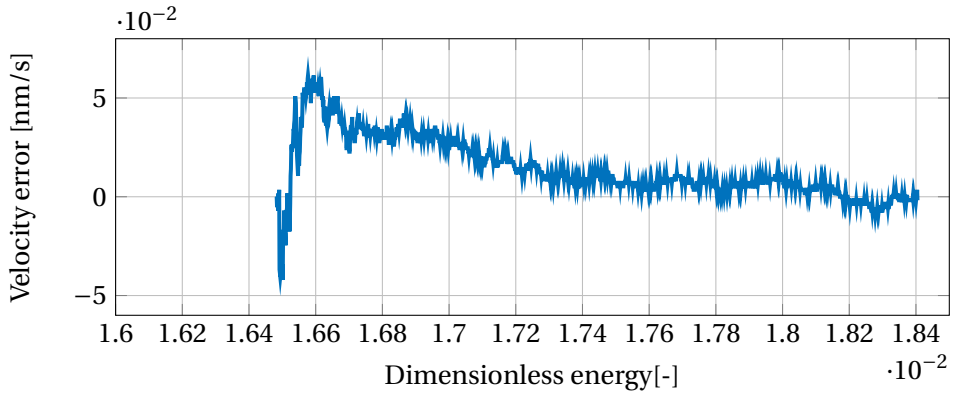


Figure 6.17: Error in velocity of spherical, dimensionless coordinates (with energy as independent variable). Integration step of 1.0968×10^{-6} dimensionless energy units. Integration for the first 18 seconds of the entry trajectory.

A situation like the previous one would however not happen in an NPC guidance: in fact, the logic is only triggered once the drag deceleration has a minimum value of 0.05 g. For this reason, a

second test is done, in which the simulation begins with a drag of 0.05 g, to check whether this set of equations is valid when used in the guidance. A comparable value of the drag is found at 91 km altitude and with a relative velocity of 11 km s^{-1}. This happens about 10 s after the beginning of the baseline simulation. For this reason, this test is started with the same conditions found after 10 s of integration of that simulation, and lasts 10 s less.

The error in the non-rotating case (plot on Figure 6.18) is much smaller than the one shown in Figure 6.16.The error peak is still about 8 cm s^{-1}. However, when the step-size is reduced, not much happens in this case: the deviation is then probably caused by the approximation $\dot{E} = -DV$.

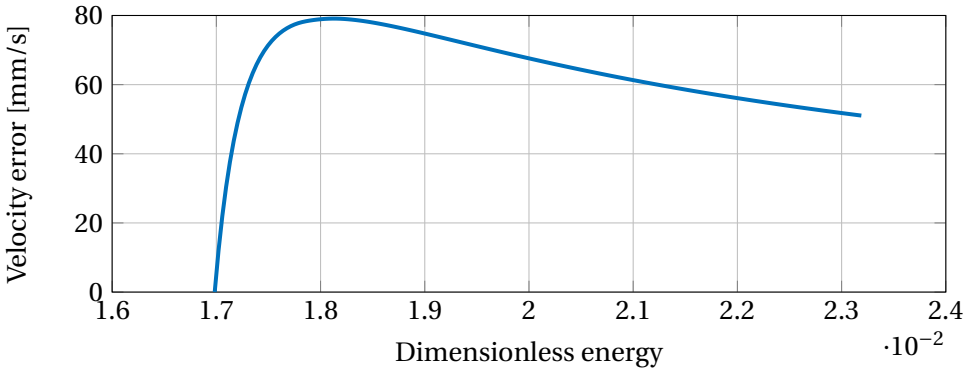


Figure 6.18: Error in velocity of spherical, dimensionless coordinates (with energy as independent variable). Integration step of 3.6500×10^{-4} dimensionless energy units. Integration for the 17 seconds following the first 10 seconds of the entry trajectory.

During aerocapture, one finds very low drag also during the exit phase. Depending on what value one sets as the exit altitude for the guidance integration, the final drag may be even much smaller than the one found in the initial part of this case. And if the drag is still large enough at that set altitude, then one would have a large error in the predicted apoapsis (that large drag would keep on acting on the spacecraft also after that altitude, and would be not accounted for, unless analytically estimated). It is therefore advised to not use energy as independent variable and uniform integration step in aerocapture guidance. In the next subsection, it will be evaluated whether the use of adaptive step is instead a better choice.

6.3.3 AEROCAPTURE CONVERGENCE

The convergence test has been done for an aerocapture mission. Such mission involves the same initial conditions as before, except for an initial velocity of 13 km s^{-1} and an initial flight path angle of -4.6934°. The commanded bank-angle is constant, and equal to 135°. The final apoapsis has an altitude of 500 km, if considering a Keplerian orbit starting from above 100 km altitude.

The same simulation has been run using uniform time-steps between 0.1 s and 20 s, with a resolution of, where possible, up to 0.1 s, and compared to the exact solution obtained as in the previous subsection. The mission lasts 511 s.

The convergence analysis obtained for the dimensional, spherical coordinates, can be seen in Figure 6.19.

The plot in Figure 6.19 is particularly useful because it can be used to fulfill the requirements for the integrators. Keeping in mind the fact that the curve is almost exactly the same for a Cartesian inertial propagator, considering that an error in apoapsis of around 10 m would be acceptable for the simulator, an integration with a time-step of 0.5 s is sufficient with an ode3, and a time-step of 1 s is sufficient with an ode5. The guidance usually has a frequency of 1 Hz: therefore, an order 3

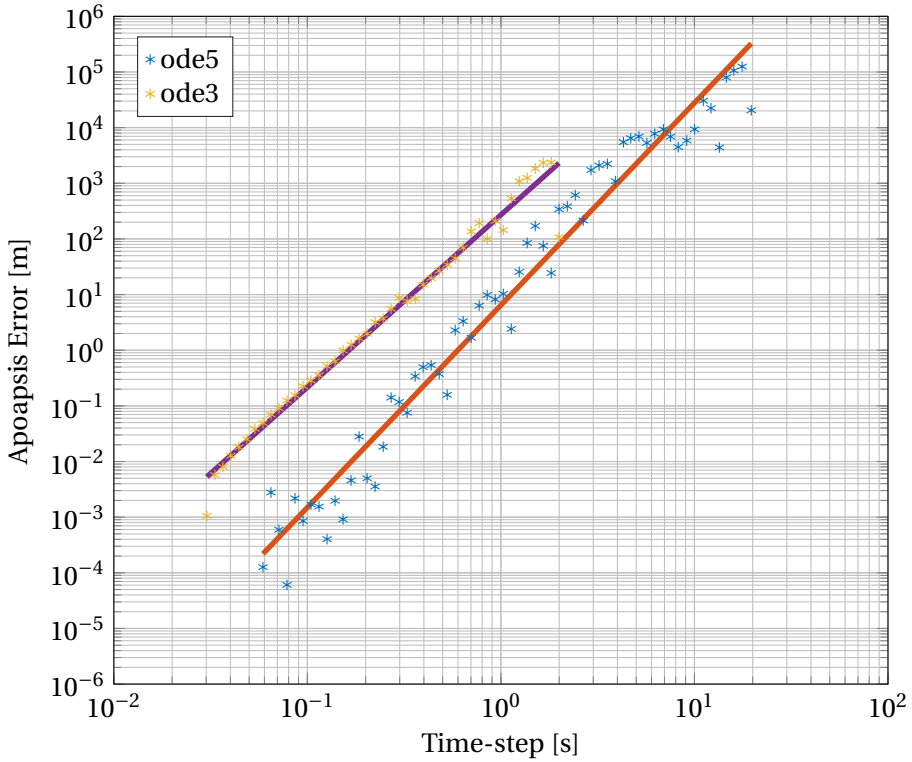


Figure 6.19: Error in final apoapsis as a function of integration time-step (integration in spherical coordinates).

with step-size 0.5 s is used.

Strangely enough, the convergence exponent[3] $p+1$ of Equation 6.10 is, for an order 5 method, quite similar to that of an order 3 (they are, respectively, 3.64 and 3.1). This convergence analysis was done also with RK1, RK2 and RK4. Integrators of order up to 3 have an error that decreases similarly according to Equation 6.10; such a behavior stops existing with solvers of higher order than that. A possible explanation for this phenomenon is discussed in Subsection 6.3.4.

For the inner loop, a lower accuracy is sufficient. Even an error on the entire trajectory of a few kilometers in apoapsis would be acceptable, because such an error in prediction would be corrected during flight. Moreover, the closer to the exit phase, the smaller the error in the prediction becomes; to have high accuracy at the end it is necessary to be accurate in the prediction mainly at the end of the atmospheric phase. Nevertheless, an error in prediction in the initial part of aerocapture makes the guidance less optimal.

6.3.4 ADAPTIVE ORDER AND STEP TRADE-OFF

In this subsection, the Matlab subroutines ode113, ode45 and ode23 are applied to the aerocapture problem, with both the use of dimensional and dimensionless variables. Figure 6.20 shows that none of the adaptive methods is capable to achieve an accuracy comparable to that of fixed step solvers. This may be explained by looking at Figure 6.21: it is in fact seen that a tolerance lower than 10^{-10} does not cause a further increase in number of time-steps. Moreover, the smallest time-step, when integrating at the highest allowed relative tolerance with ode45, is only 0.16 s, much larger than what is used to achieve high accuracy solution with a very similar non adaptive method (the Runge-Kutta of order 5). Also, the general behavior of the time-steps size is relatively strange, espe-

[3]The convergence exponents have been computed with linear regression of the logarithm of the step size and the logarithm of the error.

cially for the ode45 (both shown in in Figure 6.22, together with the ode23) and ode113 solvers, and might have something to do with interpolated data, which causes discontinuities in higher order derivatives. The latter statement may be further strengthened by the comparison of the time-step size with a use of the ode23 solver. The ode23 is an embedded Runge-Kutta method, specifically of the Bogacki-Shampine kind, which uses only three function evaluations per step. The time-step size for that solver follows a much less irregular path. Since this problem is thought to be happening because of aerodynamic data, a simulation with constant aerodynamic coefficients is run; however, the same pattern holds. By further analysis and visual comparison, it is found that the peaks in step-size of the ode45 and ode113 integrators occur at altitudes of approximately 71 km and 86 km, which are two points of discontinuity in the temperature gradient, which in turn causes a discontinuity in the scale-height, as explained in Subsection 3.6.1. The discontinuity does affect the ode23 solver as well, but much less. This is again probably because of Equation 6.10.

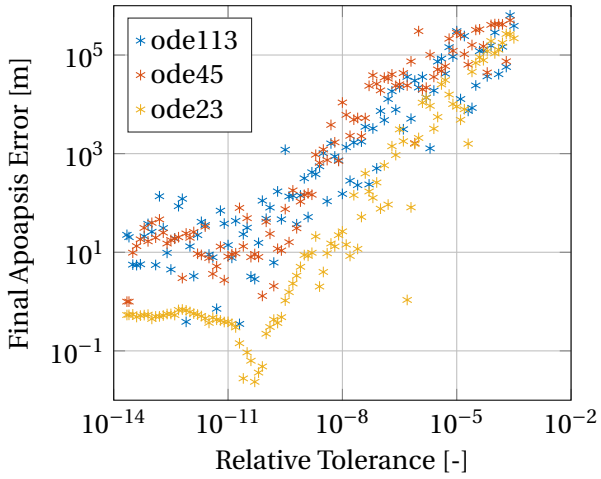


Figure 6.20: Comparison of apoapsis error as a function of relative tolerance settings, using dimensional spherical coordinates.

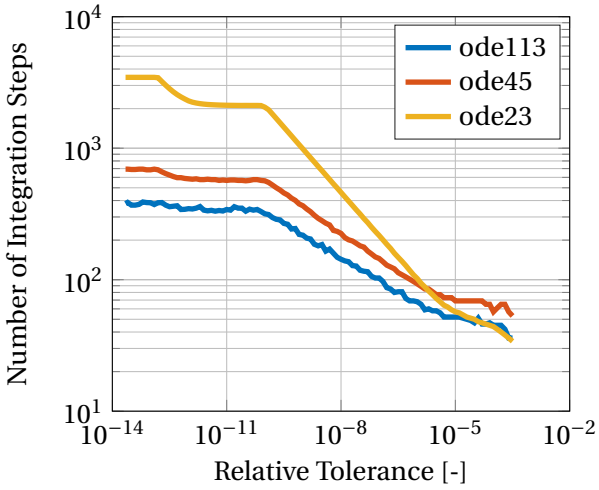


Figure 6.21: Comparison of number of steps required for similar relative tolerance setting when integrating aerocapture using dimensional spherical coordinates.

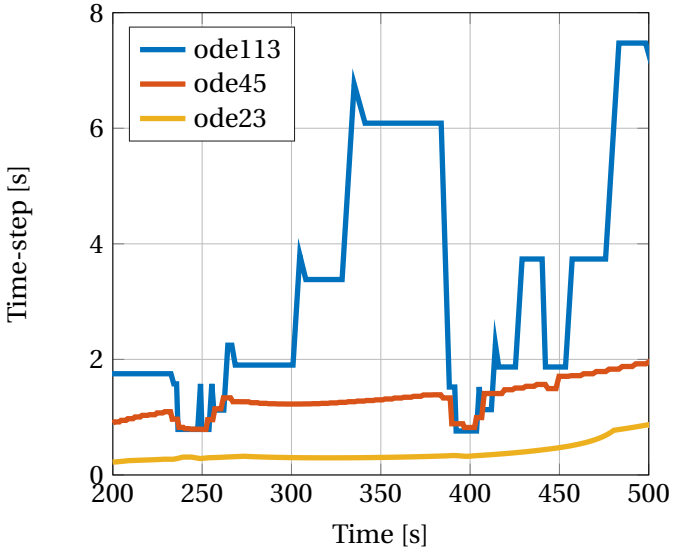


Figure 6.22: Stepsize of different integrators as a function of time.

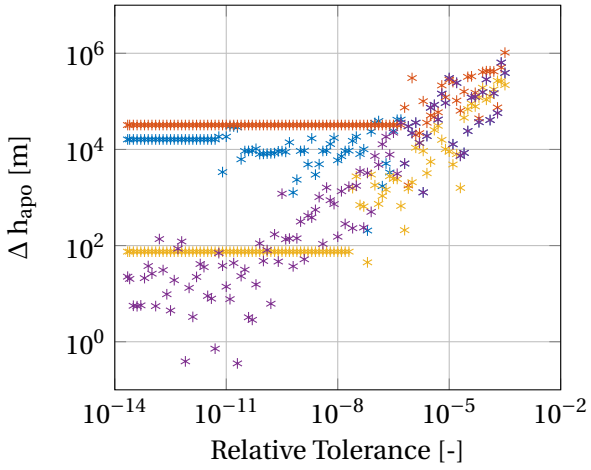


Figure 6.23: Error in apoapsis as a function of number of steps used for different solvers, with dimensional propagator ("d" stands for dimensional).

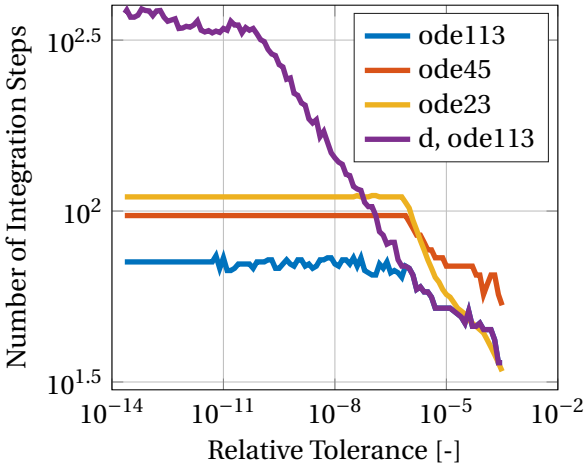


Figure 6.24: Error in apoapsis as a function of number of steps used for different solvers ("d" stands for dimensional).

Figure 6.20 and 6.21 should be considered when doing the trade-off. The parameter one has control over is the relative tolerance, and not the number of time-steps, nor the average time-step. Figure 6.20 shows that there is a large variance in accuracy with a certain relative tolerance. To ensure an accuracy of 1 km, one should use a tolerance of no larger than 10^{-10} with ode113; a tolerance of 10^{-7} is sufficient to achieve the same goal with ode23. Analyzing Figure 6.21, one can see that those values correspond to 300 time-steps for ode113, and only 200 for ode23 (which corresponds to 600 function evaluations). Hence, despite the larger variance of ode113, that solver is still the preferred one for the goal of this research. Comparing these results with the fixed step-size solvers, it is seen that the same accuracy is achieved with about 500 steps with a third order solver and with about 170 steps for the fifth order solver (which corresponds to 1020 function evaluations). Therefore, the adaptive solver is more convenient than a uniform solver if high accuracy is not required (for some reasons, most of these solvers seem to not be able to achieve a better accuracy than 200 m, except for ode23). It is required to have an accuracy of order of magnitude of about 10^2 m; also in this case, it turns out that ode113 is more efficient than both ode23 and ode45. Therefore, ode113 is chosen, and the relative tolerance is set to 10^{-10}.

The analysis of this subsection was referred to the dimensional equations, supposing that the same would hold for the dimensionless equations. However, it will be seen in the next subsection that, unfortunately, this is not the case.

To end this paragraph, it shall be reminded that this analysis is only preliminary, and one should be careful in generalizing these results to all aerocapture cases. Before doing that, a more thorough analysis should be done, which would be including different aerocapture scenarios, with different bank-angles, different initial conditions and target conditions.In brief, ode113 is thought to be the best choice for this work, but there is not a 100 % confidence in this choice, since it is based on the analysis of a single case, and there is nor the time nor the interest to have a more statistically relevant analysis of this for the current research. Moreover, also different atmospheric models may affect the choice: it is indeed quite likely that the use of a simple exponential model for the atmosphere would lead to a higher efficiency of the ode45 and ode113 solvers, whereas the use of an atmospheric model that includes, for example, the so-called small scale perturbations, which cause frequent discontinuities in the temperature gradient, would probably make the ode23 solver more efficient than the others.

6.3.5 NPC PROPAGATOR TRADE-OFF

Now that it has been decided that ode113 is the best integrator, it is interesting to evaluate which propagator works best with ode113. Even though each propagator includes different computations, it is believed that the difference in computational requirements between the three spherical propagators is negligible. In fact, the largest difference may occur when using energy as independent variable, and consists in the removal of one of the equations; nevertheless, the amount of transcendent functions remains the same. For this reason, the comparison in computational requirements will be, also in this case, based on number of function evaluations.

Despite the hope that the previous choice of integrator would have held also for the other propagators, it was immediately found that the situation is more problematic than expected. In fact, the use of ode113 in spherical dimensionless coordinates cannot ensure an accuracy higher than 10 km; the situation is even worse when using ode45, in which case the minimum error is in the order of magnitude of 30 km. Nevertheless, when using ode23, the accuracy improves to less than 100 m. Since all the solvers seem to get stuck in accuracy at an arbitrary point, another, slightly perturbed, mission is simulated (the one using constant coefficients of drag and lift); however, very similar results are found. It can then be expected that the point at which accuracy stops improving is not so arbitrary. It is also curious to notice how the dimensional and dimensionless solutions overlap exactly for tolerances of 10^{-6} or larger, both in error and number of steps.

With Figures 6.27 and 6.28, one can easily compare the efficiency of using the dimensional propagator with the ode113 solver (the best solver for dimensional propagator) with respect to using any other solver with the dimensionless propagator. In accordance with those it is hence chosen that the equations of motion for the NPC will be integrated using the ode23 in combination with the dimensionless propagator. This is because this choice provides very little variance (which means more robustness), high accuracy and low computational cost. The default setting will be a relative tolerance of 10^{-6}. An additional reason consists of the fact that ode113 shows (as ode45) large and sudden stepsize variations in proximity of discontinuities. Such variations also affect the computations.

6.3.6 SUMMARY

The conclusions below are made based on the integrators that are provided by Matlab. It could be possible that different integrators may be more efficient.

It is stressed that the results of this analysis should be limited to the point mass dynamics. When including attitude dynamics, it is likely that this analysis would not be valid anymore.

To summarize:

- High order integrators do not perform as expected because of discontinuities. This trend has been found to be true for RK4, RK5 and ode45. ode113 is an exception, possibly because it might be reducing the order of the integration in the proximity of the discontinuities. It could be possible to stop and restart the integration where the discontinuities occur. This though, requires a root search during integration, which also has its drawbacks in terms of computational time.A root search would be required because the location of the discontinuities is known in terms of altitude, but the independent variable is only either the time or energy. Thus, in proximity of the discontinuity, the integrator should also solve for $t\ h(t) = h_{disc}$, where h_{disc} is the altitude at which the discontinuity occurs. Because of the many discontinuities, this would happen many times during a single simulation. Consequently, RK3 with step-size of 0.5 s is chosen as default integrator for the simulator.

- With same number of time-steps, adaptive solvers are more accurate than uniform step-size solvers. The advantage is not very large, though.

- Using energy as independent variable is not advised for aerocapture. This is because the

derivatives become very large when propagating the motion in proximity of the atmospheric exit. This in turn causes large errors.

- ode113 combined with a dimensional propagator, and set relative tolerance, is the best solution, in terms of ratio between accuracy and number of time-steps. Nevertheless, it is affected by large time-step variations. These variations are caused by discontinuities in the atmosphere, and slow down the computational time. The variations are displayed in Figure 6.22.
- For the latter reason, ode23 in dimensionless coordinates is preferred to ode113 in dimensional coordinates.

As a last remark, it should be stated that the MatLab built-in adaptive solvers do not seem to improve their accuracy for relative tolerances smaller than 10^{-10}, despite the minimum allowed tolerance is 2.2×10^{-14}. They do not even seem to be trying, since the number of steps does not increase either. In some cases, this happens for even larger relative tolerances. It is possible that such a problem may not occur if setting an absolute tolerance.

6.4 NEURAL GUIDANCE

Figure 5.1 should be kept in mind while reding this section. The neural guidance has been designed in a way such that it would most resemble, in terms of inputs, the original guidance by Lu et al. [2015]. There, the inputs are the six state variables, together with the two additional filtered densities ρ_L and ρ_D. However, a few transformations are done to make the input supposedly more "intuitive" to the neural logic, but the filtered densities are instead used as such.

The only state variables that are kept the same are the relative velocity and altitude (in the NPC, the range and the latitude are indeed used to compute the altitude, despite this does not qualify as one of the six state variables). Of the remaining four state variables, the longitude is simply removed, since it does not have any effect on the propagation of the motion, nor on the final performance. The flight-path angle, azimuth and latitude are substituted by the eccentricity, the inclination, and the total orbital energy. This substitution might sound controversial, but without going into any mathematical proofs, it is sufficient to think that even though not explicitly, all these three Keplerian elements, together with altitude and relative velocity, are nonlinear functions of all the spherical state variables, except for the longitude.

Hence, no information is lost in this substitution. On the contrary, the information has become more significant: the three Keplerian elements define exactly the target orbit, and relative velocity, altitude, and the filtered densities define exactly not only the current aerodynamic force (and the gravitational one), but they can also be used to predict exactly how that force would evolve in time, if the real atmospheric profile were the US76 *c*.

This choice of variables differs from that of Gelly and Vernis [2009], and makes the set simply more robust, even though larger. In their work, in fact, they use the same three Keplerian elements, the relative velocity, and the sensed acceleration. This set has the advantage of being more compact, but is less robust. As an example, the neural guidance cannot distinguish between deviations in the lift or the drag coefficients. Also, in a non-spherical planet, it might have a hard time determining its inertial velocity or range. This is because it would somewhat estimate its range from the density (deduced from relative velocity and sensed acceleration): doing this would not take into account of the oblateness of the planet. However, what is missed here is the deviation from the average oblateness of the planet, and thus the error made would be much smaller than the 20 km of difference between equatorial and polar radius. While the latter is not a big drawback, not having a distinction between deviations in lift and drag might be problematic. Nevertheless, the results obtained in their work were very successful.

The choice of the outputs is instead different, depending on whether supervised or reinforcement learning is used. In the case of reinforcement learning, the same choice of as in Gelly and

Vernis [2009] is made. The bank-angle is computed from the outputs of the network μ_1 and μ_2:

$$\sigma_{cmd} = \text{atan2}\left(\mu_1, \mu_2\right). \tag{6.11}$$

However, in supervised learning, this does not seem an advisable choice, since the cosine of the bank-angle would have an error that comes from the error of two outputs. In some cases it would cancel out, but in other cases it would sum up, leading to large errors. Hence, the design choice is such that:

$$\cos\sigma_{cmd} = \mu_1; \tag{6.12}$$
$$\text{sign}\sin\sigma_{cmd} = \text{sign}\mu_1. \tag{6.13}$$

Of course, the command is then corrected such that the bank-angle is real.

As a last remark, the inputs to the neural networks should be normalized (this is not needed for the output instead, since the last layer is linear). Normalization is usually done in a way such that the inputs are as much as possible symmetrical around zero, and that have a standard distribution not larger than one. As an alternative, it is possible to have them such that they are uniformly distributed, with extrema smaller than 1.5 (even though this is not a strict boundary). Hence, the inputs are normalized as follows (the bar means that the variable is normalized; unless specified, the original variables are in SI units):

$$\bar{E} = 2\,E/V_c^2, \tag{6.14}$$
$$\bar{e} = e - .5, \tag{6.15}$$
$$\bar{i} = (i - i^\star)/5, \quad i \text{ in degrees}, \tag{6.16}$$
$$\bar{\rho}_D = 2(\rho_D - 1), \tag{6.17}$$
$$\bar{\rho}_L = 2(\rho_L - 1), \tag{6.18}$$
$$\bar{V}_{\mathcal{R}} = V_{\mathcal{R}}/V_c - 1.5, \tag{6.19}$$
$$\bar{h} = 3h/h_{ex} - 1.5. \tag{6.20}$$

Some variables end up not being symmetrical with respect to zero. One such example is the altitude, which never goes to 0 km, and also never reaches h_{ex}, because the guidance is shut down before that. Searching for the best normalization would an effort that would lead to marginal, if any, benefits.

At last, the neural guidance architecture can be summarized as the following list:

1. Obtain state and filtered densities.
2. Transform state.
3. Normalize input.
4. Compute output with the FFNN.
5. Compute output.

Every block of this sequence has been verified manually, except for the fourth, which was verified in Chapter 5.

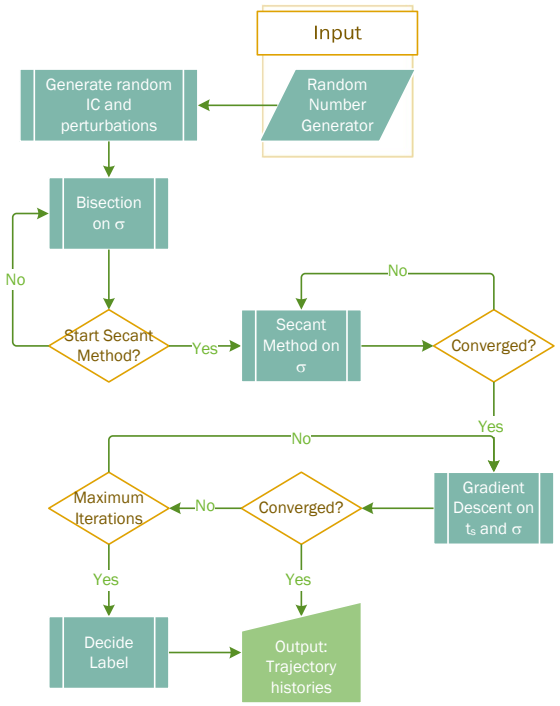


Figure 6.25: Flow chart for the architecture of the trajectory solver.

6.5 SUPERVISED LEARNING ARCHITECTURE

The supervised learning architecture consists of two main components. One is the trajectory solver, which generates the data. The other is the actual learning tool.

A trajectory solver has been chosen, instead of an optimizer, for three major reasons. The first is that the optimal solution is likely to be very similar to a bang bang trajectory: if the network were taught to follow that, the system would lose in robustness.

The second reason is that, if the optimal trajectory were not a bang bang (which, in the 3-dimensional space, is possible, even though it should not differ much from it), it would be necessary to either prove that the optimal trajectory is unique, or to set up the problem such that the solution would be optimal. The first case is very hard to do; the second case can be done by making the problem convex, and is also a very difficult task. A recent success to formulate a reentry problem as a convex optimization problem was done by Liu et al. [2016].

A third reason comes from the fact that it is sufficient to have a guidance that outputs the constant required bank-angle to obtain an optimal aerocapture guidance, comparable with the one by Lu et al. [2015]. In fact the following structure could be given to the guidance: as long as the output bank-angle were lower than a certain threshold, which we should call again σ_d, the guidance should give a command equal to σ_0; if, instead, the output were larger, a Phase 2 would be triggered,

and from then on the commanded bank-angle would always be equal to the one computed by the network. The careful reader might now see that this concept is exactly equivalent to the optimal NPC aerocapture. The only the difference lies in the fact that here the commanded output would be computed almost instantenously by the network.

The solver looks for the unique combination of constant bank-angle and reversal time that takes the spacecraft from the initial conditions to the desired orbit. For the solution to be unique, there needs to be a uniqueness in the choice of the initial sine of the bank-angle: this was chosen such that, initially, the inclination error would always increase. The initial bank-angle is therefore dependent on the initial conditions. At this point, it is clear that the attempt is that of replicating the *modlat* guidance.

The flow chart for the trajectory solver is given in Figure 6.25. At first, the cosine of the bank-angle that leads to the desired apoapsis is sought with te bisection-secant method. Then, an initial guess for the reversal time is given. Every time the reversal time is changed, the final apoapsis is also modified. Hence, at this point, the system in two equations

$$i(\cos\sigma, t_{rev} - i^\star) = 0, \tag{6.21}$$

$$r_{apo}(\cos\sigma, t_{rev}) - r^\star_{apo} = 0, \tag{6.22}$$

is solved. The method used is a modification of Newton-Raphson. In this case, the variables are updated one at a time, and the update is damped. This is done to avoid instabilities that would otherwise be encountered. The solver is not efficient at all, but does what it is meant to do. Once the trajectory is solved (or when the limit of iterations, set to 200, is exceeded), the solver gives a label l to the trajectory according to the following:

1. $l = 1$ if convergence is highly accurate;
2. $l = 0.5$ if limit of iterations is exceeded, but the trajectory is still accurate enough;
3. $l = 0.2$ if the cosine of the bank-angle is larger than one in absolute value;
4. $l = 0$ if none of the above conditions hold.

These labels are important during training. As an example, if $l = 0.2$, the data concerning the cosine might still be used in training, but the one concerning the sign of the sine would not. The perturbations in the solver are of larger magnitude than those generated for the Monte Carlo run (1.2 times exactly), and the atmosphere used is the US76 *c*.

Initially, there was the interest in using the US76 l also in the solver, such that, in theory, the network would have been able to learn the command including some part of the perturbations. This would have been very interesting, since it had the potential to make the neural guidance more robust than the NPC, even though it would in turn have made the dataset more noisy though. In the end, this idea was not pursued. Eventually, the perturbations are 20 % larger than those used in the Monte Carlo simulations, and are generated from a different random seed.

Concerning the learning architecture, there is not much to add with respect to what was written in Chapter 5. For completeness, the corresponding flow chart is provided in Figure 6.26.

6.5.1 VERIFICATION

The verification of the learning tool is mainly discussed in Chapter 5, where all the algorithms have been tested. The remaining blocks in the architecture have been verified manually.

The verification of the solver was also rather simple. In fact, the first part (the bisection-secant method) was already verified in Chapter 4. Instead, the second part, has been verified automatically by the labeling block. A trajectory that did not converge would by labeled as such.

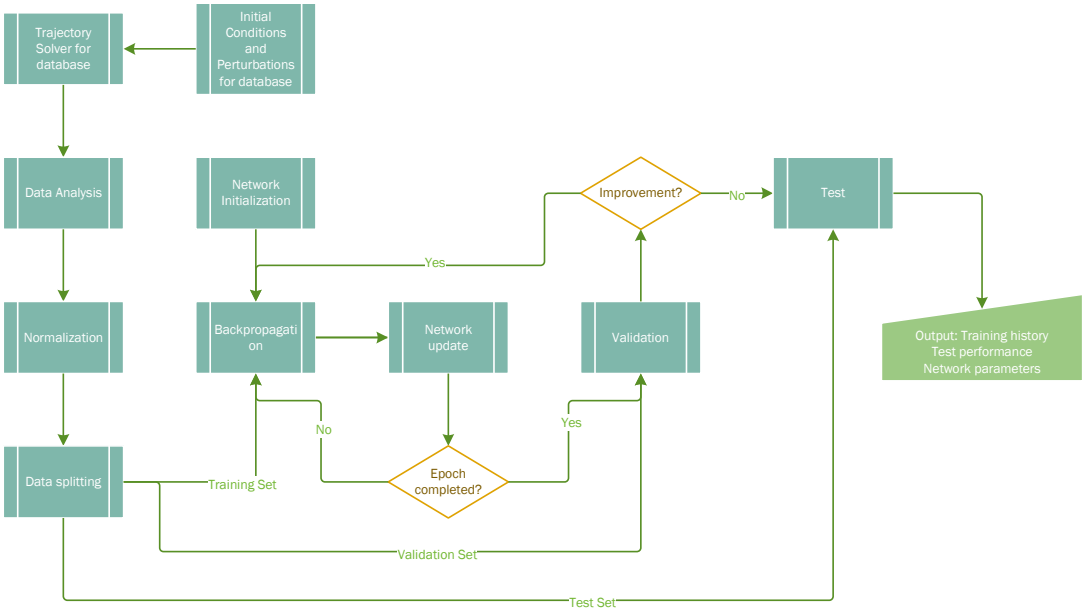


Figure 6.26: Flow chart for the architecture of the trajectory solver.

6.6 REINFORCEMENT LEARNING ARCHITECTURE

The main part of the architecture, which is the update algorithm, has been verified in Chapter 5.

The PGPE algorithm has the possibility of being used in different ways. The first consists of using it as a global optimizer, since it has such capability. In such case, one would do a setup equivalent to that of Gelly and Vernis [2009], but would substitute their evolutionary optimizer with the PGPE. Their work was very successful, but also computationally very demanding. Moreover, there is no real reason to believe that the PGPE would substantially improve their method (or improve it at all).

However, PGPE is also capable of giving an, even though very noisy, estimate of the gradient, with four evaluations of the function only. Hence, the problem could be seen as something very similar to supervised learning with SGD. The function to minimize would be, instead of the error of an input-output set, would be the peformance of the network with some given perturbations in initial conditions and atmosphere. The gradient would then be estimated, instead of using back-propagation, by PGPE. The update would then occur according to the currently estimate of the gradient, as well as to the set metaparameters.

As in SGD, the gradient obtained by a single sample is already a noisy estimate of the general function; moreover, in this case, the gradient of the sample is itself noisy, since it is obtained heuristically. For these reasons, the method, despite potentially very strong, may also be very challenging.

A flow chart of the reinforcement learning architecture can be seen in Figure 6.27.

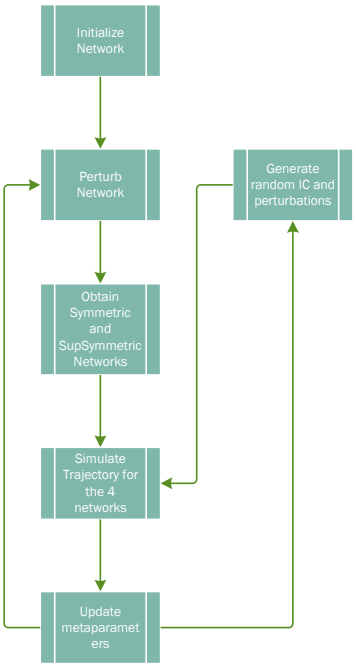


Figure 6.27: Flow chart for the reinforcement learning architecture.

As a last remark, the inputs used are the same as those for the supervised learning problem. However, the output is in this case the same as in Gelly and Vernis [2009], since it already proved to be successful for a reinforcement learning problem. In addition, a modification was included such that the bank-angle controller would always bring the spacecraft to the commanded bank-angle via the shortest path. As an example, if the current bank-angle were 90°, for a commanded bank-angle of −95° the controller would go to 265° instead.

As in the supervised learning paradigm, the perturbations are 20 % larger than those used in the Monte Carlo, and are generated from a different random seed. However, in this case, the baseline atmosphere used is the GRAM99 *l*.

6.6.1 CREDIT ASSIGNMENT

That of assigning a proper reward function is a one of the major problems of RL. As shown in Chapter 4, the trajectory minimizing the ΔV minimizes also load factor peak, heat-rate peak and integrated heat-load. This is very helpful, since the problem is automatically reduced from a multi-objective (minimum ΔV and minimum heat-load) constrained (load factor peak and heat-flux peak) to a single objective, unconstrained optimization.

However, if randomly initialized, a network would also lead to many crashes. It is therefore important to set up a credit assignment for those case as well. Giving simply a very high value, equal for any crash, is not a good solution, since the PGPE uses gradient information. Hence, it is necessary to distinguish between the "quality" of different crashes.

The conditions according to which the simulator stops because the crash has become inevitable are the same each time. Given this, one may compute the time after which the crash happens. If one repeats the trajectory with the same conditions, and every time changes the network such that the crash happens a little later, at some point the crash should not happen anymore, because the point of discontinuity between crash and success would be reached. Hence, following the opposite the gradient of the time would be a good way to "get out" of the crashing zone. This reasoning is, however, just intuitive, and it is possible that, unless otherwise proven, following the direction of increasing time may lead to a local maximum, in which the trajectory is still crashing. However, this method is further strengthened by analyzing the duration of flight for a trajectory with constant bank-angle, which, before skipout is reached, is constantly increasing, as shown in Figure 6.28.

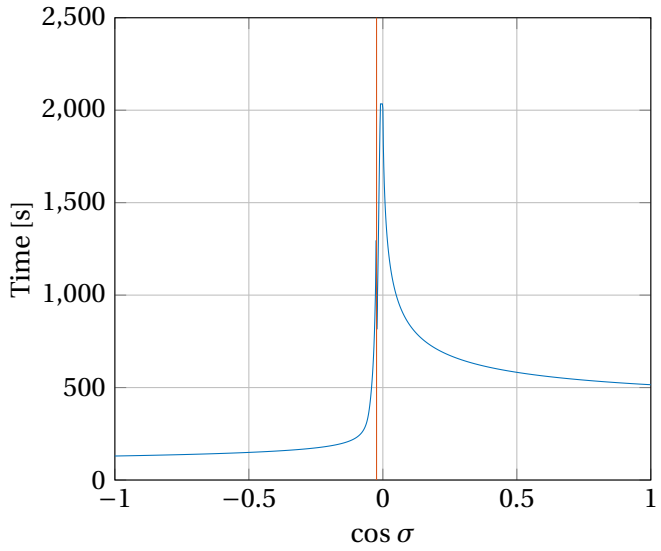


Figure 6.28: Duration of aerocapture as a function of cosine of the bank-angle, for the same conditions as in Figure 4.11.

From this, two different credit assignment functions were proposed (keeping in mind that the PGPE maximizes the fitness function):

$$f_1 = \begin{cases} \frac{1,000}{t_{end}} & \text{if crash} \\ \frac{1,000}{\Delta V_{tot}} & \text{otherwise} \end{cases} \tag{6.23}$$

$$f_2 = \begin{cases} 700 - t_{end} & \text{if crash} \\ 3,000 - \Delta V_{tot} & \text{otherwise} \end{cases} \tag{6.24}$$

Unless later specified, with ΔV_{tot} it is meant the ΔV including the out-of-plane component.

Both these functions have strong discontinuities, which seem to be inevitable. This might or not be a problem. For sure, it is well known that having discontinuities in the proximity of the optimal point causes major issues in any gradient based optimizer.

6.6.2 VERIFICATION

The algorithm has already been verified in Chapter 5. However, here, its application to the RL problem is verified. The verification of this concept consisted of an aerocapture trajectory optimization.

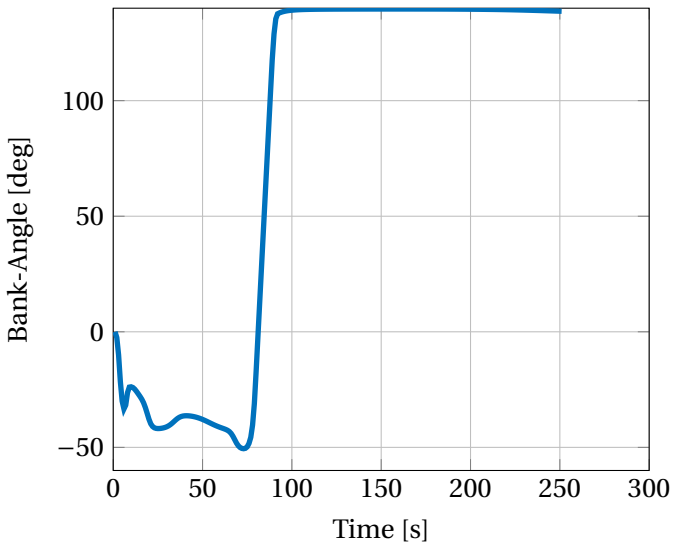


Figure 6.29: Bank-angle history for the trajectory optimized by the PGPE.

The optimization was done using a neural network: simply stated, the same network that would later be used in the reinforcement learning, is here used as a parametric guidance for a single trajectory. This verification is useful to understand whether the two functions f_1 and f_2 are suitable to the problem, together with verifying whether the entire architecture is properly functioning.

The verification has been done for the same conditions, with three sets of different metaparameters.

The best results have been obtained with $\alpha_\mu = 0.5$, $\alpha_\sigma = 0.1$ and σ_0 equal to 2. The trajectory was optimized with 5,000 iterations, which equals to a total of 20,000 simulated trajectories. However, nothing can be said about how these parameters would affect the learning in a reinforcement learning paradigm. Figure 6.29 shows the quasi bang-bang control history obtained, where the bank reversal and the shift from Phase 1 to Phase 2 (even though those phases are here not explicitly set, but found by the optimizer) happen suddenly and concurrently, while moving from −50° to 150°. The trajectory hence combine the two major shifts in bank-angle in a single one. Eventually, this leads to a ΔV of about 83 $\mathrm{m\,s^{-1}}$ (the value includes the lateral burn); in similar conditions, the *mod* σ_d =120° achieved a ΔV around 3 $\mathrm{m\,s^{-1}}$ smaller. The ΔV of the trajectory obtained here includes also the lateral component though. The trajectory obtained with this method is less optimal than what could theoretically be achieved with an instantaneous bang bang trajectory. Nonetheless, three considerations are worth mentioning:

- The constraints on attitude dynamics are enforce during this simulation. Hence, an instantaneous shift from $\sigma = 0°$ to $\sigma = 180°$ would not be possible.
- The optimized trajectory minimizes the full ΔV, which includes the lateral component. Thus, a problem more complex than the optimization of the planar ΔV is being solved.
- The trajectory is flown in the GRAM-99 *l* atmosphere.

For these reasons, it could be interesting to further investigate how PGPE can be used as a trajectory optimizer, in combination with an FFNN, but also with other kinds of parameterization.

In conclusion, this method can be considered verified. It is very promising, even though the challenges are evident.

7

RESULTS

This chapter presents the results obtained while training and testing the networks with the two different methods described in Chapter 5, supervised learning and reinforcement learning. Respectively, their training setup has been described in Sections 6.5 and 6.6. The test setup was described in Section 6.2.

Section 7.1 describes in detail the results obtained with the supervised learning setup. After many attempts and lessons learned, some successful results, as defined in Section 2.7 were obtained. In Subsection 7.1.5, partial conclusions concerning this setup are drawn.

Section 7.2 is much shorter, and discusses the results obtained with the reinforcement learning architecture. The training process is described, and some initially obtained results are presented. Nonetheless, it is clear that a large problem arises.

The networks have been trained extensively and have been tested in different environments. Throughout this chapter, the reader should keep in mind the fact that, from a high-level point of view, this problem is extremely large. Possible variables in the supervised learning problem are:

1. Topology of the network.
2. Distribution of the perturbations in the data set.
3. Size of the data set.
4. Bank reversal speed (finite or infinite) in the data set.
5. Use of one or two networks for sine and cosine of the bank-angle.
6. Training algorithm used.
7. Choice of the input set.
8. Splitting of the data into training, validation, and testing set.

Among these variables, the only one that requires further explanation is the last one. The validation and test data sets are usually picked randomly from the full data set. However, in this case, the data set is sampled from a certain number of trajectory. Hence, it might seem that splitting the data set into training trajectories and validation trajectories might make more sense. In fact, it might be possible that the network overfit the trajectories, in the sense that it would learn how to generalize along a trajectory, but not between one trajectory and the other.

Possible variables in the reinforcement learning problem are:

1. Topology of the network.

2. Distribution of the perturbations in the training simulations.
3. Choice of the input set.
4. Credit assignment function.
5. Metaparameters settings.

Moreover, in case difficulties were encountered, an additional option could consist of reducing the difficulty of the problem, by excluding the lateral guidance.

Therefore, the reader should not expect that all the possible variations will be explored.

7.1 SUPERVISED LEARNING NETWORKS

The results of the supervised learning can be divided into three stages. From one stage to the other, the variables previously mentioned have been changed, according to some of the lessons learned from the precedent stage.

Different network topologies were explored. The three main options were a small, single hidden layer network with only 25 hidden neurons (labeled SN), a larger one with 4 hidden layers of 16 hidden neurons each (labeled 4DN), and a very large one with 6 hidden layers of 128 neurons each (labeled VLN). Repsectively, each network has 226, 961, and 83,713 parameters. The first two networks were trained with LM, which is better performing than any other method, if networks and dataset are not too large. For the second network, however, the dataset always had to be reduced to a maximum of 200,000 (which means 140,000 in the training set, since the remaining 60,000 would make the validation and testing sets).

The choice of the networks' size was rather simple: a small network could use the LM algorithm and a large dataset, a medium size network could also use LM, but would require a smaller dataset; however, it would allow more complex functions to be reproduced. A large network would instead require SGD, but would allow for much more complex functions to be reproduced. The choice of a deep network in contrast to a single hidden layer network for the medium size network comes from the results obtained by Sanchez-Sanchez et al. [2016], who showed that, for equal number of parameters, a deep network is a better guidance solution (for terminal landing, in this case) than a shallow one.

Concerning the choice of the network, one may think that the one used by Gelly and Vernis [2009] could be a good solution. Their network had 12 hidden neurons and one single layer only, and their output layer was nonlinear. However, the formulation of the problem in terms of RL is very different from when the problem is in terms of supervised learning. In fact, an additional problem of shallow layers is that of finding being stuck in local optima; as stated in Section 5.6, such problem does not really occur in deep networks, and of course does not happen for shallow networks if evolutionary optimization is used, as in Gelly and Vernis [2009]. A proof of this is the fact that Sanchez-Sanchez et al. [2016] also showed that larger networks outperform shallower ones.

7.1.1 STAGE 1

In the first stage, a data set of 5,000 trajectories was used, in which the bank reversal time was finite, and equal to the one used in the prediction of the *modlat* guidance ($10.5\,°\,s^{-1}$). The trajectories were dispersed in the exact same way as described in Section 6.5. The total number of data points was 536,639.

The first problems occurred already with the search for the switching time. Being the reversal time finite, it would affect the trajectory consistently. Hence, there would be an additional instability in the solver when the bank-angle were close to 90°: in fact, the reversals were set to pass by the 0° or 180° angle depending on which the bank-angle were closest. Hence, it was necessary to block the reversal direction during the solving. This lead to a non-uniqueness of the solution, since some

trajectories with bank-angle below 90° ended up having a reversal upwards, and vice versa. It was initially believed that this would not cause any noticeable problems.

The concept developed in Stage 1 is not comparable to any of the guidance systems from Chapter 4. In some sense, it is similar to the *simple* guidance; however, it does include the lateral logic. Moreover, in this case, the bank-angle is commanded in a way such that the reversal, is already included in the prediction, at the right moment, and in its whole duration. Another way of seeing this is thinking of a *modlat* guidance, in which however the logic goes to Phase 2 since the beginning, and in which at each iteration both the bank-angle and the reversal time are iterated. Hence, it solves the trajectory as a function of two variables, as explained in Section 6.5. Such a solution is computationally much more demanding than any of those obtained by the NPC, and is therefore even less likely to be possible in real-time: this fact justifies even further the use of ANNs. If working, this would have a much more robust lateral guidance than the *modlat*. It could then be made bang-bang by setting $\sigma_{cmd} = \sigma_0$ as long as $\sigma_{cmd} < \sigma_d$.

In this part of the research, any unfeasible trajectory would still be labeled as non-converging, and hence the network would not learn it. The states during the reversal were given the sign of the sine as it would be after the reversal: this is because the lateral logic starts the reversal when the output changes sign.

The regularization parameter was always kept equal to 10^{-5}. The very first trainings ended up stopping very early, with an RMS of[1] about 0.11. It was easy to find out that the main problem was how the output for the sine was set: the fact that it had to suddenly change value from 1 to -1, or vice versa, caused large errors in the proximity of the shift, which in turn led the learning to early stop. Hence, it was decided to split the problem in two: one network would learn the cosine, and another one would learn the sign of the sine.

At this point, the training was done only for the medium and small size networks: since already some troubles were encountered, as will be shown below, Stage 1 was interrupted before training the 6DN.

TRAINING

After all the issues mentioned above, the training for the SN and the 4DN turned out to be rather successful, even though very different from each other. During training, the performance of the network is evaluated in terms of root means square (RMS) error with respect to the validation set.

One can see from Figure 7.1 that there is a large difference between the accuracy obtained by SN and by 4DN, as well as there is between the accuracy of the vertical and the longitudinal guidance.

The vertical guidance reach an RMS of 0.017 in the case of the SN, and 7.8×10^{-4} for the 4DN. These mean, supposing all the errors were equal in magnitude, an average error in the cosine of the bank-angle of, respectively, 0.132 and 0.028, or 6.6 % and 1.4 %. These imply some rather large errors; however, reminding the large variations in guidance commands that were found in Chapter 4, which were caused by the atmospheric perturbations, such a deviation does not seem too bad. Hence, these networks will be tested in a Monte Carlo run. The performance of the 4DN was checked also on the data samples outside of its reduced training set: the overall RMS turned out to be 0.001, only 20 % worse than the validation set.

Also of interest is to check how the guidance reproduces the dataset. In Figure 7.2, it is seen how the two guidance concepts show the last 1,000 samples of the dataset, which amount to around 9 trajectories. An oscillatory behavior occurs, and inaccuracies are large for the SN, especially at the beginning of each trajectory. Nevertheless, the DN4 seems to be performing very well, as expected from the previously analyzed data.

The case of the lateral guidance is, at first sight, much more critical. However, evaluating the lateral guidance from its RMS error is unfair, since it only has to indicate the correct sign. A more

[1] For the entirety of this chapter, by RMS it is meant its squared value, or the mean of the square errors.

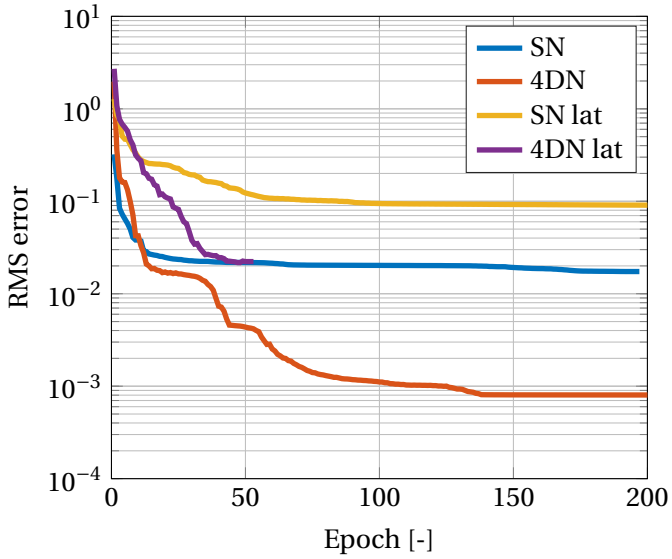


Figure 7.1: Performance of the networks on the validation sets during learning.

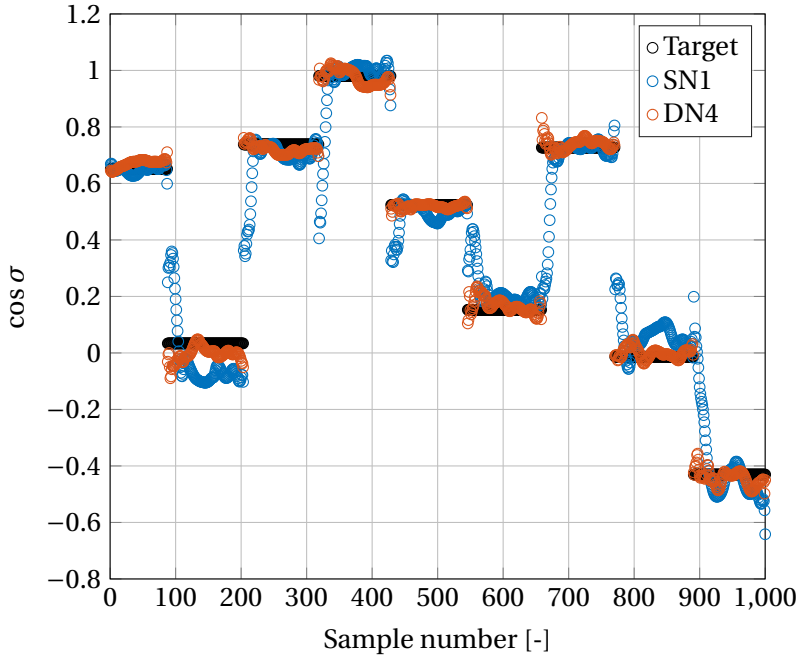


Figure 7.2: Targets and networks outputs for the last 1,000 samples of the dataset.

fair way to evaluate its performance consists of checking how often the guidance has a different sign from what it is supposed to have. This check has been done for both networks, for all the points in the dataset (this means that the 4DN network has been tested for points outside its training data as well): the SN ended up being correct 98.56 % of the times, whereas the 4DN was correct 99.09 % of the times. Hence, both lateral logics can be considered to be properly trained.

Considering the not very large difference between the two lateral logics, it was decided to only use the one from the 4DN, independently from the network that would be used in the vertical guidance. This same lateral guidance would be kept also during Stage 2.

TESTING

The neural guidance logics were tested in the same Monte Carlo runs described in Section 4.8, but for the US76 *c* and the GRAM-99 *ls* environments only. The choice was made because the US76 *c* is the same environment used in the solver, and hence is useful in understanding the behavior of the networks, whereas the GRAM-99 *ls* is the most challenging atmosphere.

Figure 7.3 shows the longitudinal performance of the neural logics in the US76 *c* environments, whereas Figure 7.4 shows their performance in the GRAM-99 *ls*. In both environments, and for the 4DN only, there is a cloud of about 4 % of the total samples that has been cut from the figure, lying at altitudes between 1,000 and 3,000 km, for flight-path angles between -5.35° and -5.15° in the GRAM-99 *ls* and between -5.50° and -5.05°. In all four cases there is a clear line in which the guidance is relatively accurate in apoapsis (even though still many orders of magnitude less than any of the NPC guidance systems from Chapter 4), which goes from an initial flight-path angle of −6.4° to about −5.5°. The same outliers found in Chapter 4 are also there, even though not very clearly visible. The corresponding ΔV follows a curve similar to that of the *simple* guidance, as expected. In that range of entry angles, in the GRAM-99 *ls* atmosphere, the SN has an average error of 20.5 km in apoapsis and an average ΔV of 247.2 m s^{-1}. In that same range, the 4DN has an average error of 19.0 km and a ΔV of 243.7 m s^{-1}. In the US76 *c* the 4DN turns out instead to be a little more accurate, with an average error of 16.5 km in apoapsis altitude against the average 20.7 km scored by the SN (always in that same range of entry angles).

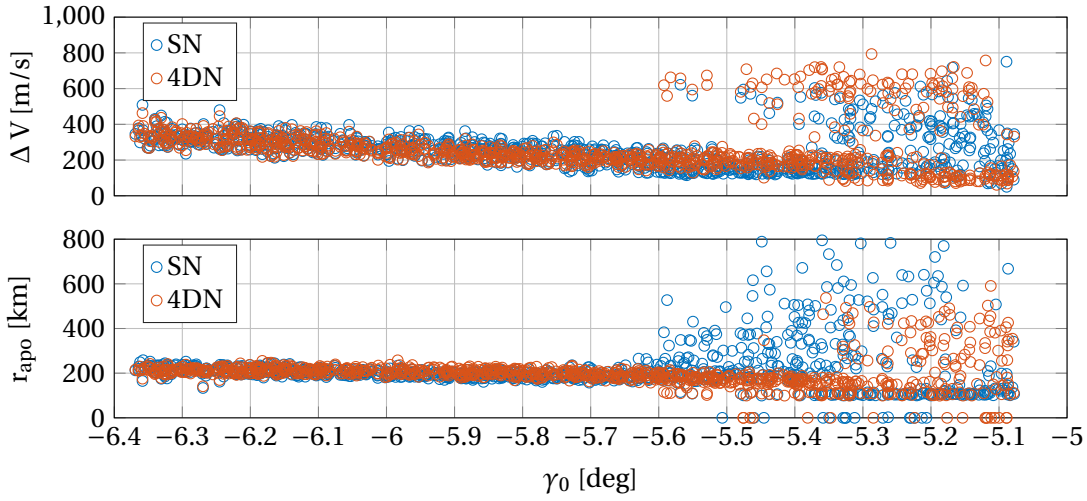


Figure 7.3: Performance of the networks in the US76 *c* environment.

In addition to the main line there are however at least three recognisable patterns in each case: a cloud of points, spread for a large range of low shallow flight-path angles, with very high apoapsis (including the cloud for the 4DN above mentioned, and not shown in the plots), a line of trajectories with very low apoapsis (all between 100 and 120 km altitude) for shallow entry angles, and a few crashes, also occurring for shallow entries. It is rather counterintuitive that a shallow entry may lead to a low apoapsis, or even to a crash. These patterns (excluding the crashes) cause the cloud of high ΔVs for shallow entry angles that occur in every case. Despite not being very intuitive, Figure 7.5 helps visualize this correlation.

Given the strange distribution of the performance, a table summarizing the statistics of the network would be of no use.

If one were to notice a difference in the performances of the two networks, it would be the fact that when entry angles become shallow, they start deviating in opposite directions: the 4DN has a low apoapsis for entries between −5.4° and −5.3° (in the GRAM-99 *ls*), and then has high apoapsis;

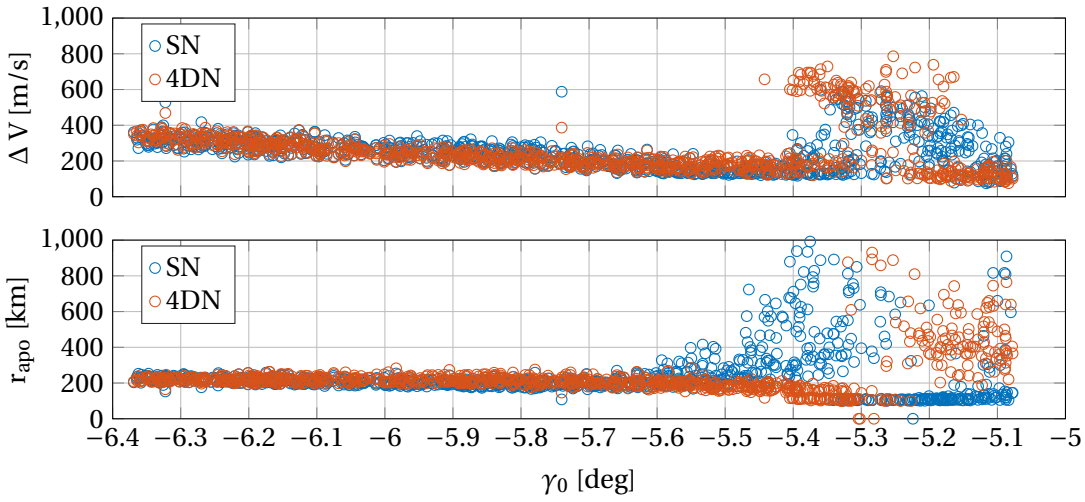


Figure 7.4: Performance of the networks in the GRAM-99 *ls* environment.

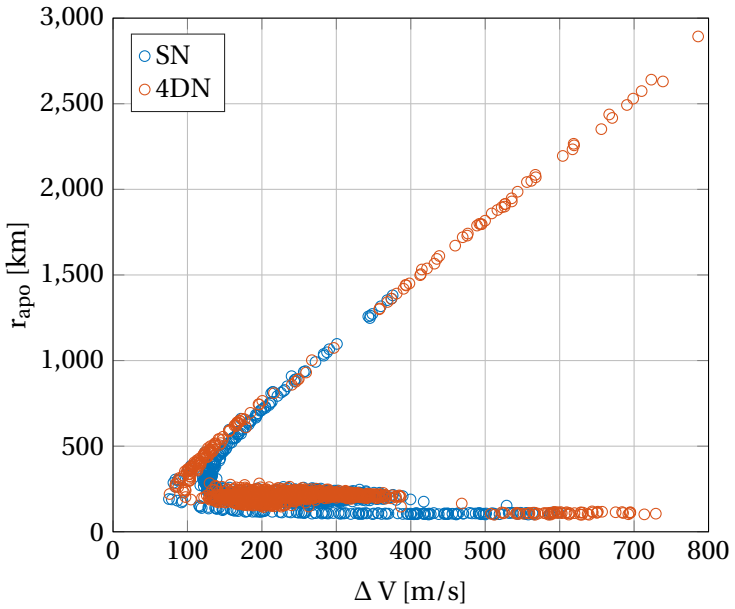


Figure 7.5: Correlation between ΔV and apoapsis for both networks in the GRAM-99 *ls* environment.

the SN, instead, starts having high apoapsis at entries of around −5.6°, and then has low apoapsis for entry angles shallower than −5.4°.

The performance of the lateral guidance is worth being mentioned. As the longitudinal guidance, also the lateral guidance becomes inaccurate for shallow entries. However, it becomes very inaccurate also for very steep entries. This makes sense, thoguh,since in those cases the bank-angle is very close to 0°. Figure 7.6 (left) shows this pattern.

Eventually, the behavior of the network along specific trajectories is analyzed next.

REQUIREMENTS ASSESSMENT

Because of what was concluded in Chapter 4, the only parameter to be analyzed to meet the requirement will be the ΔV. In terms of the requirement GD-1 of Section 2.7, the SN and the 4DN perform equivalently well. For both networks, and both environments tested, there is a very wide range of

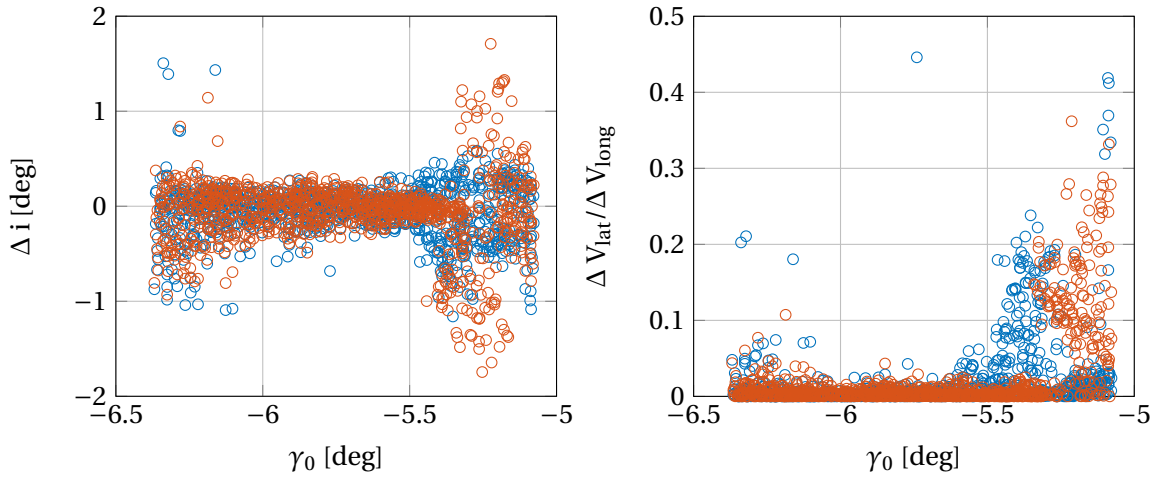


Figure 7.6: Performance of the lateral guidance: final inclination error (left) and ratio between lateral and longitudinal ΔV (right).

entry angles for which the ΔV is bounded with 3 standard deviations. Such value is 450 $\mathrm{m\,s^{-1}}$, and it can be ensured for entry angles between -6.4° and -5.6°. Within such a range, and for entry angles between -6.1° and -5.6°, the lateral guidance gives a lateral ΔV of maximum 5 % of the planar ΔV. Hence, for the latter range, the requirement GD-1 is met with a total ΔV_{max} of 475 $\mathrm{m\,s^{-1}}$. Such a performance is still very far from that of the concepts of Chapter 4. Thus, further effort will be made in the next subsections to lower the ΔV_{max}. Before that, an additional analysis of the specific behavior of the guidance is carried out.

BEHAVIOR

This subsection analyzes the specific behavior of the neural guidance. Such an analysis is important to give further insight in the problem, and find possible solutions to the problems encountered.

Figures 7.7 and 7.8 show the bank-angle history for the two networks in both the US76 *c* and the GRAM-99 *ls* environments; respectively, they show a relatively steep entry ($\gamma_0 = -5.991°$), and a shallower one ($\gamma_0 = -5.374°$). Unfortunately, there is no guidance concept from Chapter 5 that is similar to the one reproduced by these networks, which is a hybrid between the *simple* and the *modlat*. This is meant in the sense that it does not perform a bang-bang trajectory, but it keeps into account the future reversal. As a reference, the guidance *simple* flew the trajectories with an almost constant bank-angle of, respectively, 47.5° and 83.0°. The trajectories taken here shall be considered as two examples only.

Figure 7.7 shows that, for the same initial conditions, the SN network gives a much noisier outputs than the 4DN. In addition, the initial error seen from the training samples in Figure 7.2 is here very evident, and causes an initial bump of about 20°. The behavior of the 4DN is particularly good instead. It is indeed almost suspicious that it can keep a constant bank-angle during the entirety of the trajectory in the GRAM-99 *ls* environment, especially after the reversal. If one remembers Figure 4.37, in which the trajectories obtained with the *Lu* guidance in the GRAM-99 *ls* environment were extremely noisy, especially towards the end, it is evident that this linearity may even look suspicious. In addition, while doing so, it aims the apoapsis with an error of 800 m only, as shown in Table 7.1.

Figure 7.8 shows that in the shallow entry case, the solutions obtained by the SN and the 4DN networks diverge from each other after the reversal. In addition, in the case of the GRAM-99 *ls*, there occurs a problem strictly related to the one found in the solver. In fact, the network receives the command to reverse when the bank-angle is slightly below 90°, and thus turns in the opposite direction from the others guidance systems. This, in turn, leads to a completely different trajec-

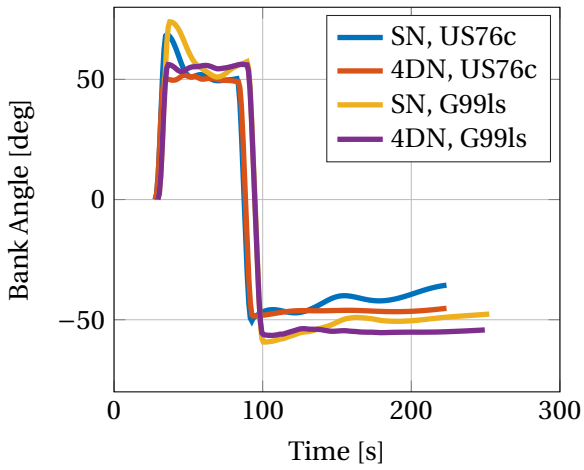


Figure 7.7: Bank-angle history for the two networks, in two different atmospheres.

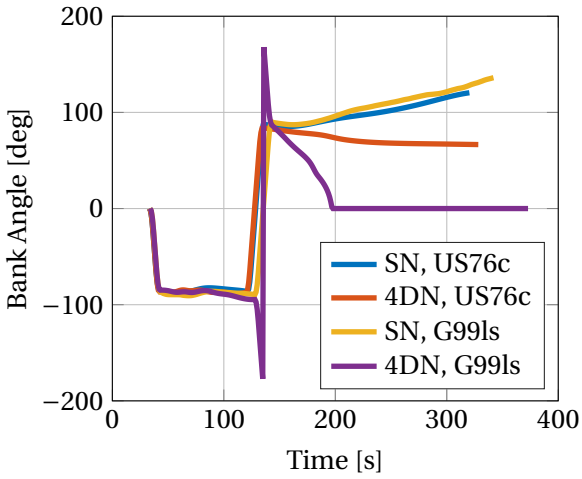


Figure 7.8: Bank-angle history for the two networks, in two different atmospheres.

tory. Moreover, it seems likely that the network had "expected" that the turn should have been upwards. In fact, when the guidance turns upwards (in the US76 *c* environment), the bank-angle after the reversal remains quite the same (at least immediately after the reversal); instead, when it turns downwards, it diverges much faster. This plot shows two clear drawbacks of both logics: the first consists of diverging with a low flight-path angle; the second is the problem of the ambiguity of the bank-angle reversal, which, if different from what expected, may lead to undesired results. In fact, in that case, the final apoapsis turns out to be 98 km below the target. To facilitate the comparison, Tables 7.1 shows the performance of the guidance systems, together with the comparison of the *Lu*, the *modlat*, and the *simple* for the same conditions.

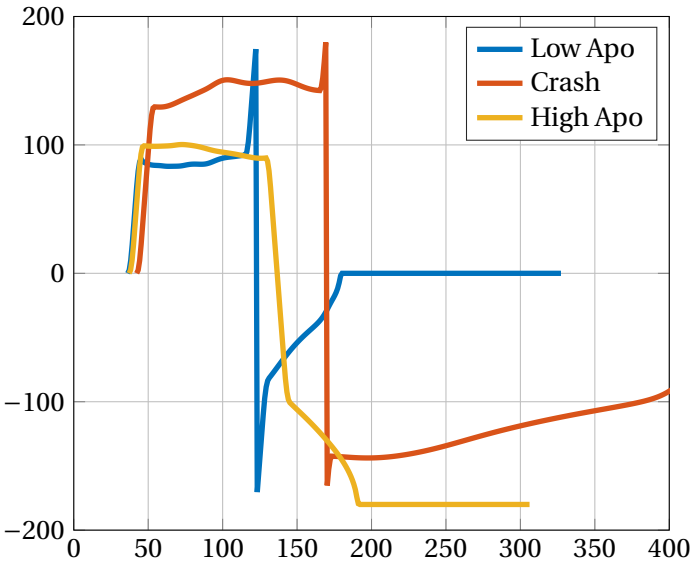


Figure 7.9: Examples from each of the three different possible failure modes, obtained with the 4DN, in the US76 *c*.

Eventually, it is interesting to see how some samples from the three failing patterns behave, to look for possible solutions. Figure 7.9 shows three trajectories that can help understand what happens in each of the three cases. It is clear that what causes the difference between a low apoapsis failure and a high apoapsis failure is the commanded bank-angle when the reversal is triggered, and

Table 7.1: Performance of the trajectories from Figures 7.7 and 7.8.

		Fig. 7.7 ($\gamma_0 = -5.99°$)				Fig. 7.8 ($\gamma_0 = -5.37°$)			
Guidance	Atmo	r_{apo} [km]	ΔV [m s^{-1}]	ΔV_{tot} [m s^{-1}]	Δi [deg]	r_{apo} [km]	ΔV [m s^{-1}]	ΔV_{tot} [m s^{-1}]	Δi [deg]
SN	US76 *c*	191.2	319.0	319.5	-0.14	292.3	121.9	124.9	0.18
	GRAM-99 *ls*	187.8	271.8	271.9	-0.07	329.7	120.4	130.4	0.32
4DN	US76 *c*	200.5	293.1	294.4	-0.21	147.7	226.4	227.7	-0.17
	GRAM-99 *ls*	201.9	247.2	247.5	-0.10	102.5	692.3	693.0	-0.23
simple	US76 *c*	200.0	284.2	-	-	200.0	122.5	-	-
	GRAM-99 *ls*	199.8	243.6	-	-	198.3	109.1	-	-
Lu	US76 *c*	200.0	83.7	-	-	200.0	72.9	-	-
	GRAM-99 *ls*	199,6	82.0	-	-	200.0	74.9	-	-
modlat σ_d =120°	US76 *c*	200.0	76.1	76.9	-0.08	200.0	51.1	51.1	-0.00
	GRAM-99 *ls*	199.2	71.7	72.1	-0.06	200.0	49.9	49.9	-0.00

the consequent direction of the reversal. The crashing case is instead a very shallow entry ($\gamma_0 = -5.106°$). In that case, it is less likely that the cause of the deviation from the correct trajectory is the reversal. An indication of what the problem could be can however be found by analyzing the training data set. Figure 7.10 shows a histogram of the outputs of $\cos\sigma$. It is seen that trajectories with very low $\cos\sigma$ are particularly rare. Hence, the networks are not well trained at guiding the vehicle when they are supposed to give a command with $\cos\sigma$ close to -1. In addition, trajectories flying close to such conditions are already inherently more unstable, as stated in Subsection 2.5.2. Moreover, this reason would automatically explain also why there are less crashes in the GRAM-99 *ls* environment: trajectories with $\cos\sigma \approx -1$, are correlated to not only very shallow flight-path angles, but also very low densities. Figure 7.11 shows the correlation between $\cos\sigma$ and $\bar{\rho}_D$. Since the density variations are much larger than the C_D variations, this can bee seen as a correlation between density displacement of the atmosphere and $\cos\sigma$ (in fact, the corresponding plot with $\bar{\rho}_L$ is very similar). In the GRAM-99 *ls* the density variations from the nominal US76 profile are smaller than in the US76 *c*, and hence there are less trajectories flown at very high bank-angle. This is further proved by the fact that all the crashes (in the US76 *c* environment) occur not only with shallow initial flight-path angles, but also with atmospheric densities that are between 30 % and 50 % less than nominal.

The latter reasoning explains two major problems of the current concept: one is the reversal, which, when included in the training, causes an ambiguity that leads many trajectories to a large error. There was an attempt to change the reversal direction logic in many different ways: this included shifting the discrimination angle (from 90°, to 95°, 100°, or 80°), which brought some light benefits when moved to 95°, or uniforming the rotation direction, which worsened the problem.

The second major problem is indeed the initial conditions and perturbations in the trajectories used as database. In fact, despite the training distribution is representative of the distribution encountered in testing, it is not a choice that enhances robustness, since some conditions are encountered too rarely during the training. Hence, a distribution of conditions that is not realistic will be expected to give more robustness to the networks.

Another issue that arises from Figure 7.9 is that of infeasible, or close to infeasible, trajectories. Despite all the three trajectories do reach saturation, they clearly do so too slowly. A clear case is that of the crashing trajectories, which reaches full lift-up only after its velocity is way below circular

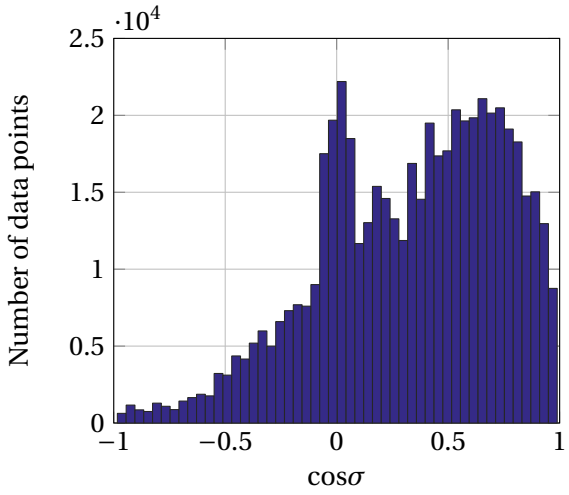


Figure 7.10: $\cos\sigma$ distribution in the training data set of Stage 1.

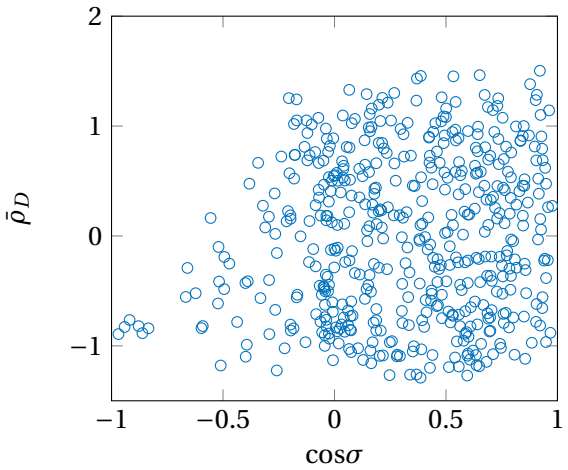


Figure 7.11: Correlation between $\cos\sigma$ and $\bar{\rho}_D$ for the training set of Stage 1. For 3,000 data points picked randomly.

velocity, and hence there is no possibility for aerocapture to occur anymore. This issue is due to the fact that any trajectory that would not reach apoapsis would not be included in the training set.

LESSONS LEARNED

A summary of what can be concluded and implemented from Stage 1 follows. Some of these lessons were then implemented in the following Stage.

1. Given the used formulation of the output, it is convenient to separate the sine from the cosine into two different networks. This was noticed immediately in the beginning of this Stage, and was implemented already.

2. The bank reversal causes a non-uniqueness in the function that has important and negative consequences. Many trajectories that behave very well until the reversal lose their target afterwards.

3. A subset of trajectories with shallow initial flight-path angle and low density profiles are poorly represented in the training data set.

4. The networks, even for steep and easy entries, are still many orders of magnitude less accurate than the NPC. At this point of the training, it is believed that such a large error is caused by the reversal, which brings the vehicle in regions of the input space for which it has not been very well trained.

5. The networks do not know what to do when they are in a state in which the trajectory has become infeasible.

7.1.2 STAGE 2

In the second stage, the previously mentioned lessons learned were implemented as follows:

1. The first item of the previous list was already implemented in the previous stage.

2. The trajectories in the solver were modified to have an instantaneous bank reversal: this way, the non-uniqueness was removed. There was also an attempt to have a finite time reversal, but always in the same direction. This attempt was aborted after too many trajectories did not converge in the solver. This happened for both the reversal directions.

3. The distribution of the entry angles was modified to have shallow entries more frequently than before.

4. Unfeasible trajectories were included in the training set.

At this point of the research, any unfeasible trajectory would still not be considered: this way, it would be possible for the user to decide whether to include any of these solutions. However, it became soon evident that including unfeasible trajectories with the value of the cosine equal to either 1 or -1 was not a good idea. In fact, doing this would cause a non-smoothness of the function, which would be difficult for the networks to learn, and cause the learning to stop early. Hence, at this stage, it would not be possible to implement the last item of the list.

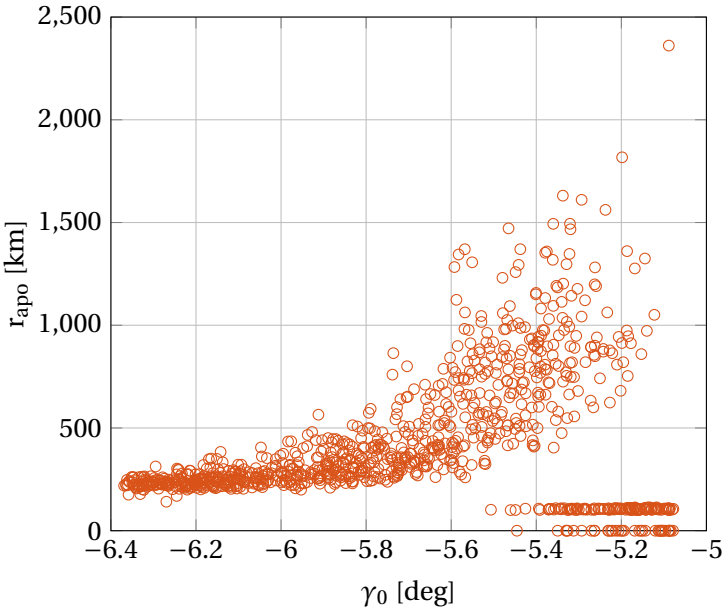


Figure 7.12: Apoapsis altitude for the 4DN of Stage 2 in the US76 *c* environment.

With this correction, the networks were able to achieve RMS during learning similar to those of Stage 1. However, when testing, the performances of the network turned out much worse than the previous case. It is believed that an important issue that occurred is the fact that, after the reversal, which occurs, during testing, in a finite time, the network would end up in states that were never explored during training. Because of that, their accuracy would be much lower. Figure 7.12 shows the performance of the 4DN in the US76 *c*. The problem is clear: any trajectory whose reversal is downwards ends up in a low apoapsis or crashing trajectory, whereas any trajectory whose reversal is upwards ends in a high apoapsis. The pattern is a little similar to that of the SN of Stage 1; however, much more enhanced.

LESSONS LEARNED

This Stage has not improved the previous one. That of the bank reversal is a major issue in this problem, and will hence be avoided for the remaining of this research. In addition, an analysis of the training trajectories showed that still very few were flown with $\cos\sigma \approx 1$, which means that the probability of having a shallow entry should be increased further. The lessons learned can be summarized as:

1. That of the reversal is a complex problem that should be tackled aside from the longitudinal guidance. A possible solution would be that of including entries from a wider set of initial

headings, as well as initial flight-path angles. This will not be done in this research, and the lateral logic will be dropped.

2. Shallow entries are still not frequent enough.

3. The networks have a hard time learning infeasible trajectories if they are flown with $\|\cos\sigma\| = 1$.

7.1.3 STAGE 3

After the retrogression of the previous Stage, it was decided to simplify the problem, and add a few more modifications. The changes include (referring to the list above):

1. The solver now looks for a constant bank-angle only: there would be no reversal in the data set. Also, inclination has been removed from the inputs of the guidance.

2. The distribution of the entry angles has been modified to further increase the probebility of a shallow entry.

3. Unfeasible entries would now be flown (in the solver) with $\|\cos\sigma\| > 1$.

The entry angles distribution was obtained from a uniform distribution with extrema 0 and 1, and raised to the third power. It was then denormalized, such that a 0 would be the shallowest entry, whereas a 1 would be the steepest entry. This resulted in having the output sample distribution shown in Figure 7.13. The same figure also shows the fact that trajectories with imaginary bank-angles were flown. In addition, the number of trajectories for the database was increased to 20,000.

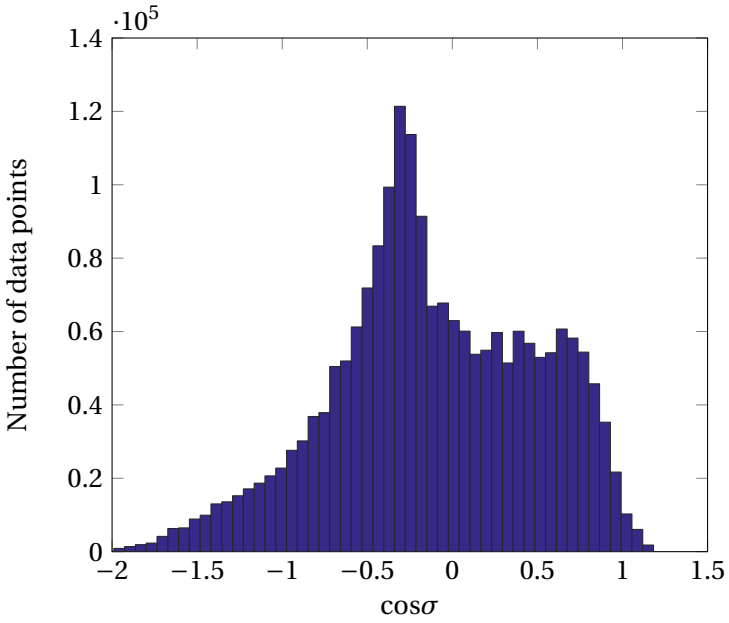


Figure 7.13: $\cos\sigma$ distribution in the training data set of Stage 3.

In this Stage, the SN has been modified to having 25 hidden neurons. To avoid any confusion, this network will be labeled as SN25.

LEARNING

The new dataset included 2,036,366 data points. The SN25 was trained with 800,000 of those, whereas the 4DN was trained with 200,000. In both cases, LM was the chosen training algorithm. In addition, a very large network (labeled VLN) with 5 hidden layers of 128 neurons each was trained with

AdaDelta, given its extremely large number of parameters (83,585). A network of equal size was also trained with AdaGrad, but no sensible differences were found in the final outcome. In this case, ϵ was set equal to 10^{-8}, and the updates were done with mini-batches of three samples at a time. After 500,000 updates, the size of the mini-batches was increased to 256.

The learning curve of the networks trained with LM are very similar to those from Stage 1. This fact is quite important, since the inclination has been removed from the inputs. This means that the differences in trajectories due to perturbations in inclination are negligible, at least from a guidance point of view. This means that deviations in the Coriolis acceleration can be neglected. This is an example of the strong pattern recognition capabilities of the machine learning framework.

The network trained with AdaDelta has a rather noisy learning, curve. The noise decreases once the size of the mini-batches increases. Despite its complexity, the network ended up achieving an RMS of only 0.036, definitely underperfoming it expectations. Nonetheless, a closer check of its performance leads to some doubts. Figure 7.14 shows indeed these patterns. At first sight, it might look like a very similar case to that of the SN shown in Figure 7.2. However, it is much more extreme, since the errors in the beginning of the trajectory are much larger, and much smaller in the remaining of it. This behavior is not even proper overfitting, and is worth further investigations. A closer look to the parameters shows that they are on average very small, leading the network to behave very close to linearly. This might be because the regularization parameter has not been adapted to this larger network, and could have caused underfitting.

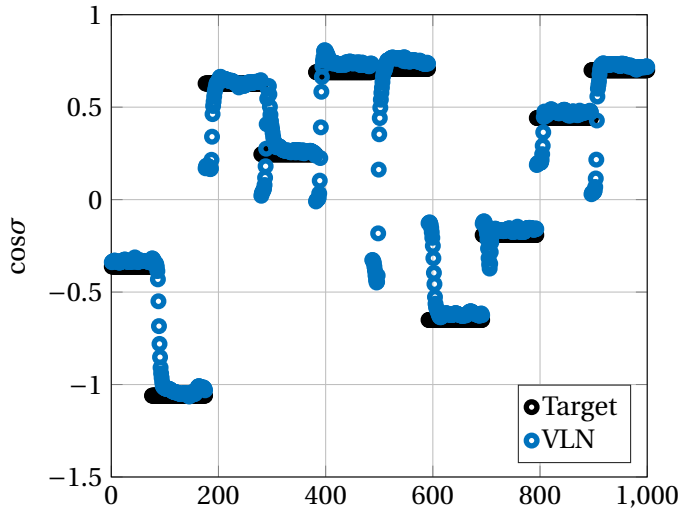


Figure 7.14: Targets and networks outputs for the first 1,000 samples of the dataset of Stage 3.

PERFORMANCE

The three networks have all been tested in the same scenarios as Stage 1.

At this point, the guidance systems have been able to perform much better, especially the SN25 and the 4DN in the GRAM-99 *ls*. An analysis of how these new logics meet the requirements of Section 2.7 will be given in the next subsubsection. Figure 7.15 shows that the VLN has a tendency to overshoot the apoapsis in the US76 *c*. This trend exponentially increases for shallow entries, leading to some misses of 1.000 km. In a case, the apoapsis even goes beyond the limits of the figure. Nevertheless, it causes no crashes, as opposed to the other networks, shown in Figure 7.16. Also in this case, however, there is an outsider that goes beyond the figure.

The figures show that the accuracy is still many orders of magnitude lower than that of the NPC. Also, it is a little difficult to understand which between the SN25 and the 4DN performs best, but it is a rather non relevant race. The VLN is instead clearly out of the competition.

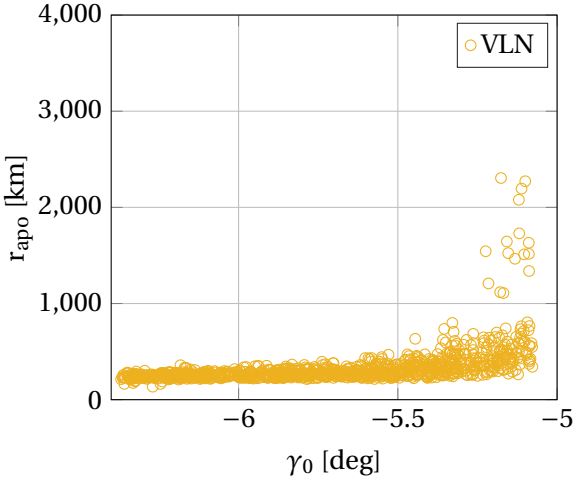


Figure 7.15: Apoapsis altitude for the VLN in the US76 *c* environment.

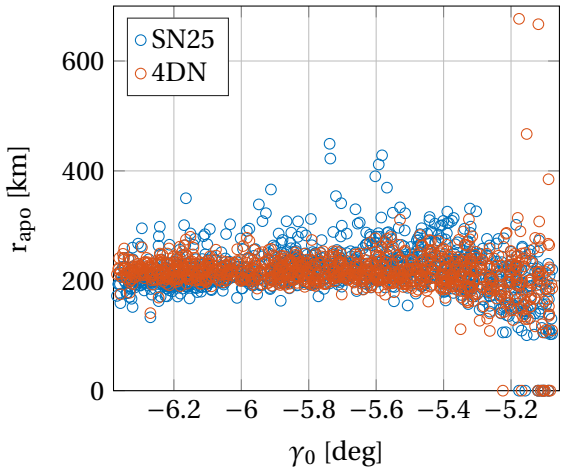


Figure 7.16: Apoapsis altitude for the SN25 and the 4DN in the US76 *c* environment.

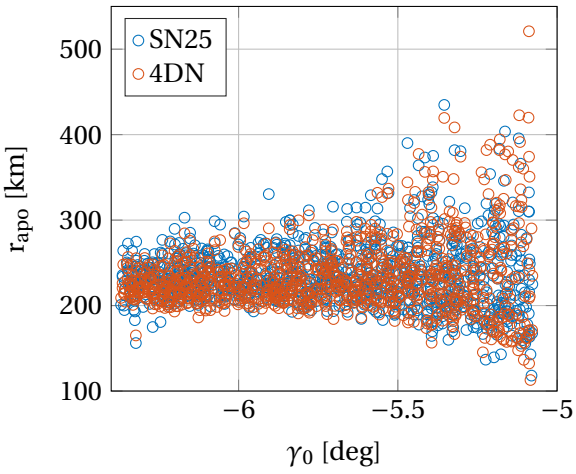


Figure 7.17: Apoapsis altitude for the SN25 and the 4DN in the GRAM-99 *ls*.

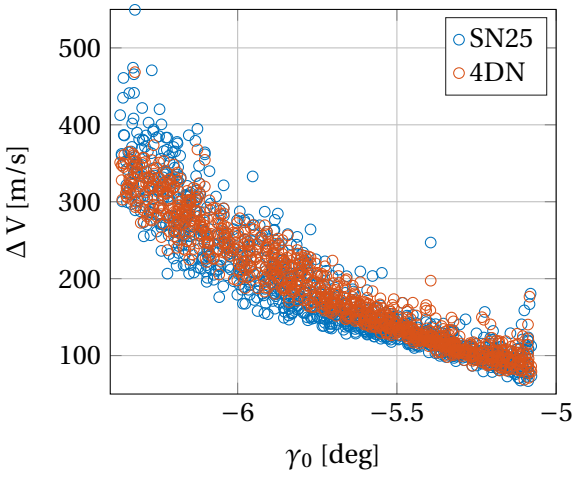


Figure 7.18: ΔV for the SN25 and the 4DN in the GRAM99 *ls*.

For all networks, the performance rather improves in the GRAM-99 *ls*. Figure 7.17 shows that no crashes occur ever for the SN25 and 4DN. The VLN in GRAM-99 *ls* is not reported: it still has its tendency to high apoapsis for shallow entries, but in a lighter way. Eventually, Figure 7.18 shows the ΔV performance in this environment. A strong resemblance to the *simple* NPC guidance is clear. From this point of view, the 4DN slightly outperforms the SN, especially for steep entries. Being the *simple* a common benchmark guidance, the performance of these networks can now be considered acceptable. However, this is in a very limited case, being that of the GRAM-99 *ls*. If larger, even though constant, perturbations were to occur, the networks would not perform well. This is somehow opposite to what happens with the NPCs, and is probably an indication of the fact that during training larger perturbations should be used. However, in terms of accuracy all networks still perform very bad.

The main conclusion of this Stage is that with the current setup, it is not possible to achieve high accuracies in the apoapsis altitude. However, this is not very relevant from a performance point of view, since the final ΔV is similar to the *simple* NPC, a common benchmark for aerocapture guidance (even though not the best performing one).

In addition, there is the suspect that the perturbations in density should be increased in the

training data, at least if it is wanted to have a network capable of performing in the US76 *c* environment.

REQUIREMENTS ASSESSMENT

Also in this case, in terms of the requirement GD-1 of Section 2.7, the SN and the 4DN perform equivalently well. For both networks, in the GRAM-99 *ls* environment, there is a range of entry angles for which the ΔV is bounded with 3 standard deviations. Such value has now been decreased to $210\,\mathrm{m\,s^{-1}}$, and it can be ensured for entry angles between -5.6° and -5.1°. In the US76 *c*, such a range has to be slightly shifted, and the upper bound increased, to avoid a few crashing trajectories.

The VLN can ensure a successful aerocapture for steep initial flight-path angles only. As shown in Subsection 2.5.1, the final ΔV is always larger in those cases. Thus, the VLN is less performing, in terms of the requirements stated for this research. It is difficult to say why this happens. Nonetheless, it is noticed that no crashes occur, for the entirety of the tested cases, even in the US76 *c*.

7.1.4 IMITATING THE OPTIMAL NPC

Given the previous decent results, it was decided to try to imitate the *Lu* guidance with the neural networks. The concept is explained in Section 6.5.

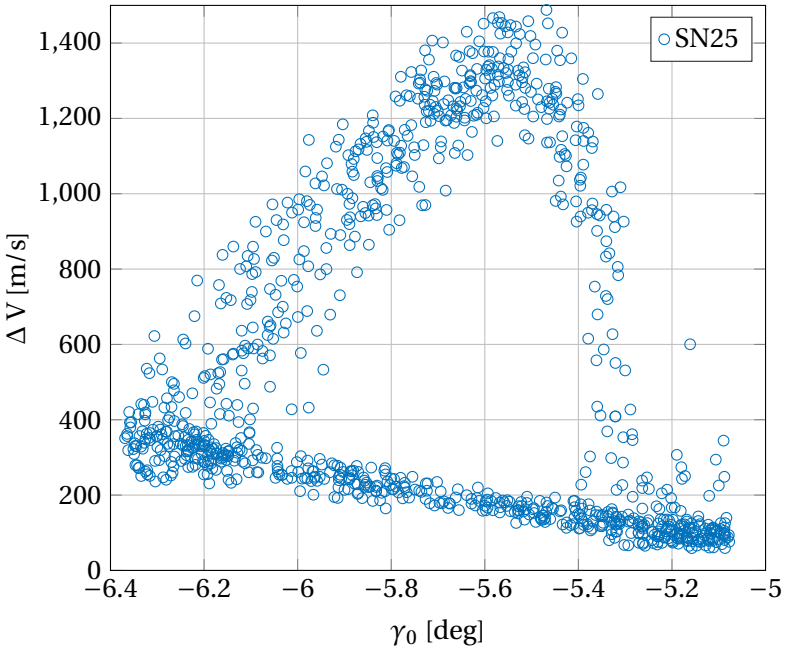


Figure 7.19: ΔV for the SN25 when imitating *Lu* in the GRAM-99 *ls*.

The results produced with this attempt are nonetheless quite disappointing. The ΔV for the SN25 in the GRAM-99 *ls* is shown in Figure 7.19. The apoapsis altitude error shows a similar pattern. The other networks displayed behaviors similar to this, and their performance is therefore not reported. All the trajectories belonging to the ark on top show the same pattern: they trigger Phase 2 way too late, and are then very slow at correcting this, taking an average of 40 s. This attempt is clearly a failure. When run in the US76 *c*, there was a clear correlation with the atmospheric density: the network would end up in the upper ark if the atmosphere was thicker than average.

This is possibly explained by the fact that the *Lu* guidance brought the vehicle in states that were never explored during training. More about this is discussed in the next subsection.

7.1.5 CONCLUSIONS

At this point, some acceptable results have been obtained. Nevertheless, the accuracy in apoapsis altitude is still many orders of magnitude worse than that of the numerical predictor correctors.

Most importantly, the failure in imitating the guidance by Lu made clear what is probably the largest flaw of this implementation, which consists of using trajectories which only sensible variations in initial conditions is the flight-path angle (velocity was also changed, but very mildly, as well as the initial heading, but that does not count when not considering the lateral guidance: the remaining 5 parameters concerned either the vehicle or the environment).

To explain this, it is necessary to go back to the 3-dimensional corridor (energy, flight-path angle, and dynamic pressure) as explained in Subsection 4.3.3. In these datasets the main perturbations were the density, the lift and drag coefficients, and the initial flight-path angle. Let us freeze the first three parameters for the moment, which give a pair of value of ρ_D and ρ_L constant along the trajectory.

A single trajectory with constant bank-angle describes a line in the three-dimensional corridor. At this point, if one varies the initial flight-path angle, the family of trajectories so produced cannot generate other than a surface. This means that, for constant ρ_D and ρ_L, the data set explores only a very small subset of the space; namely, a surface instead of an entire 3-dimensional space. This problem can only be solved by either varying another variable in the initial conditions, which would be the velocity, or by having initial conditions starting randomly in the corridor, as if the spacecraft were spawning randomly with infinite improbability at random points. If the first option is chosen also the range of initial flight-path angles should be widened.

Both approaches have their disadvantages. Varying the initial velocity would lead to having a good portion of the dataset that is outside the corridor, and therefore not really useful. However, having a randomly spawning vehicle might also be problematic, since defining the edges of the 3-dimensional entry corridor may be problematic.

This problem became evident only when trying to imitate the guidance by Lu, but is however probably the main reason why a still large variance in the apoapsis range is obtained; in fact, any perturbation, which includes imprecise commands as well, may lead to a state for which the guidance would not have been able to generalize, and would therefore would give an inaccurate command, which in turn, would act as a further perturbation. Also, this might be the reason why shallow networks perform almost as well as deep networks: not being provided any data outside of the above mentioned surfaces, their performance once out of those has no reason to differ. This is also probably the reason why Figures 7.7 and 7.8 showed no oscillations in the GRAM-99 *ls* environment: the network would indeed not be able to recognize those differences.

Hence, the most important conclusion is that:

> It is necessary to simulate trajectories initialized at random points along the corridor, or to sensibly vary the initial velocity as well.

This conclusion can be read as the necessity to include the RL principle of exploration also in supervised learning. This is because, if supervised learning is used for a control problem, the similarity between the two methods is great.

Eventually, the best performing concept is the one developed in Stage 3, which can ensure a maximum ΔV of $250\,\mathrm{m\,s^{-1}}$ in the GRAM99 *ls* environment.

Other lessons learned are:

1. The inclination is only important for the lateral guidance. This does not mean that the effect of the Coriolis acceleration is negligible, but only that there is no need to account for deviations in it.

2. Learning infeasible trajectories with infeasible commands adds robustness to the guidance.

3. To have a performing network in the US-76 *c* environment it is necessary to increase the density perturbations magnitude.

7.2 POLICY GRADIENT NETWORKS

The RL networks have been trained in the GRAM-99 *l* environment. For reasons previously explained, it means that they will not be able to generalize to environments with stronger perturbations in magnitude, such as the US76 *c*, but even the GRAM-99 *ls*.

The networks were trained using both f_1 and f_2 from Section 6.6 as performance parameter. The training setup is also described there.

The first credit assignment function f_1 grows exponentially for high performing trajectories. There were no real expectations for it to give good global performances, but it was expected to be more of a test to understand whether learning occurs properly. Given that function structure, the policy is expected to highly prioritize learning for the high performing trajectories, neglecting the low performance one, not to mention the crashing trajectories. Because of this, there is the expectation that the so-trained network might perform well for shallow flight-path angles, where the previous networks failed most often. Here, all networks have one single layer with 16 parameters.

Figure 7.20 shows the performance parameter during training for three cases, averaged on 200 iterations. The algorithm was set to stop once the average standard deviation went below a certain value. However, training became uninteresting way before those conditions were reached. In addition, in the case of Set 3, the training performance started decreasing before it was manually stopped. The three sets differ from each other only in terms of initial standard deviation: Set 1 has $\sigma_0 = 1$, Set 2 has $\sigma_0 = .5$, and Set 3 has $\sigma_0 = 2$. The particularity about Set 3 is the fact that, despite having a rather low average performance, it began at around the eight thousandth, displaying some rare but extremely high performing trajectories (with respect to the rest), that were having $f_V = 16$, consisting of a ΔV of about 60 m s^{-1}. Unfortunately, it was not stopped in time to be able to test the network when it was displaying this extreme behavior.

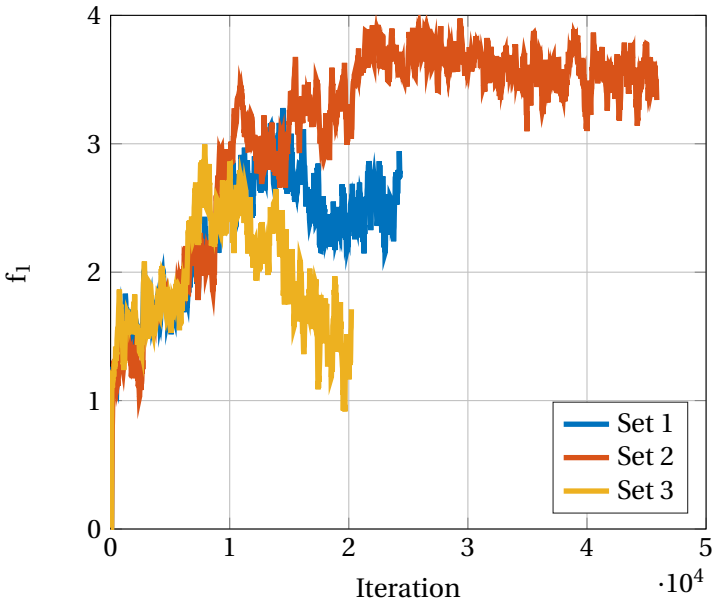


Figure 7.20: Targets and networks outputs for the first 1,000 samples of the dataset of Stage 3.

When tested in a Monte Carlo run, the results show patterns that are almost frustrating to analyze, and none of the expectations are met. To give some of such examples, Figures 7.21 and 7.22

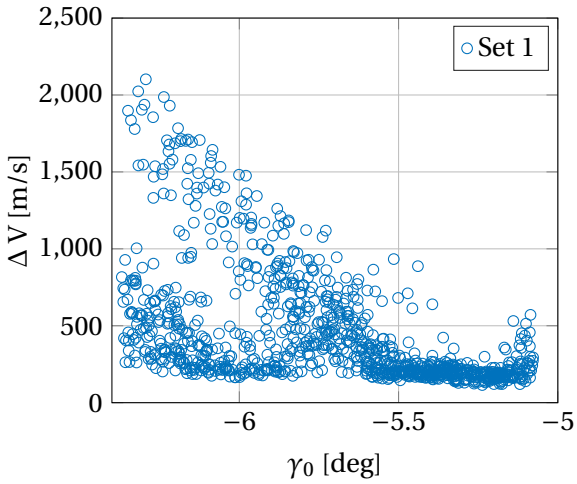


Figure 7.21: Performance of Set 1.

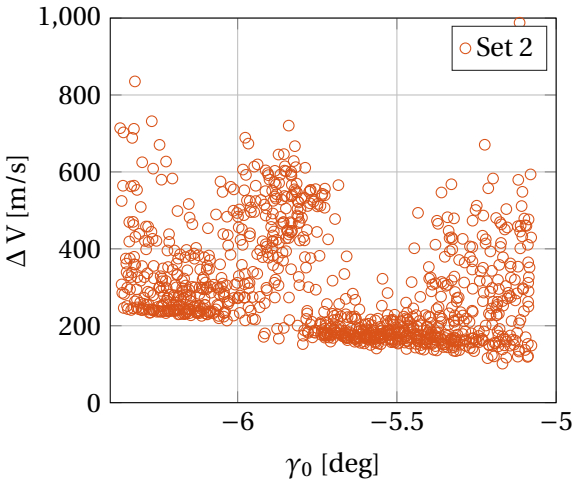


Figure 7.22: Performance of Set 2.

show the performances of Set 1 and Set 2 respectively (both have a crash rate of about 0.5 %, not included in the plots). In Set 1, a dichotomy in ΔV occurs for steep entry angles. There is no correlation between any of the perturbations and whether the performance would lie on the lower or on the higher branch. Hence, the cause of this dichotomy should be searched in the specific behavior of the guidance commands. This is done in Figure 7.23, showing two trajectories with almost same (and steep) entry angles of $-6.1°$. What happens is, once again, clear and evident. At some point, the network gives a command that is about 180° "far" from the current state. This can lead, at any time, to a rotation in one or the other direction. Despite slightly different from the problems encountered in the previous section, this is is a case strictly related to the previous one. The network attempts a turn close to 180°, very close to a bang-bang maneuver (even though, in this case, different from what expected), which has an outcome that may seem as almost random. Here, failure occurs for a different reason from the supervised learning case though. In practice, once the network "realizes" that it turned the wrong direction, it does not even try to correct its trajectory. This is because low performing trajectory are extremely poorly rewarded, no matter what.

If the commanded bank-angle were set to be included between $-180°$ and 180° during testing only, the red trajectory from Figure 7.23 would behave a little similar to the blue one. Nonetheless, it would still have a large final ΔV of $563.3\,\mathrm{m\,s^{-1}}$. This is because it would end in regions of the space that were probably never explored during training. Moreover, when doing the same modification to the entire Monte Carlo set, the average performance decreases, since other trajectories probably rely on the fact that the bank "revresal" occurs in the other direction. It is then definitely important to make this modification during training already.

There is no need to carry out further investigation. This sort of instability may be extremely complicated for a network to deal with, also because the method used relies on gradient information. If it were not for that dichotomy in the ΔV, it seems like the curve were similar (even though less performing) to the one found with the *Lu* guidance, and displayed in Figure 4.23.

Set 2 is instead more merciful. Nonetheless, the patterns are hard to understand also in that case, and the performance is still in no way close to that of the numerical guidance systems.

More training sets, with different metaparameters and credit assignment functions were being carried in parallel while analyzing these first results. None of those ever achieved any particularly good results, most likely for the same reason.

If one considers the distribution of the performance of Set 1, and removes the detached branch, it is possible to recognize a pattern that is similar to that of the *Lu* guidance, but averagely less performing.

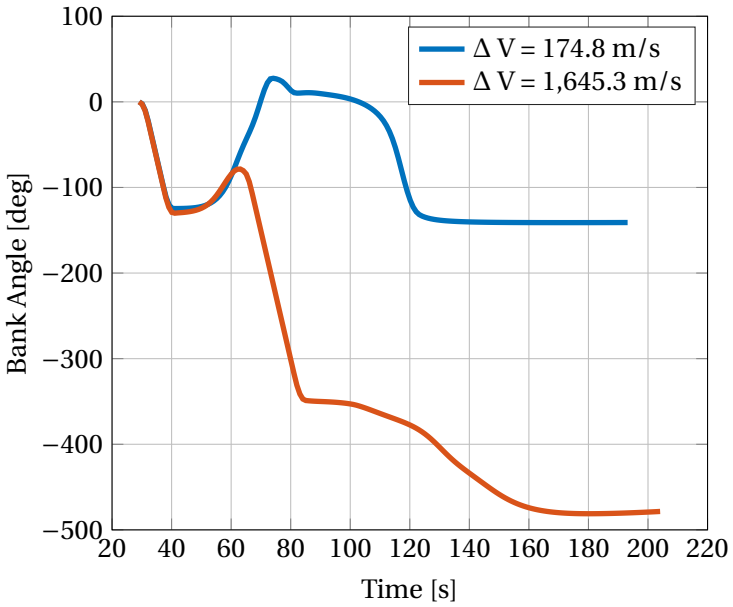


Figure 7.23: Difference in bank-angle history between a high- and a low-performance steep entry trajectory of Set 1.

Comparing the performance of the two sets, it is evident that the results are very sensitive to the training initial conditions and to the different metaparameters used. This could however still be just another consequence of the afore mentioned issue.

In relation to the problem statement of Section 2.7, both Set 1 and Set 2 can ensure aerocapture with a maximum ΔV smaller than $1000\,\mathrm{m\,s^{-1}}$. Nonetheless, such a values is very large, and much less performing than what is achievable with supervised learning networks.

It is thus believed that if the problem of the reversal is solved, good performances could be achieved. This would be quite remarkable, since the trajectories simulated during the training of Set 1 were about 100,000, as opposed to the 6,000,000 done by Gelly and Vernis [2009].

7.3 SUMMARY

This chapter ends not without having had some struggles. It is clear that application of artificial intelligence to this problem is more complex than initially believed, and anything concerning the lateral control causes a variety of unexpected problems.

Some conclusions have been drawn for the supervised learning method in Subsection 7.1.5. A summary of those can be read in the next chapter.

Notably, it is possible to achieve an aerocapture with a neural guidance trained with supervised learning with a total ΔV of less than$210\,\mathrm{m\,s^{-1}}$, for entry angles between -5.1° and -5.6° in the GRAM99 *ls*. Despite being 3 times less performing than the optimal concepts such as the *mod* and *Lu* of Chapter 4, this result is still remarkable because of the much higher computational speed of ANN. This is interesting because of the last requirement stated in Section 2.7.

On what concerns the reinforcement learning part, it was found out way too late what was a main problem during learning. Of course, it might as well not be the only one. Hence, the research question stays open; nevertheless, it can be narrowed down. The next chapter summarizes this work and, among other things, discusses the results in light of the main research question.

8

CONCLUSIONS AND RECOMMENDATIONS

This thesis aimed to answer the research question:

> *Can an artificial intelligence guide a spacecraft to achieve aerocapture in an optimal and robust way?*

The answer is partly positive, in the sense that an artificial intelligence can guide a spacecraft in a robust way, but far from optimal. The best result obtained for the case in object is a ΔV of $210\,\mathrm{m\,s^{-1}}$, whereas the optimal NPC can achieve a ΔV of $90\,\mathrm{m\,s^{-1}}$.

To answer the research question, the state-of-the-art numerical predictor-corrector has also been analyzed, modified, and improved. But before doing so, the aerocapture trajectory has been thoroughly studied in most of its aspects. Thus, the conclusions are divided into three main categories:

1. trajectory related conclusions;
2. NPC related conclusions;
3. artificial intelligence conclusions.

Additional conclusions that can be labeled as "*others*" were inferred either during verification or during the software design. The conclusions are the topic of the next section. Afterwards, recommendations for future work will be listed.

8.1 CONCLUSIONS

To not be repetitive, only the major conclusions will be reported here. Concerning the trajectory of aerocapture:

T1 It has been mathematically proven in Section 4.1 that having an exit orbit with a high periapsis can be, in terms of ΔV, approximately as important as having an exit orbit with an accurate final apoapsis. As an example, for the case in object, an exit orbit that misses the apoapsis by 1 km needs the same ΔV of an exit orbit that is extremely precise in apoapsis, but has a periapsis that is 1 km lower.

T2 In Subsection 4.3.2, it has been mathematically proven that, given some assumptions, and except for some rare exceptions, the integral in time of any function of density and velocity during aerocapture is minimized by a bang-bang trajectory. These functions include most of

the simplified formulas for the heat rate. Thus, it is a good approximation to assume that the total heat load during aerocapture is always minimized by a bang-bang trajectory. The order of the commands depends on the function itself.

T3 In Subsection 4.3.3, it has been shown, with the aid of numerical methods, that heat rate and load factor peaks during an aerocapture are minimized by a bang-bang trajectory. Specifically, such a trajectory should be lit-up lift-down.

T4 For the case in object, the problem of optimizing the aerocapture can be reduced from a multi-objective, constrained optimization, to a single-objective, unconstrained optimization. This was found in Section 4.8.

T5 Because of the short duration of the maneuver, the constraints on the attitude dynamics and kinematics of the vehicle play a major role in the maneuver. This concept was already intuited in Subsection 2.5.1, and further strengthened when analyzing the guidance logics in Section 4.8.

The numerical predictor-corrector by Lu et al. [2015] has then been analyzed, and some modifications have been proposed. Also, because of the previous results, it has been found that such guidance is optimal also in terms of minimizing heat rate and load factor peaks. For the case in object, it also minimizes the total heat load.

The following are the major conclusions that can be drawn from the analysis and the modifications brought to their guidance. Unless otherwise stated, all these conclusions are drawn from the results of Section 4.8.

G1 Including a simplification of the attitude kinematics makes the optimal aerocapture guidance concept more robust. Specifically, as shown in Section 4.7, the transition between Phase 1 and Phase 2 is predicted more accurately. This, in turn, makes it such that less tuning is required to have an optimal aerocapture guidance. Extensive testing in Section 4.8 has shown that choosing a single value of σ_d for the entire mission works appropriately. This is opposed to the original work by Lu et al. [2015], for which $\sigma_d = f\left(V_0, \gamma_0\right)$, and requires some effort to be tuned.

G2 The modified Newton - Raphson method is a numerical method that reduces the number of iterations per guidance call from an average of 10 - 20 needed with Brent's method, to only 2. In all cases tested, it met the requirements. It does not cause any decrease in final ΔV, but in highly perturbed environments such as the US76 *ls* or the GRAM99 *ls*, it leads to apoapsis misses that are averagely 4 km larger than other, more computationally demanding, numerical methods. The method does require some tuning.

G3 The bisection-secant method is a numerical method for which the maximum number of iterations to converge is smaller than Brent's method. It is thus, conceptually, more robust. Nonetheless, an analysis of the number of iterations required has not been carried out.

G4 Both numerical methods that were proposed here are capable to guide an aerocapture with initial hyperbolic velocity. The capability has not been tested though.

G5 In light of the optimal aerocapture guidance problem as stated in Section 2.7, the *mod*, σ_d =120°, guidance can ensure, for a range of initial flight-path angles included between -5.6 degree and -5.1 degree, a successful aerocapture with maximum ΔV of 80 $\mathrm{m\,s^{-1}}$. In addition, a maximum load factor of 5 can be ensured, which satisfies the requirement for manned flight, and a maximum integrated heat load of 650 $\mathrm{MJ\,m^{-2}}$ is never exceeeded. The limit of maximum heat rate

of $18\,\mathrm{MW\,m^{-2}}$ is also never exceeded. All these statements are true for a variety of environments, and do not include the targeting of a specific final inclination. A decrease of 10° in σ_d leads to an average increase of $7\,\mathrm{m\,s^{-1}}$. Such an improvement disappears when small-scale perturbations are included.

G6 The lateral guidance here proposed is very sensitive to how the atmospheric perturbations are modeled. Specifically, in the GRAM99 *ls* there is no wide enough range of initial flight-path angles for which the component of the ΔV due to out-of-plane correction is less than 50 % of the planar component. In the US76 *c* environment instead, such a component is less than 5 % for the same range of initial flight-path angles in which the guidance is optimized. It has not been verified at the moment whether such a low performance in *ls* environments is due to the coupling of the lateral guidance with the modified Newton method, even though this is, at least in part, quite likely, because of the feature stated at the end of Subsection 4.5.2.

On what concerns the guidance logics making use of artificial intelligence, the conclusions are less extensive. These are all based on the results presented in Chapter 7.

AI1 When using machine learning, the choice of the set of trajectories chosen for the training is very important to make the concept robust enough. This set should go beyond the conditions that are expected to be found during testing. Specifically, adding trajectories flown with imaginary bank-angle (which implies $\|\cos\sigma\| > 1$) ensures robustness for flights close to saturated conditions.

AI2 The best result achieved in this research can ensure an aerocapture with a maximum ΔV of $210\,\mathrm{m\,s^{-1}}$. This value slightly changes depending on the environments in which the guidance is tested. Both the SN25 and the 4DN can achieve such a result, when trained using LM. A larger network trained with AdaDelta is instead less performing. It is believed that better performances have not been obtained mainly because of the previous conclusion. Despite being less performing, the neural guidance so obtained is very interesting, since it is sure it can be used in real time. Such a statement cannot be said certainly true for the NPC instead.

AI3 For the setup used in this research, the lateral guidance works best if treated by a separate network. This is because of the discontinuous nature of that logic.

AI4 PGPE is an RL method that still seems promising for application to the optimal aerocapture guidance concept. Nevertheless, it is difficult to use. As an example, a seemingly very small detail lead, during this research, to the impossibility of having a converging network. It is possible that additional problems need to be addressed before such a method can be successful.

Eventually, *other* conclusions include:

OT1 It was shown, in Section 6.6, that PGPE, combined with a parameterization provided by an FFNN, can be used as a trajectory optimizer. The result achieved was not the theoretical optimum. Nonetheless, the so optimized trajectory was obtained for the GRAM99 *l* environment, it included the ΔV due to inclination correction, as well as constraints caused by the rotation. Hence, the difference from the theoretical optimum can be justified. Nonetheless, further comparisons should be carried out in this direction. Eventually, the optimized trajectory was obtained after 5,000 iterations, which implies 20,000 simulated trajectories.

OT2 A way to reproduce optimal attitude control in the simulation, while respecting the constraints on maximum angular velocity and accelerations has been proposed, and proven to work efficiently, in Subsection 6.2.6. Optimality is in the sense of achieving the target command in minimum time, without overshoot.

OT3 When treating the vehicle as a point mass, the benefits of using high order integrators highly reduce. This is mainly caused by the discontinuities in temperature gradients of the atmosphere, which, in turn, cause discontinuity in the scale height. The convergence exponent of RK5 is reduced to 3.64, instead of the theoretically achievable 3.1. This conclusion was drawn in Section 6.3.1.

8.2 RECOMMENDATIONS

Recommendations for future work are strictly related to the conclusions. The following would be interesting topics for further studies:

1. Related to Conclusions T2 and T4, it would be interesting to study the conditions required for the minimum heat load trajectory to be the same as the minimum ΔV trajectories. So far, it has been only shown that the two trajectories do coincide for the case in object, but do not for lower entry trajectories. Dependencies on the vehicle should also be studied.

2. It is believed that what has been stated in Conclusion T3 might be mathematically proven. Such a demonstration would be of valuable interest.

3. Related to Conclusion G1, it would be interesting to see how the *mod* concept works if the actuators are less than optimal. Such a concept requires indeed an estimation of $\dot{\sigma}_{avg}$, which has been tuned for the constraints enforced in this research. Such constraints have not been varied during testing. Thus, evaluating the response of the guidance to such a deviation would be of valuable interest. Of course, the robustness of the guidance to errors in navigation should also be studied.

4. The statement of Conclusion G3 should be put to test. The same should be done for Conclusion G4.

5. Strictly related to Conclusion G6, it should be studied whether the inaccuracies of the lateral guidance in *ls* environments is cause by a coupling with the modified Newton method. In other words, the *modlat* concepts should be tested with the bisection-secant method.

6. Concerning Conclusions AI1 and AI2, a dataset with a wider range of initial conditions, that include variations in initial velocities, as well as wider range of initial flight-path angles, should be used. It is believed indeed that the networks in this research have been trained for only a subspace of all the possible states, making them less robust.

7. A consequence of Conclusion AI4 is the fact that it should be studied whether PGPE can give better performance for this problem, if the issue found in Section 7.2 is solved.

8. In general, it would be of valuable interest to use different sets of inputs for the neural guidance logics.

9. In relation to Conclusion OT1, the optimization method should be compared to the results of other optimizers, in similar conditions. Moreover, it should also be tested with other parameterizations, such as the more common $\sigma = f(E)$.

10. Concerning Conclusion OT2, it would be of valuable interest to use the same concept as a real attitude dynamics controller, even though this would require some modifications of the outputs. Interestingly, such a concept is believed to be possibly used in real-time, even at very high frequencies, since it only requires solving four explicit equations. Thus, there would be no need of tracking the planned bank-angle trajectory, since such a trajectory could be recomputed almost instantaneously.

11. The statement of Conclusion OT3 should be complemented with an evaluation of even higher order solvers, including, possibly, integrators of classes other than RK. Moreover, it would be interesting to evaluate whether such a conclusion would be valid when including attitude dynamics as well. Attitude dynamics are usually much faster than longitudinal dynamics, and thus in such a case it is likely that high order integrators would still be beneficial.

Appendices

A

PREDICTIVE LOAD-RELIEF

The predictive load-relief is a method developed by Lu [2014] by which an NPC can, at the same time, satisfy the constraints on heat flux peak and on load factor, and still maximize the performance. This concept is somewhat opposite to that of the Apollo skip entry guidance, in which, when the load factor or the heat flux is sensed to be larger than the constraint, full lift-up is commanded [Moseley, 1969]. This way, the vehicle completely loses its target. Moreover, the constraint will be exceeded anyway, since the loads would keep on increasing for some time even despite the full lift-up command. With the predictive load-relief developed by Lu [2014], not only the vehicle starts reacting way before the loads are exceeded, but also, it does so while still not abandoning its target.

The concept is simple, but extremely ingenious: at any point in the predicted trajectory, the derivatives of heat flux and load factor are computed: if they are such that, after having been linearized, it is predicted that one of the constraints would be exceeded within some time δ, then the bank angle at that point starts to deviate from the guessed bank angle by a certain amount, proportional to how much the excess would be. This happens in the prediction, and therefore, once convergence occurs, one has a predicted trajectory whose bank angle is constant for most of its part, but increases whenever the load constraints are dangerously approached. This way, not only the constraints are never approached, but the load-relief is also predicted since the beginning of the trajectory. Lu [2014] shows how this method is successful in entry guidance. In general, it leads to an initial more downward bank angle, which causes a larger energy dissipation in the beginning, to then increase the bank angle, when the loads become relevant.

The method was applied to the aerocapture NPC, as advised by Lu et al. [2015]. Conceptually speaking, this method has a few drawbacks when applied to aerocapture.

First of all, its major benefit is that of causing larger energy dissipations in the beginning, when the load factor and heat rate are not too large, such that less energy needs to be dissipated later. This is useful for an entry problem, where there is a constraint in range, but not much in aerocapture, where there is not such a constraint. It is indeed this lack of range constraint that makes it such that the bang-bang approach is the most effective also from a constraint point of view.

Second, the fact that this NPC guidance is divided in two phases, makes it a little complicated to apply this method. It might make sense to use it only in Phase 2, since in Phase 1 the bank angle is fixed. Nevertheless, it should be reminded that Phase 2 is initially planned to be flown at a certain constant σ_d, which is limited to a certain value for robustness purposes: if Phase 2 were flown at even larger bank angles (which is what this predictive load-relief causes it to do), the whole purpose of limiting σ_d would be lost. In other words, the method would decrease the robustness.

Third, the minimum load trajectory is indeed a bang-bang, and there is not much that can be done about it: in most cases, all the peaks happen in Phase 1, after a full lift-up segment of flight,

which means that there is no way that the loads can be reduced further. If that is not the case, it would mean that the entry flight path angle was very shallow, and the peaks would not be exceeded. In the verification (immediately subsequent to this subsection), it will be shown how the method was implemented, and validated it on an entry case; however, it will also be shown that, when applied to aerocapture, the method causes some problems, and this makes its application not very interesting. At last, below is reported the algorithm used by Lu [2014], step by step (only the load factor is treated, but the derivation is absolutely equivalent for the heat flux).

The main concept on which the derivation is based is that the dynamics of the flight-path angle are slower than the altitude dynamics. Therefore, the sine of the flight-path angle is used as pseudocontrol for the slower altitude dynamics, since:

$$\dot{r} = V \sin\gamma \tag{A.1}$$

The derivative of the acceleration $a = \sqrt{L^2 + D^2}/m$ is computed as (for constant aerodynamic coefficients):

$$\dot{a} = -\frac{2aD}{V} + aV\beta_r \sin\gamma := A_a + B_a \sin\gamma, \tag{A.2}$$

where the approximation $\dot{V} = -D$ has been used. A_a and B_a are both negative. δ is a prediction horizon parameter. Linearizing, the acceleration at the end of the finite horizon is approximated:

$$a\left(t+\delta\right) = a\left(t\right) + \dot{a}\delta = a + A_a\delta + \delta B_a \sin\gamma. \tag{A.3}$$

Since the goal is that of enforcing $a\left(t+\delta\right) \leq a_{max}$:

$$\sin\gamma \leq \frac{a_{max} - a - A_a\delta}{B_a\delta} := \sin\gamma_{max}. \tag{A.4}$$

During the entirety of the trajectory, a reference flight-path angle is defined:

$$\sin\gamma_{ref} = \max\left\{\sin\gamma, \sin\gamma_{max}\right\}, \tag{A.5}$$

for sufficiently small δ, $\gamma_{ref} = \sin\gamma$ for most of the trajectory.

At this point, one defines also $\dot{h}_{ref} = V\sin\gamma_{ref}$; if such altitude rate is enforced, the maximum acceleration should not be exceeded in the finite horizon δ. Eventually, in the prediction (as well as in the guidance logic output) the bank-angle control is:

$$L\cos\sigma_{cmd} = L\cos\sigma_{base} - k_0\left(\dot{h} - \dot{h}_{ref}\right), \tag{A.6}$$

where $k_0 \leq 0$ is a constant gain. This latter passage is justified by the fact that $L\cos\sigma$ directly influences $V\dot{\gamma}$ in the equations of motion. σ_{base} is the constant bank angle used in the prediction. However, Equation A.6 is used also when computing the guidance logic output. It is stressed that despite the one applied is a finite horizon prediction in the interval $\left[t, t+\delta\right]$, the consequence is equivalent as to having an infinite horizon prediction, since Equation A.6 is enforced along the predicted trajectory: therefore, t slides along the entire trajectory.

A.1 VERIFICATION

The method has been implemented and included in the NPC guidance. Figures A.1 and A.2 show the open-loop trajectories predicted when setting different constraints. Entry conditions are the same for all three cases. There is no target crossrange, and thus the baseline bank angle has been manually changed from one case to another, being 50° for the unconstrained case, 100° for the case constrained at 4 g and 150° for the trajectory constrained at 2.5 g. The prediction seems to be properly working: however, the trajectories are obtained by letting the cosine of the bank angle unconstrained, which, in turn, becomes larger than one. Thus, the figures show that the concept works properly but tuning is not appropriate for these cases (in which $k_0 = 1000$ and $\delta = 16$ s).

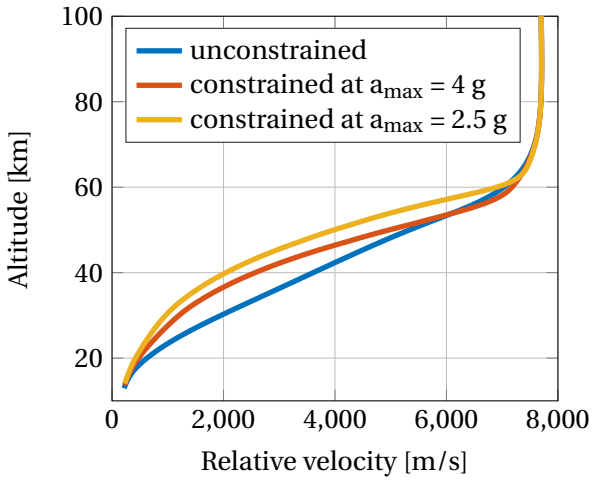


Figure A.1: Open-loop simulation of re-entry with predictive load relief.

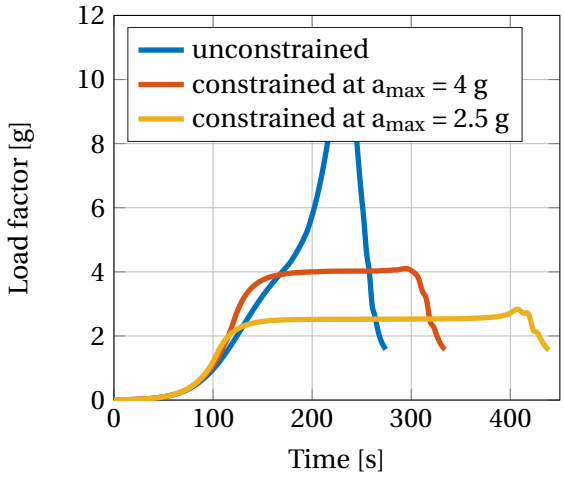


Figure A.2: Load factor history for open-loop simulation of a re-entry with active predictive load relief.

In the open-loop guidance, cosines of the bank angle larger than 1 are allowed. In fact, because of the adaptive solvers used, constraining the cosine of the bank angle would trigger major instabilities. As an example, when running the simulation of the trajectory constrained at 2.5 g, and constraining the cosine of the bank angle, the simulation requires 4915 time-steps, 91 % of which are computed in a region of time between 155 s and 165 s, with an average step-size of 2.3 ms. As a comparison, only 500 steps are needed (highest tolerance is being used at the moment). These instabilities can be seen in Figure A.4. Moreover, the so introduced constraint does not improve the quality of the trajectory: the only difference now is that, since control authority is limited, the trajectory does not satisfy the constraint. Figure A.3 shows in fact that the trajectory predicted limiting the cosine of the bank angle violates largely the limit on the load factor, for a relatively large period of time. Therefore, introducing this constraint in the NPC only does not offer any advantages.

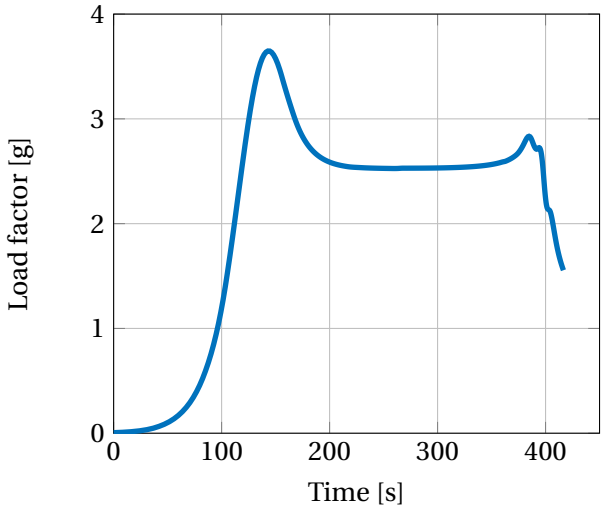


Figure A.3: Load factor history for the an open-loop infeasible trajectory with constraints on $\cos\sigma$.

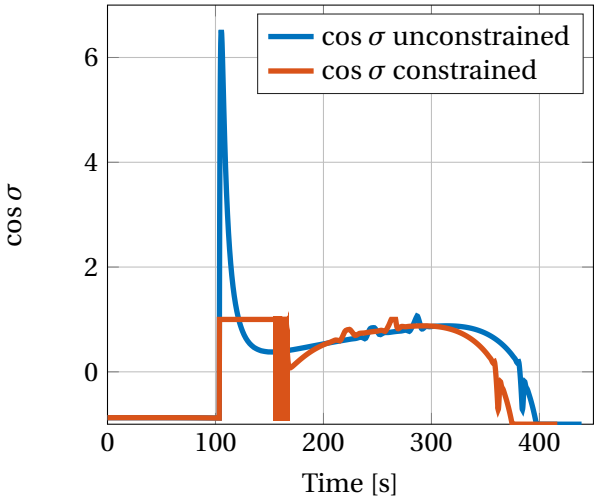


Figure A.4: Bank angle profiles for trajectories with and without constraints on $\cos\sigma$.

An extensive trade-off has been done to attempt to optimize the parameter in a way that the constraints are respected as much as possible, while keeping the cosine of the bank angle as low as possible. After analysing Figures A.5, A.6, A.7 and A.8, in which different combinations of the

parameters have been attempted, it is noticed that a way to both satisfy the load factor constraint and the bank angle constraints, consists of using a large horizon and a small gain. This way, it is given more time to the guidance to react to possible future load factors. Moreover, a small gain is necessary if one does not want to force the bank angle to exceed its physical limits.

However, such situation may cause too large deviations in the original trajectory, and also causes the trajectory to stay very far from its constraints. This becomes especially problematic in aerocapture. For these reasons, the values advised by Lu [2014] will also be used in aerocapture as first guess, and then it will be evaluated whether they should or not be changed. However, it was immediately noticed that a large value of k_0 leads to too frequent skipout: hence, when used, the predictive load relief will have a very small valu of k_0, compensated, if necessary, by more conservative constraints.

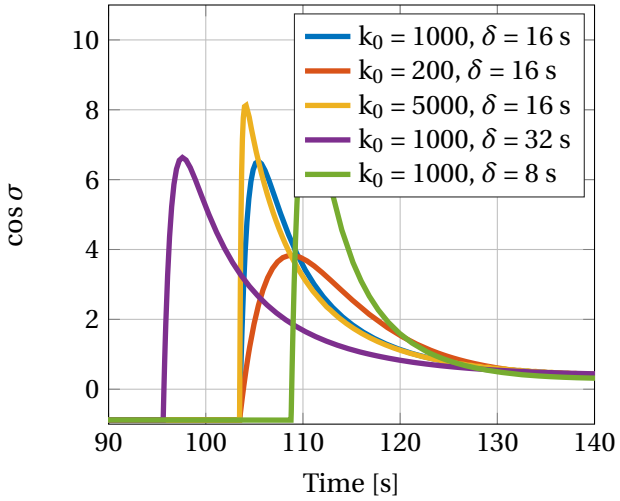


Figure A.5: Sensitivity of the bank angle profile (cosine) planned by the NPC for the case of flight constrained at 2.5 g with respect to the parameters k_0 and δ.

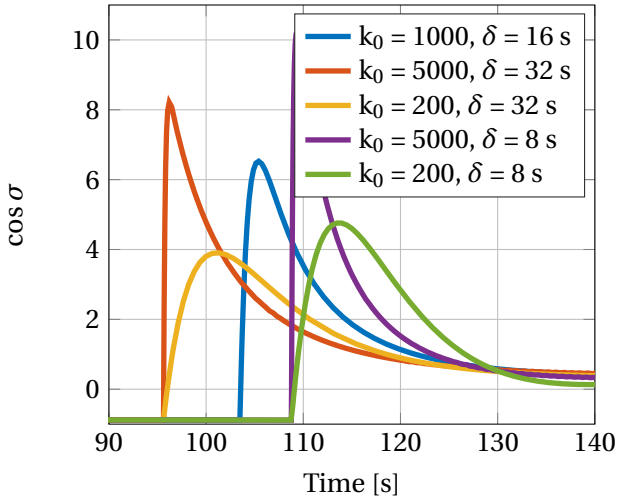


Figure A.6: Sensitivity of the bank angle profile (cosine) planned by the NPC for the case of flight constrained at 2.5 g with respect to the parameters k_0 and δ.

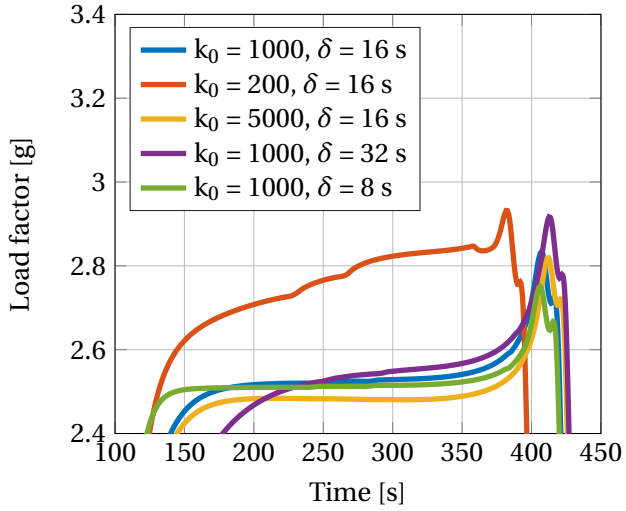


Figure A.7: Sensitivity of the load factor history for the trajectory planned by the NPC for the case of flight constrained at 2.5 g with respect to the parameters k_0 and δ.

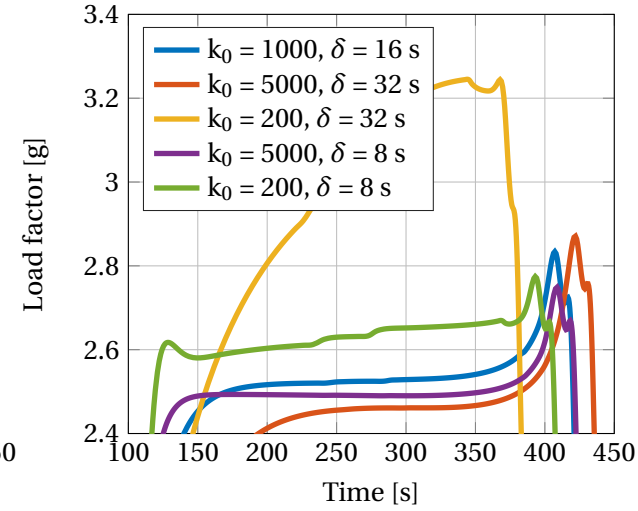


Figure A.8: Sensitivity of the load factor history for the trajectory planned by the NPC for the case of flight constrained at 2.5 g with respect to the parameters k_0 and δ.

As a final remark, one should again look at Figure A.3. Despite such load factor profile is very

close to that in Figure 17 of Lu [2014], which is for an unfeasible trajectory as well, one might notice that the second peak has no reason to exist. In fact, when comparing the trajectory with the corresponding bank angle profile, (red line in Figure A.4), it is obvious that the bank angle is not saturated when the peak happens, about 370 s after t_0. Such a peak also happens in both constrained trajectories in Figure A.2. In a more thorough analysis of the trajectory constrained at 2.5 g, with no limit on the cosine of the bank angle, it is noticed that the computation of $\dot{a}$ is wrong, and begins to be negative at about 365 s after t_0; at that point, a is still larger than a_{max}, and therefore the bank angle is still different from the baseline. In fact, the predicted future acceleration after δ is larger than the maximum allowed until $t = 395$ s; after that, the bank angle is reduced to the baseline profile, whereas the load factor is still increasing. This peak can therefore be caused by a wrong assumption when analytically computing $\dot{a}$. The main simplification done in that process is that of neglecting gravity when assuming $\dot{V} = -D$, therefore neglecting the radial component of the gravity (the other components are all much smaller); the error done is in fact negligible in the initial phase of the trajectory, when the flight path angle is very small, but may become considerably larger in the final phase, when the trajectory is steeper for a capsule-like vehicle.

Moreover, this error is relatively more important, the smaller the maximum acceleration. To solve this, the method is rewritten assuming $\dot{V} = -D - g_r \sin\gamma$. The result can be found in Figure A.9: there, the remaining, much smaller, peaks, are caused only by large changes of the aerodynamic coefficients, that mainly occur at small Mach number, and therefore should not occur during aerocapture. These peaks in fact disappear when simulating using constant aerodynamic coefficients. In the figure, one can also find the acceleration predicted by the guidance after time δ: since the controller in use is only proportional to the difference between the maximum admissible load factor and the predicted one, such acceleration is never brought exactly to zero, and this causes an almost constant bias of 0.02 g during the entire trajectory.

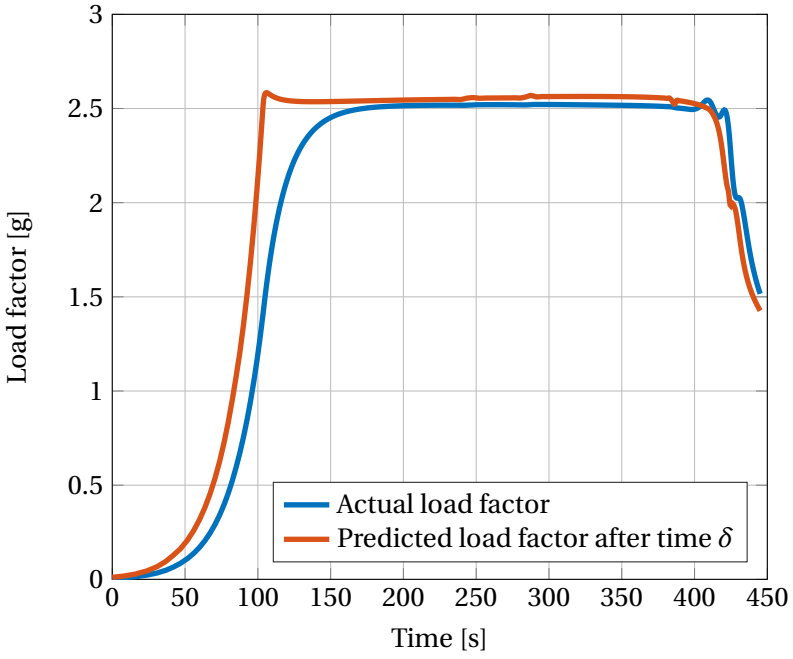


Figure A.9: Load factor history using predictive load relief that assumes $\dot{V} = -D - g_r \sin\gamma$. Constraint at 2.5 g, cosine of the bank angle unconstrained.

BIBLIOGRAPHY

Acikmese, B. and Blackmore, L. [2011]. "Lossless convexification of a class of optimal control problems with non-convex control constraints", *Automatica*, volume 47, no. 2, pp. 341–347, doi:10.1016/j.automatica.2010.10.037.

Amari, S.-i. [1998]. "Natural Gradient Works Efficiently in Learning", *Neural Computation*, volume 10, no. 2, pp. 251–276, doi:10.1162/089976698300017746.

Amari, S.-i. [2016]. "Natural Gradient Learning and Its Dynamics in Singular Regions", in *Information Geometry and Its Applications*, doi:10.1007/978-4-431-55978-8.

Amari, S.-i. and Douglas, S. [1998]. "Why natural gradient?", in *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98 (Cat. No.98CH36181)*, volume 2, IEEE, pp. 1213–1216, doi:10.1109/ICASSP.1998.675489.

Amari, S.-i., Park, H., and Fukumizu, K. [2000]. "Adaptive Method of Realizing Natural Gradient Learning for Multilayer Perceptrons", *Neural Computation*, volume 12, no. 6, pp. 1399–1409, doi:10.1162/089976600300015420.

Anderson, J. D. [2000]. *Hypersonic and High Temperature Gas Dynamics*, American Institute of Aeronautics and Astronautics.

Armellin, R. and Lavagna, M. [2008]. "Multidisciplinary Optimization of Aerocapture Maneuvers", *Journal of Artificial Evolution and Applications*, volume 2008, pp. 1–13, doi:10.1155/2008/248798.

Bellman, R. E. [1957]. *Dynamic Programming*, Princeton University Press.

Bhatnagar, S., Sutton, R. S., Ghavamzadeh, M., and Lee, M. [2009]. "Natural actor – critic algorithms", *Automatica*, volume 45, no. 11, pp. 2471–2482, doi:10.1016/j.automatica.2009.07.008.

Brent, R. P. [1973]. *Algorithms for Minimization without derivatives*, Prentice-Hall, Englewood Cliffs, NJ.

Carandente, V., Savino, R., Iacovazzo, M., and Boffa, C. [2013]. "Aerothermal Analysis of a Sample-Return Reentry Capsule", *Fluid Dynamics & Materials Processing*, volume 9, no. 4, pp. 461–484.

Casoliva, J., Lyons, D., Wolf, A., and Mease, K. [2008]. "Robust Guidance via a Predictor-Corrector Algorithm with Drag Tracking for Aero-Gravity Assist Maneuvers", in *AIAA Guidance, Navigation and Control Conference and Exhibit*, American Institute of Aeronautics and Astronautics, Reston, Virigina, doi:10.2514/6.2008-6816.

Cazaux, C., Naderi, F., Whetsel, C., Beaty, D., Gershman, B., Kornfeld, R., Mitcheltree, B., and Sackheim, B. [2004]. "The NASA/CNES Mars sample return— a status report", *Acta Astronautica*, volume 54, no. 8, pp. 601–617, doi:10.1016/j.actaastro.2003.07.001.

Cerimele, C. J. and Gamble, J. D. [1985]. "A Simplified Guidance Algorithm for Lifting Aeroassist Orbital Transfer Vehicles", in *AIAA 23rd Aerospace Sciences Meeting*, Reno, NV.

Cheng, T. and Lewis, F. L. [2007]. "Neural network solution for finite-horizon H-infinity constrained optimal control of nonlinear systems", *Journal of Control Theory and Application*, volume 5, no. 1, pp. 1–11.

Ciancolo, D., Alicia, M., Davis, J. L., Engelund, W. C., Komar, D. R., Queen, E. M., Samareh, J. a., Way, D. W., Zang, T. a., Murch, J. G., Krizan, S. a., Olds, A. D., Powell, R. W., Shidner, J. D., Kinney, D. J., McGuire, M. K., Arnold, J. O., Covington, M. A., Sostaric, R. R., Zumwalt, C. H., and Llama, E. G. [2011]. "Entry, Descent and Landing Systems Analysis Study: Phase 2 Report on Exploration Feed-Forward Systems", Technical Report July, NASA/TM-2010-216720.

Cruz, M. I. [1979]. "The aerocapture vehicle mission design concept", in *Conference on Advanced Technology for Future Space Systems*, Hampton, VA, pp. 195–201.

Desjardins, G., Simonyan, K., Pascanu, R., and Kavukcuoglu, K. [2015]. "Natural Neural Networks", in *Advances in Neural Information Processing Systems 28 (NIPS 2015)*.

Detra, R. W. and Hidalgo, H. [1961]. "Generalized Heat Transfer Formulas and Graphs for Nose Cone Re-Entry Into the Atmosphere", *ARS Journal*, volume 31, no. 3, pp. 318–321.

Diebel, J. [2006]. "Representing Attitude: Euler Angles, Unit Quaternions and Rotation", Technical report, Stanford University.

Drake, B. G., Hoffman, S. J., and Beaty, D. W. [2010]. "Human exploration of mars, design reference architecture 5.0", in *IEEE Aerospace Conference Proceedings*, number July.

Duchi, J., Hazan, E., and Singer, Y. [2011]. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization", *Journal of Machine Learning Research*, volume 12, pp. 2121–2159, doi: 10.1109/CDC.2012.6426698.

Engelbrecht, A. P. [2007]. *Computational Intelligence: An Introduction*, 2nd edition, John Wiley & Sons.

Gamble, J. D., Cerimele, C. J., Moore, T. E., and Higgins, J. [1988]. "Atmospheric Guidance Concepts for an Aeroassist Flight Experiment", *Journal of the Astronautical Sciences*, volume 36, pp. 45–71.

Gelly, G. and Vernis, P. [2009]. "Neural Networks as a Guidance Solution for Soft-Landing and Aerocapture", in *AIAA Guidance, Navigation, and Control Conference*, pp. 1–12.

Gill, P., Murray, W., and Saunders, M. A. [2007]. "User's Guide to SNOPT Version 7: Software for Large-scale Nonlinear Programming", Technical report, University of California.

Gnoffo, P. A. [2003]. "Computational Aerothermodynamics in Aeroassist Applications", *Journal of Spacecraft and Rockets*, volume 40, no. 3, pp. 305–312, doi:10.2514/2.3957.

Grondman, I., Busoniu, L., Lopes, G. a. D., and Babuška, R. [2012]. "A survey of actor-critic reinforcement learning: Standard and natural policy gradients", *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, volume 42, no. 6, pp. 1291–1307, doi: 10.1109/TSMCC.2012.2218595.

Gurley, J. G. [1993]. "Guidance for an aerocapture maneuver", *Journal of Guidance, Control, and Dynamics*, volume 16, no. 3, pp. 505–510, doi:10.2514/3.21038.

Hairer, E., Wanner, G., and Norsett, S. P. [1993]. *Solving Ordinary Differential Equations I: Nonstiff Problems*, Springer.

Hall, J. L., Noca, M. a., and Bailey, R. W. [2005]. "Cost-Benefit Analysis of the Aerocapture Mission Set", *Journal of Spacecraft and Rockets*, volume 42, no. 2, pp. 309–320, doi:10.2514/1.4118.

Hamel, J.-F. and Lafontaine, J. D. [2006]. "Improvement to the Analytical Predictor-Corrector Guidance Algorithm Applied to Mars Aerocapture", *Journal of Guidance, Control, and Dynamics*, volume 29, no. 4, pp. 1019–1022, doi:10.2514/1.20126.

Hastie, T., Tibshirani, R., and Friedman, J. [2001]. *The Elements of Statistical Learning*, 2nd edition, Springer.

Huang, G., Huang, G. B., Song, S., and You, K. [2015]. "Trends in extreme learning machines: A review", *Neural Networks*, volume 61, pp. 32–48, doi:10.1016/j.neunet.2014.10.001.

Huang, G.-B., Zhu, Q.-y., Siew, C.-k., Ã, G.-b. H., Zhu, Q.-y., Siew, C.-k., Huang, G.-B., Zhu, Q.-y., and Siew, C.-k. [2006]. "Extreme learning machine: Theory and applications", *Neurocomputing*, volume 70, no. 1-3, pp. 489–501, doi:10.1016/j.neucom.2005.12.126.

iMars Working Group [2008]. "Preliminary Planning for an International Mars Sample Return Mission Report of the International Mars Architecture for the Return of Samples (iMARS) Working Group", Technical report, iMars.

Jacchia, L. G. [1970]. "New Static Models of the Thermosphere and Exosphere with Empirical Temperature Profiles", Technical report, Smithsonian Astrophysical Institute.

Junkins, J. L. and Turner, J. D. [1986]. *Optimal Spacecraft Rotational Maneuvers*, Elsevier.

Justus, C. G., Jeffries, W. R., Yung, S. P., and Johnson, D. L. [1995]. "The NASA / MSFC Global Reference Atmospheric Model - 1995 Version (GRAM-95)", .

Justus, C. G. and Johnson, D. L. [1999]. "The NASA/MSFC Global Reference Atmospheric Model - 1999 Version (GRAM-99)", , no. May.

Kakade, S. M. [2001]. "A Natural Policy Gradient", *Advances in neural information processing systems*, pp. 1531–1538, doi:10.1.1.19.8165.

van Kampen, E.-J., Chu, Q., and Mulder, J. [2006]. "Continuous Adaptive Critic Flight Control Aided with Approximated Plant Dynamics", in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, American Institute of Aeronautics and Astronautics, Reston, Virigina, doi:10.2514/6.2006-6429.

Kolmogorov, A. N. [1957]. "On the Representation of Continuous Functions of Several Variables as Superpositions of Continuous Functions of one Variable and Addition", *Doklady Akademii Nauk SSSR*, volume 114, no. 5, pp. 953–956.

Kurková, V. [1992]. "Kolmogorov's theorem and multilayer neural networks", *Neural Networks*, volume 5, no. 3, pp. 501–506, doi:10.1016/0893-6080(92)90012-8.

Lafleur, J. M. [2011]. "The Conditional Equivalence of Delta-V Minimization and Apoapsis Targeting in Numerical Predictor-Corrector Aerocapture Guidance", Technical report, NASA/TM-2011-216156, Johnson Space Center, Houston, TX.

Le, Q. V., Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., and Ng, A. Y. [2011]. "On optimization methods for deep learning", in *Proceedings of 28th International Conference of Machin Learning*.

LeCun, Y., Bengio, Y., and Hinton, G. [2015]. "Deep learning", *Nature*, volume 521, no. 7553, pp. 436–444, doi:10.1038/nature14539.

LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K. R. [1998]. "Efficient BackProp", in *Neural Networks: Tricks of the Trade*, Springer.

Leslie, F. W. and Justus, C. G. [2011]. "The NASA Marshall Space Flight Center Earth Global Reference Atmospheric Model—2010 Version", Technical report, NASA/TM-2011-216467.

Liu, X., Shen, Z., and Lu, P. [2016]. "Entry Trajectory Optimization by Second-Order Cone Programming", *Journal of Guidance, Control, and Dynamics*, volume 39, no. 2, doi:10.2514/1.G001210.

Lockwood, M. K. [2004]. "Neptune Aerocapture Systems Analysis", in *AIAA Atmospheric Flight Mechanics Conference and Exhibit*, American Institute of Aeronautics and Astronautics, Reston, Virigina, pp. 1–16, doi:10.2514/6.2004-4951.

Lu, P. [2014]. "Entry Guidance: A Unified Method", *Journal of Guidance, Control, and Dynamics*, volume 37, no. 3, pp. 713–728, doi:10.2514/1.62605.

Lu, P., Cerimele, C. J., Tigges, M. A., and Matz, D. A. [2015]. "Optimal Aerocapture Guidance", *Journal of Guidance, Control, and Dynamics*, volume 38, no. 4, pp. 553–565, doi:10.2514/1.G000713.

Lyne, J. E. [1994]. "Physiological constraints on deceleration during the aerocapture of manned vehicles", *Journal of Spacecraft and Rockets*, volume 31, no. 3, pp. 443–446, doi:10.2514/3.26458.

Lyons, D. T. [2000]. "Aerobraking at Venus and Mars: A Comparison of the Magellan and Mars Global Surveyor Aerobraking Phases", Technical report, NASA JPL.

Masciarelli, J. and Queen, E. [2003]. "Guidance Algorithms for Aerocapture at Titan", in *39th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, American Institute of Aeronautics and Astronautics, Reston, Virigina, doi:10.2514/6.2003-4804.

Masciarelli, J., Rousseau, S., Fraysse, H., and Perot, E. [2000]. "An analytic aerocapture guidance algorithm for the Mars Sample Return Orbiter", in *Atmospheric Flight Mechanics Conference*, American Institute of Aeronautics and Astronautics, Reston, Virigina, pp. 525–532, doi:10.2514/6.2000-4116.

Menart, J. a. and Henderson, S. J. [2008]. "Equilibrium Properties of High-Temperature Air for a Number of Pressures", *Journal of Thermophysics and Heat Transfer*, volume 22, no. 4, pp. 718–726, doi:10.2514/1.36141.

Miele, A., Wang, T., Lee, W. Y., Zhao, Z. G., Group, A.-a., and Box, P. O. [1990]. "Optimal trajectories for the aeroassisted flight experiment", *Acta Astronautica*, volume 21, no. 11-12, pp. 735–747, doi:10.1016/0094-5765(90)90116-3.

Miyamae, A., Nagata, Y., Ono, I., and Kobayashi, S. [2010]. "Natural Policy Gradient Methods with Parameter-based Exploration for Control Tasks", *Advances in Neural Information Processing Systems 23*, pp. 1660–1668.

Mooij, E. [1997]. *The motion of a vehicle in a planetary atmosphere*, Delft University Press.

Mooij, E. [2014]. "AE 4870B Draft Lecture Notes: Re-entry Systems", Technical report, Delft University of Technology.

Moseley, P. E. [1969]. "The Apollo Entry Guidance: A Review of the Mathematical Development and Its Operational Characteristics", Technical report, TRW, Note 69-FMT-791, Houston, TX.

Moss, J. N., Glass, C. E., and Greene, F. A. [2006]. "DSMC Simulations of Apollo Capsule Aerodynamics for Hypersonic Rarefied Conditions", in *9th AIAA/AS; Thermophysics and Heat Transfer Conference*, number June, San Francisco, CA, pp. 5–8.

Munk, M. M. and Moon, S. A. [2008]. "Aerocapture Technology Development Overview", in *2008 IEEE Aerospace Conference*, IEEE, New York, NY, doi:10.1109/AERO.2008.4526545.

NASA [1965]. "Aerodynamic Data Manual for Project Apollo", Technical report.

NASA [1976]. "U.S. Standard Atmosphere, 1976", Technical report.

NASA [1999]. "Mars Climate Orbiter Mishap Investigation Board: Phase I Report", Technical report.

O'Neil, W. J. and Cazaux, C. [2000]. "The mars sample return project", *Acta Astronautica*, volume 47, no. 2-9, pp. 453–465, doi:10.1016/S0094-5765(00)00085-0.

Pascanu, R. and Bengio, Y. [2014]. "Revisiting Natural Gradient for Deep Networks", in *International Conference in Learning Representations*.

Peng, J. [1993]. "Efficient Dynamic Programming-Based Learning for Control", Ph.D. thesis, Northeastern University, Boston.

Peng, J. and Williams, R. J. [1996]. "Incremental Multi-Step Q-Learning", *Machine Learning*, volume 22, no. 1-3, pp. 283–290, doi:10.1007/BF00114731.

Percy, T. K., Bright, E., and Torres, A. O. [2005]. "Assessing the Relative Risk of Aerocapture Using Probabilistic Risk Assessment", in *Proceedings of the 41st AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, Tucson, pp. 1–15.

Peters, J. and Schaal, S. [2008]. "Natural Actor-Critic", *Neurocomputing*, volume 71, no. 7-9, pp. 1180–1190, doi:10.1016/j.neucom.2007.11.026.

Povey, D., Zhang, X., and Khudanpur, S. [2015]. "Parallel training of DNNs with Natural Gradient and Parameter Averaging", *International Conference in Learning Representations*.

Powell, R. W. and Braun, R. D. [1993]. "Six-Degree-of-Freedom Guidance and Control Analysis of Mars Aerocapture", *Journal of Guidance, Control, and Dynamics*, volume 16, no. 6, pp. 1038–1044.

Prakash, A. and Zhong, X. [2009]. "Numerical Simulation of Planetary Reentry Aeroheating over Blunt Bodies with Non-equilibrium Reacting flow and Surface Reactions", in *47th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition*, Orlando, FL.

Rere, L. R., Fanany, M. I., and Arymurthy, A. M. [2015]. "Simulated Annealing Algorithm for Deep Learning", *Procedia Computer Science*, volume 72, pp. 137–144, doi:10.1016/j.procs.2015.12.114.

Ro, T. and Queen, E. [1998]. "Study of Martian aerocapture terminal point guidance", *23rd Atmospheric Flight Mechanics Conference*, pp. 1–17, doi:doi:10.2514/6.1998-4571.

Robinson, J. S., Wurster, K. E., and Mills, J. C. [2009]. "Entry Trajectory and Aeroheating Environment Definition for Capsule-Shaped Vehicles", *Journal of Spacecraft and Rockets*, volume 46, no. 1, pp. 74–86, doi:10.2514/1.30998.

Rousseau, S. [2001]. "An Energy Controller Aerocapture Guidance Algorithm for the Mars Sample Return Orbiter", in *11th Annual AAS/AIAA Space Flight Mechanics Meeting*, Santa Barbara, CA.

Rousseau, S., Perot, E., Graves, C., Masciarelli, J. P., and Queen, E. [2002]. "Aerocapture Guidance Algorithm Comparison Campaign", in *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, Monterey, CA.

Roux, N. L., Manzagol, P.-a., and Bengio, Y. [2008]. "Topmoumoute online natural gradient algorithm", *Advances in Neural Information Processing Systems 20*, , no. 1299.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. [1986]. "Learning representations by back-propagating errors", *Nature*, volume 323, no. 6, pp. 533–536, doi:10.1038/323533a0.

Samareh, J. A. [2009]. "A Multidisciplinary Tool for Systems Analysis of Planetary Entry, Descent, and Landing (SAPE)", Technical report, NASA, Langley, VA.

Sanchez-Sanchez, C., Izzo, D., and Hennes, D. [2016]. "Optimal Real-Time Landing Using Deep Networks", Technical report, European Space Agency, German Research Center for Artificial Intelligence.

Schmidhuber, J. [2015]. "Deep Learning in neural networks: An overview", *Neural Networks*, volume 61, pp. 85–117, doi:10.1016/j.neunet.2014.09.003.

Schwehm, G. [1989]. "Rosetta — Comet Nucleus Sample Return", *Advances in Space Research*, volume 9, no. 6, pp. 185–190, doi:10.1016/0273-1177(89)90228-7.

Sehnke, F. [2013]. "Efficient Baseline-Free Sampling in Parameter Exploring Policy Gradients: Super Symmetric PGPE", in *Lecture Notes in Computer Science (LNCS)*, volume 8131, pp. 130–137, doi:10.1007/978-3-642-40728-4_17.

Sehnke, F., Osendorfer, C., Rückstieß, T., Graves, A., Peters, J., and Schmidhuber, J. [2008]. "Policy gradients with parameter-based exploration for control", *Lecture Notes in Computer Science (LNCS)*, volume 5163, pp. 387–396, doi:10.1007/978-3-540-87536-9_40.

Sehnke, F., Osendorfer, C., Rückstieß, T., Graves, A., Peters, J., and Schmidhuber, J. [2010]. "Parameter-exploring policy gradients", *Neural Networks*, volume 23, no. 4, pp. 551–559, doi:10.1016/j.neunet.2009.12.004.

Sehnke, F. and Zhao, T. [2015]. "Baseline-Free Sampling in Parameter Exploring Policy Gradients: Super Symmetric PGPE", in *Artificial Neural Networks*, Springer, pp. 271–293, doi:10.1007/978-3-319-09903-3_13.

Serrano-Martinez, J. and Hechler, M. [1989]. "Aerocapture guidance and navigation for the Rosetta Comet Nucleus Sample Return Mission", *16th Atmospheric Flight Mechanics Conference*, doi:doi:10.2514/6.1989-3381.

Shen, Z. and Lu, P. [2004]. "Dynamic Lateral Entry Guidance Logic", *Journal of Guidance, Control, and Dynamics*, volume 27, no. 6, pp. 949–959, doi:10.2514/1.8008.

Shuster, M. D. [1993]. "A Survey of Attitude Representations", *The Journal of the Astronautical Sciences*, volume 41, no. 4, pp. 439–517, doi:10.2514/6.2012-4422.

Sigal, E. and Guelman, M. [2001]. "Aerocapture with minimum total heat", in Intergovernmental Panel on Climate Change (editor), *Proceedings of the 52nd International Astronautical Congress*, Cambridge University Press, Toulouse, France, doi:10.1017/CBO9781107415324.004.

Spencer, D. A. and Tolson, R. H. [2007]. "Aerobraking Cost and Risk", *Journal of Spacecraft and Rockets*, volume 44, no. 6, pp. 1285–1293.

Suratgar, A. A., Tavakoli, M. B., and Hoseinabadi, A. [2007]. "Modified Levenberg–Marquardt Method for Neural Networks Training", *World Academy of Science, Engineering and Technology*, volume 1, no. 6, pp. 1745–1747.

Sutton, K. and Graves, R. a. [1971]. "A General Stagnation-Point Convective Heating Equation For Arbitrary Gas Mixtures", Technical report, NASA, Washington, D. C.

Sutton, R. S. and Barto, G. [2012]. *Reinforcement Learning: An Introduction*, 2nd edition, The MIT Press, Cambridge, MA.

Tauber, M. E. and Sutton, K. [1991]. "Stagnation-point radiative heating relations for Earth and Mars entries", *Journal of Spacecraft and Rockets*, volume 28, no. 3, pp. 43–49, doi:10.2514/3.26206.

Tolson, R. H., Prince, J., and Konopliv, A. [2013]. "An Atmospheric Variability Model for Venus Aerobraking Missions", in *AIAA Modeling and Simulation Technologies (MST) Conference*, American Institute of Aeronautics and Astronautics, Reston, VA, doi:10.2514/6.2013-4830.

Transtrum, M. K. and Sethna, J. P. [2012]. "Improvements to the Levenberg-Marquardt algorithm for nonlinear least-squares minimization", .

Uesugi, K. [1996]. "Results of the MUSES-A "HITEN" mission", *Advances in Space Research*, volume 18, no. 11, pp. 69–72, doi:10.1016/0273-1177(96)00090-7.

Vallado, D. A. [2001]. *Fundamentals of Astrodynamics and Applications*, 2nd edition, Space Technology Library.

Wakker, K. F. [2015]. *Fundamentals of Astrodynamics*, Institutional Repository, Delft University of Technology.

Watkins, C. J. C. H. [1989]. "Learning from Delayed Rewards", Ph.D. thesis, King's College.

Wertz, J. R. [2009]. *Orbits & Constellations Design & Management*, The Space Technology Library, Hawthorne, CA.

Wie, B. [2008]. *Space Vehicle Dynamics and Control*, 2nd edition, American Institue of Aeronautics and Astronautics, Inc., Reston, VA.

Willcockson, W. H. [1999]. "Stardust Sample Return Capsule Design Experience", *Journal of Spacecraft and Rockets*, volume 36, no. 3, pp. 470–474, doi:10.2514/2.3468.

Williams, R. J. [1992]. "Simple statistical gradient-following algorithms for connectionist reinforcement learning", *Machine Learning*, volume 8, no. 3, pp. 229–256, doi:10.1007/BF00992696.

Wingrove, R. C. [1963]. "Survey of Atmosphere Re-Entry Guidance and Control Methods", *AIAA Journal*, volume 1, no. 9, pp. 2019–2029.

Yu, H. and Wilamowski, B. M. [2011]. "Levenberg-Marquardt training", in *Industrial Electronics Handbook, vol. 5 – Intelligent Systems*, doi:doi:10.1201/b10604-15.

Zeiler, M. D. [2012]. "ADADELTA: An Adaptive Learning Rate Method", Technical report, Google Inc., USA, New York University, USA.

Zhou, H., Rahman, T., and Chen, W. [2014]. "Neural Network Assisted Inverse Dynamic Guidance for Terminally Constrained Entry Flight", *The Scientific World Journal*, volume 2014.

Zhou, R. [2002]. "Design of Closed Loop Optimal Guidance Law Using Neural Networks", *Chinese Journal of Aeronautics*, volume 15, no. 2, pp. 98–102, doi:10.1016/S1000-9361(11)60137-4.

Zumwalt, C. H., Sostaric, R. R., Westhelle, C. H., and Cianciolo, A. D. [2010]. "Aerocapture Guidance and Performance at Mars for High-Mass Systems", *Proceedings of the AIAA Guidance, Navigation and Control Conference*, , no. August, pp. 1–15, doi:10.2514/6.2010-7971.