POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

# Control Techniques for Pinpoint Landing on Mars: Performance Enhancement through LQR and Model Predictive Control

Tesi di Laurea Magistrale in
Space Engineering - Ingegneria Spaziale

Author: **Adriano Tessarollo**

Student ID: 247466
Advisor: Prof. Paolo Lunghi
Co-advisors: Ing. Paolo Martella
Academic Year: 2024-25

# Abstract

This thesis investigates different control techniques, within the broader framework of Guidance Navigation and Control (GNC), to perform a pinpoint landing maneuver on Mars.
In recent years, pinpoint landing on celestial bodies such as the Moon and Mars has become one of the most relevant challenges in space exploration. This growing interest is driven by the need, for future missions, to precisely reach predefined landing sites for scientific, logistic, and even human exploration purposes.
This study was carried out in collaboration with Thales Alenia Space - Italy (TAS-I) in Turin, in particular in the project office of the European Space Agency (ESA) mission ExoMars Rosalind-Franklin, which provided the main working material.
Starting from the Proportional–Integral–Derivative (PID) controller as a benchmark solution already implemented by TAS-I, two advanced control techniques are analyzed: Linear Quadratic Regulator (LQR) and Model Predictive Control (MPC). Both techniques are tested under two sensor configurations of the lander: a configuration with three sensors (inertial measurement unit, radar doppler altimeter, landing vision system) and the second one using only altimeter, without radar doppler.
The results show that the LQR improves the most important performance parameters for a pinpoint landing for both configurations, whereas the MPC exhibits limitations in the two-sensor configuration, also due to its higher computational burden and increased tuning complexity.

**Keywords:** Mars Landing, Pinpoint Landing, Guidance Navigation and Control (GNC), Proportional-Integral-Derivative (PID), Linear Quadratic Regulator (LQR), Model Predictive Control (MPC).

# Sommario

L'obiettivo di questa tesi è di investigare diverse tecniche di controllo, nel quadro più ampio di Guida Navigazione e Controllo (GNC), per eseguire una manovra di atterraggio di precisione su Marte.
Recentemente, l'atterraggio di precisione su altri corpi celesti, come la Luna o Marte, è diventata una delle sfide più importanti per l'esplorazione spaziale. Questo crescente interesse è guidato dal desiderio, per le missioni future, di raggiungere con precisione siti di atterraggio predefiniti per motivi scientifici, logistici, o anche per l'esplorazione umana.
Questo studio è stato realizzato in collaborazione con Thales Alenia Space - Italia (TAS-I) a Torino, in particolare nel project office della missione ExoMars Rosalind-Franklin dell'ESA (European Space Agency), che ha fornito il principale materiale di lavoro.
Partendo da un controllore Proporzionale-Integrale-Derivativo (PID), implementato da TAS-I, come termine di paragone per i risultati, due tecniche di controllo avanzate sono analizzate: il Linear Quadratic Regulator (LQR) e il Model Predictive Control (MPC). Entrambe le tecniche sono testate in due diverse configurazioni di sensori per il lander: una con tre sensori (unità di misura inerziale, radar doppler altimetro, sistema di visione in atterraggio) e la seconda con solo l'altimetro a disposizione, senza il radar doppler.
I risultati mostrano che il controllore LQR migliora i parametri di performance più importanti relativi all'atterraggio di precisione per entrambe le configurazione, mentre il controllore MPC mostra limitazioni nella configurazione con due sensori, anche a causa del suo alto carico computazionale e della sua maggiore complessità di tuning.

**Parole chiave:** Atterraggio su Marte, Atterraggio di Precisione, Guida Navigazione e Controllo (GNC), Controllo Proporzionale-Integrativo-Derivativo (PID), Linear Quadratic Regulator (LQR), Model Predictive Control (MPC).

# Contents

# List of Acronyms

| | |
|---|---|
| **ALO** | ALtimeter Only configuration |
| **AOCS** | Attitude and Orbit Control System |
| **ASDS** | Autonomous Spaceport Drone Ship |
| **CNSA** | Chinese Space Agency |
| **EDL** | Entry-Descent-Landing |
| **EDM** | Entry, Descent and landing Demonstrator module |
| **ESA** | European Space Agency |
| **G-FOLD** | Guidance for Fuel Optimal Large Divert |
| **GNC** | Guidance Navigation and Control |
| **GPS** | Ground Positioning System |
| **HVAC** | Heating Ventilation and Air Conditioning |
| **IMU** | Inertial Measurement Unit |
| **JPL** | NASA Jet Propulsion Laboratory |
| **LQR** | Linear Quadratic Regulator |
| **LUT** | LookUp Table |
| **LVS** | Landing Vision System |
| **MC** | Monte Carlo |
| **MPC** | Model Predictive Control |
| **MSL** | Mars Science Laboratory |
| **NASA** | National Aeronautics and Space Administration |
| **NMPC** | Non-linear Model Predictive Control |
| **PID** | Proportional-Integral-Derivative |
| **PD** | Proportional-Derivative |
| **PWD** | Pulse-Width-Modulation |
| **RCS** | Reaction Control System |
| **RDA** | Radar Doppler Altimeter |
| **SOCP** | Second Order Conical Programming |
| **SW** | Software |

| | |
|---|---|
| **TAS-I** | Thales Alenia Space Italy |
| **TD** | Touchdown |
| **TGO** | Trace Gas Orbiter |

# List of Symbols

| Variable | Description | SI unit |
|---|---|---|
| $\dot{\mathbf{x}}$ | Derivative of the state vector | - |
| $\mathbf{x}$ | State vector | - |
| $\mathbf{u}$ | Control vector | - |
| $L$ | Cost function | - |
| $t$ | Time | $s$ |
| $t_f$ | Final time | $s$ |
| $\mathbf{x}_r$ | Reference state | - |
| $\mathbf{u}_r$ | Cost reference | - |
| $\mathbf{A}$ | State dynamics matrix | - |
| $\mathbf{B}$ | Control matrix | - |
| $\mathbf{Q}$ | State cost matrix | - |
| $\mathbf{R}$ | Control cost matrix | - |
| $\mathbf{S}$ | Solution of algebraic Riccati equation | - |
| $\mathbf{u}^*$ | Optimal control solution | - |
| $\mathbf{K}$ | State-feedback gain matrix | - |
| $\Delta t$ | Sample time | $s$ |
| $\mathbf{e}_p$ | Error on position | $m$ |
| $\mathbf{e}_v$ | Error on velocity | $m/s$ |
| $\mathbf{p}_r$ | Reference position | $m$ |
| $\hat{\mathbf{p}}$ | Estimated position | $m$ |
| $\mathbf{v}_r$ | Reference velocity | $m/s$ |
| $\hat{\mathbf{v}}$ | Estimated velocity | $m/s$ |
| $\mathbf{a}_c$ | Commanded translational acceleration | $m/s^2$ |
| $k_{i,i}$ | Component of the gain matrix $\mathbf{K}$ | - |
| $\mathbf{a}_r$ | Reference translational acceleration | $m/s^2$ |

| Variable | Description | SI unit |
|---|---|---|
| $\mathbf{a}_t$ | Total translational acceleration | $m/s^2$ |
| $\hat{\mathbf{a}}_t$ | Total translational acceleration implemented | $m/s^2$ |
| $Q_{p_j}$ | Position component of matrix $\mathbf{Q}$ | - |
| $Q_{v_j}$ | Velocity component of matrix $\mathbf{Q}$ | - |
| $R_j$ | Component of matrix $\mathbf{R}$ | - |
| `Z_bp` | Altitude breakpoints vector | $m$ |
| $\sigma$ | Weights of the $\mathbf{Q}$ matrix | - |
| $\mathbf{e}_a$ | Angular error vector | $rad$ |
| $\mathbf{e}_w$ | Angular velocity error vector | $rad/s$ |
| $\boldsymbol{\alpha}$ | Angular acceleration | $rad/s^2$ |
| $\mathbf{M}$ | Inertia matrix | $kg\,m^2$ |
| $\mathbf{I}_3$ | Identity matrix $3 \times 3$ | - |
| $\boldsymbol{\tau}$ | Control torques | $Nm$ |
| $N_s$ | Number of prediction steps | - |
| $N_m$ | Number of control steps | - |
| $p_H$ | Prediction Horizon | $s$ |
| $\mathbf{X}$ | State constraints set | - |
| $\mathbf{U}$ | Control constraints set | - |
| $\dot{\mathbf{r}}$ | Derivative of position vector | $m/s$ |
| $\mathbf{a}_{cf}$ | Centrifugal acceleration | $m/s^2$ |
| $\mathbf{F}$ | Force vector | $N$ |
| $\boldsymbol{\omega}_M$ | Mars angular velocity | $rad/s$ |
| $\mathbf{a}_{cor}$ | Coriolis acceleration | $m/s^2$ |
| $\mathbf{g}_M$ | Mars gravitational acceleration | $m/s^2$ |
| $\mathbf{q}$ | Quaternion | - |
| $\dot{\mathbf{q}}$ | Quaternion derivative | - |
| $\boldsymbol{\Omega}(\boldsymbol{\omega})$ | Angular velocity skew matrix | $rad/s$ |
| $\dot{\boldsymbol{\omega}}$ | Angular acceleration | $rad/s^2$ |
| $I_{ii}$ | Principal moment of inertia | $kg\,m^2$ |
| $T$ | Thrust magnitude | $N$ |
| $m$ | Mass | $kg$ |
| $\dot{m}$ | Mass flow rate | $kg/s$ |

| Variable | Description | SI unit |
|---|---|---|
| $I_{sp}$ | Specific impulse | $s$ |
| $g_0$ | Standard gravitational acceleration | $m/s^2$ |
| $\mathbf{x}^p$ | Predicted state vector | - |
| $\hat{\mathbf{x}}_k$ | Estimated initial state | - |
| $\mathbf{u}_0$ | Unit delay initial condition | - |
| $\mathbf{s}$ | Set of reference predicted states | - |
| $\bar{\boldsymbol{\omega}}$ | Average angular velocity | $rad/s$ |
| $\mathbf{E}(\mathbf{q})$ | Quaternion kinematic mapping matrix | - |

# 1 | Introduction

## 1.1. Mars exploration and precision landing

The exploration of Mars has recently become one of the most challenging adventures in the space sector. The study of Mars allows us to further enhance the knowledge of our solar system and how planets form and evolve over time. It has already been discovered that liquid water was abundant on Mars in the past, with rivers and lakes.

Now, research is going on, and future missions are already being carried on to further study the red planet, with the long-term goal of enabling human exploration.

Different strategies are being studied and have been used to land on Mars, from the early 1960s missions related to the cold war, to the current developing missions like ExoMars, now called Rosalind Franklin mission, which has the aim of using a drill to study Mars soil composition, or Mars Sample Return which is a mission announced by ESA and NASA to take samples of rocks and terrain and to bring it back to Earth in order to analyze it, and many others.

The main technological challenge to face now is the transition from the so called ellipse landings to the precision, also known as pinpoint, landing.
The ellipse landings are characterized by a large landing area, which represents the uncertainty propagation of the landing position on an extraterrestrial body (Moon or other planets) where the lander is supposed to touch ground. These areas can range from hundreds of kilometers to some kilometers, still being quite large. The accuracy in this technology has improved from the 1960s till today, but still is not enough for some purposes. Just considering the idea of building a base makes the problem clear, since if there are objects or materials to bring to a specific point, the landers, carrying to Mars these objects, could not land kilometers far from the target landing point. It would be difficult or impossible to move heavy materials for kilometers, and the problem gets even worse thinking about humans landing kilometers away from the target. Another issue is related to hazard avoidance, as, in order to avoid rocks or slopes while landing, the accuracy becomes crucial. This is why pinpoint landing is one of the most important challenges when

looking at missions that involve the exploration of other celestial bodies. With precision landing the aim would be to have an accuracy in the landing position of some tens of meters from the target, that would be very convenient in the scenarios described before.

There are many issues that have to be faced for the pinpoint landing problem, starting from the delay in communication from Mars to Earth, that forces the system to be completely autonomous in landing, hence to have a perfectly tested Guidance, Navigation and Control subsystem; as well as the environmental problems, having to land in a rocky and sometimes steep terrain so possibly having to deal with slopes and hazard avoidance to dodge dangers.

## 1.2. Scope of the thesis

This study has been carried on in collaboration with *Thales Alenia Space Italy (TAS-I)* in Turin, and in particular the focus of this work is in the control subsystem during the landing part, so in the big picture of EDL (Entry-Descent-Landing), the study of the thesis is about the control section of GNC (Guidance Navigation and Control). The baseline is a GNC simulator in Matlab/Simulink environment gently given by TAS-I, where Guidance, Navigation, and Control are already built having as benchmark the ExoMars Rosalind Franklin mission, but with the aim of pinpoint landing. The objective of the work is to improve, study and test the Control subsystem. Starting from a PID Controller in the baseline simulator, the steps are to implement a Linear Quadratic Regulator (LQR) controller initially only on translation and then integrated both in translation and rotation, and then to implement a Model Predictive Control (MPC) logic to see the performance changes and perform a trade-off analysis between the various controllers.

For the environment considered in this work, relevant landings to be cited are mainly the ones aiming to Mars, and not to the Moon, since the two scenarios present different challenges and, as a consequence, need different technologies. The main difference is given by the presence of atmosphere on Mars, that even if it has a density that is about 1/100 of the one on Earth it allows the use of parachutes to slow down the velocities during the descent phase of the lander in the martian atmosphere, and this has been done for all the past Mars landing missions. This strategy is not usable in order to land on the Moon, which is the reason why only retro-propulsion is used to safely land on our satellite.
An additional advantage given by the presence of atmosphere is the possibility to use aerodynamic surfaces, exploiting the atmospheric lift and drag in order to control the lander during the Entry phase.
Another important difference between the Moon and Mars is the difference in gravity

force, since Mars gravity is more than double that the one on the Moon. In this trade-off in which Mars and Moon are compared, note that the possibility to apply parachutes to perform landing corresponds to a propellant saving whilst the compensation of the gravity represents a propellant penalty.

The following chapter will present a brief overview of the precision landing problem in terms of state of art, technologies involved, missions related to the landing problem, as well as a short introduction to the Model Predictive Control and its use, together with a short presentation of the EDL and GNC framework. Subsequently, the baseline GNC simulator provided by TAS-I is presented. Then there will be the two main chapters that present the core of the thesis, dedicated to the design and implementation of the LQR control and the MPC, with the related results. Finally, the last two chapters draw the conclusions and suggest future developments.

## 1.3. History of autonomous Mars landings

Mars exploration began with the space run during the cold war between USA and URSS, in late 1960s. The first human-made object to touch the martian ground was the *Mars 2* from URSS, landed on 27 November 1971, even if the touchdown was too fast, leading to a crash of the lander on the terrain [1]. Just some days later, URSS performed a successful soft landing on 2 December 1971 with mission *Mars 3*, but the lander transmitted only for a few seconds, then the communications fell for unknown reasons [2].
The first ever mission to be considered a full success on Mars landed on 20 July 1976, the USA mission *Viking 1*, immediately followed by *Viking 2* on 3 September 1976 [3]. The first was operative for 6 years, while the second for 3 years and their objective was to send the first photos of the martian ground, as well as to measure the atmosphere and soil composition [3]. These two missions have been the landmark of a successful EDL procedure completely achieved by an onboard autonomous system .

After years of disinterest in Mars exploration, in 1997 landed the *Mars Pathfinder* from USA, a milestone mission for the red planet studies, since it was composed of both a lander and a wheeled rover called Sojourner, first rover in history, demonstrating that Mars can also be explored "walking" [4]. The mission accomplished the objectives of analyzing different rocks, the Martian soil and the atmosphere, lasting almost 3 months. The Pathfinder EDL sequence consisted of thermic shield, parachutes and during the landing phase the impact was protected with inflatable airbags [4], which was the new main technology tested, since all the landings tried and succeeded before used just parachutes and retropropulsion.

From the 2000s the rover era on Mars began, with the missions *Spirit Rover* and *Opportunity Rover* in 2004 from NASA. Both of them used again airbag technology to land. They were supposed to last 90 days, but Spirit lasted 6 years, while Opportunity has the longevity record having operated for 14 years. It is thanks to these two rovers that it was discovered that, long ago, on Mars there were lakes and rivers, and so liquid water [5] [6]. In 2008 the *Phoenix lander* of NASA arrived near Mars north pole and discovered the presence of iced water under the soil. For this rover, the landing procedure no longer relied on airbags, instead a retropropulsion system was used [7].
Later, in 2012, the *Curiosity* rover landed with the mission *Mars Science Laboratory (MSL)*, which is still operational and which has provided us with some of the most important information we have on Mars, about its possible past habitability, and contributed to gain knowledge on how to perform a manned mission on the red planet [8]. New landing technologies were used with Curiosity in combination with parachute, first of all the sky crane [9], a platform designed to lower the rover using a tethered descent stage. During the rover descent to the ground, the platform is suspended in the air thanks to the retropropulsion compensating gravity. When the rover touches the ground, the sensors send a message to the descent stage that performs a fly-away maneuver in order to make the platform crashing far from the rover itself, about a kilometer away, to keep the landing site intact and uncontaminated by exhaust gas or debris. Another technology successfully tested was the Guided Entry. In summary, this technology corresponds to creating an unbalance between the center of mass and the center of pressure. In this manner, the controlled modification of the bank angle, identifiable in first approximation as the angle around the symmetry axis, determines a modulation of the lift and drag, and so the possibility of regulating the cross range and down range with respect to a predefined point. To achieve this unbalance some balance masses are jettisoned, just after the entry interface point, then at the end of the entry phase some other masses on the opposite side of the lander are also jettisoned, so re-establishing a symmetry in the mass distribution before the landing phase, as reported in [10].
In 2018 the *InSight* mission was accomplished with parachute and retropropulsion, again by the USA, it was a stationary lander to investigate the seismic behavior of the planet, and it revealed active tectonic [11].
The following landing on Mars was performed in 2021, by the NASA mission *Mars 2020* with the rover called *Perseverance*, which was also carrying the helicopter *Ingenuity*, the landing was performed as for Curiosity with sky crane and balance mass technologies. In Fig. 1.1 and Fig. 1.2 the landing sequence of Perseverance is presented, that is similar to every EDL procedure on Mars, as well as a representation of the sky crane.
The Perseverance rover is still operational, testing technologies for humans and searching

for past microbial life [12]. Ingenuity has performed more than 70 flies on Mars, it was the first controlled aircraft on another planet, and became inoperative in 2024 [13].

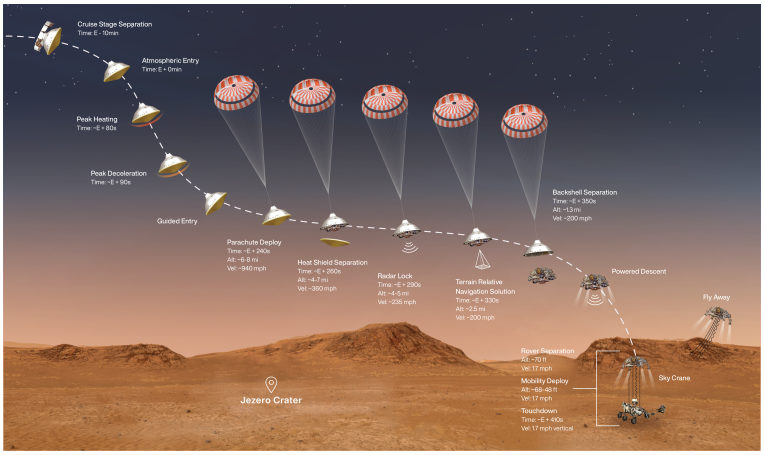


Figure 1.1: Perseverance autonomous landing sequence. Credits: NASA/JPL-Caltech.

Figure 1.2: Sky Crane technology.

The last landing mission is *Tianwen-1* again in 2021 with the *Zhurong* rover, the first and already successful attempt by China using only parachute and retropropulsion. It operated for a year with the aim of studying the atmosphere, soil, searching water, and mapping the topography of the surface [14]. The Chinese mission adopted a different strategy to perform aerodynamic control during entry, instead of jettison balance masses, it used a trim flap system to guide the lander during the entry phase, before opening the parachute. This choice was taken because studies showed that using a trim flap of some kilograms allowed to reach the same objective of jettisoning hundreds of kilograms of balance masses, saving useful mass for the lander [15].

| Mission | Year | Landing accuracy | Country |
|---|---|---|---|
| Mars 3 | 1971 | ∼100 km | USSR |
| Viking 1 | 1976 | ∼280 × 100 km | USA |
| Viking 2 | 1976 | ∼280 × 100 km | USA |
| Mars Pathfinder | 1997 | ∼100 × 20 km | USA |
| Spirit | 2004 | ∼65 × 9 km | USA |
| Opportunity | 2004 | ∼65 × 9 km | USA |
| Phoenix | 2008 | ∼80 × 20 km | USA |
| Curiosity | 2012 | ∼20 × 7 km | USA |
| InSight | 2018 | ∼85 × 24 km | USA |
| Perseverance | 2021 | ∼8 × 7 km | USA |
| Tianwen-1 | 2021 | ∼56 × 22 km | CHINA |

Table 1.1: Past Mars landings

In the table above, Tab. 1.1, there is a summary of all successful landings on Mars with their relative landing accuracy, which is represented by the landing ellipse, which is the area on a body's surface where the lander is expected to land with a probability of 99.7%, that means a confidence of $3\sigma$.

With Figure 1.3, from [16], it is possible to appreciate how the landing precision has grown over the years, with the development of technologies, but it still has areas of square kilometers. The implementation of active guidance, together with more accurate EDL systems, allowed the reduction of the dimensions of the ellipses.

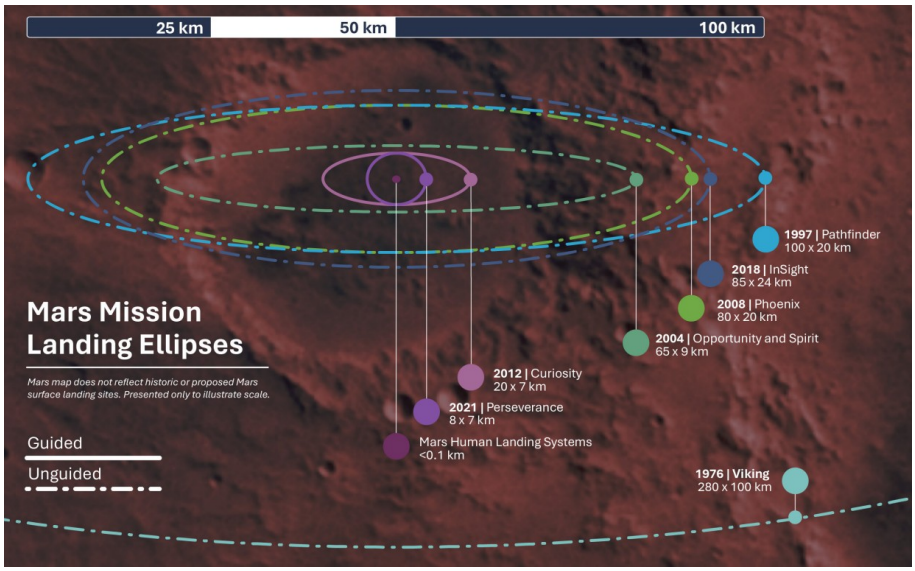


Figure 1.3: Landing ellipses of NASA missions. Credits: ESA/DLR/FU-Berlin/NASA/JPL-Caltech, [16].

## 1.4. Future missions on Mars

Other missions with the aim of landing on Mars have been announced and are in progress by different space agencies, among these there is indeed the ExoMars - Rosalind Franklin Mission by ESA.
This work has been carried on by working also in the project office of the aforementioned mission in the TAS-I site in Turin, since Thales Alenia Space is the prime contractor for the mission ExoMars, having the task of design, develop and verify, among the others, the GNC system of the spacecraft during the Cruise phase and during the Entry, Descent and Landing (EDL) phase. The first part of the ExoMars mission has been launched in 2016, with the Entry, Descent, and Landing Demonstrator Module (EDM) *Schiaparelli*, which was supposed to arrive in a landing ellipse of $115 \times 25$ km. The mission reached its

main objective, i.e. the delivery at Mars of the Trace Gas Orbiter (TGO), the European Mars satellite, provided with payloads for remote science and with an antenna for data relay to the Earth. Schiaparelli successfully achieved the re-awakening from hibernation, the coasting entry and descent phases but failed the landing for a wrong setting of the IMU (Inertial Measurement Unit) saturation flag during the parachute opening. The second part of the ExoMars mission is supposed to be launched in 2028 [17]. Other future missions that are in progress and should be launched in the next years towards the red planet are:

- *Icebreaker Life* mission by NASA, scheduled for 2026, that will send a stationary lander with a drill to study ice and to search for biosignatures.
- *Mars Sample Return*, already cited, is a joint project between NASA and ESA that comprehends a series of missions to collect samples of dust and rocks in Mars soil and bring them to Earth to study and analyze them. The campaign has not started yet [18].
- *Tianwen-3* is a mission programmed by the Chinese Space Agency (CNSA), and it has the same purpose as the Mars Sample Return, so to bring samples from Mars terrain to Earth. The proposed date for the launch is in 2028, and the collected samples should arrive at Earth in July 2031.

## 1.5. History of control techniques adopted

The controller in a landing mission is crucial in order to compute the forces and torques needed to follow the trajectory and safely land on ground.
Historically, the first Entry, Descent, and Landing sequences aimed only to survive the landing, using pre-programmed sequence (open-loop control) without an actually active control of the trajectory. Then, later in time, closed-loop control was considered in order to track a pre-computed (off-line) trajectory. In more recent times, on-line adaptation of the path to be followed in order to perform the landing has begun to be used, up to considering more advanced techniques like Model Predictive Control.
In this section, a short overview of the past control landing techniques used for the EDL sequence on Mars is performed, going through the aforementioned missions and highlighting the benefits introduced by each new technique. Like on Earth, Mars atmospheric entry vehicles can be grouped in three configurations: ballistic, ballistic-lifting and lifting configuration.

All the missions relative to a Mars landing, from the soviet's Mars 2 and Mars 3 missions

up to the Mars Phoenix lander (in 2008), had an open-loop ballistic control, also known as ballistic entry. It is a control where the spacecraft practically follows a trajectory ruled only by gravity and aerodynamics [19].
Hence, all the aforementioned NASA missions like Viking 1, Viking 2, Mars Pathfinder, as well as the Spirit and Opportunity rovers, had exploited this control strategy to safely land on the surface of Mars [19]. Therefore, in these first generation missions the entry in the martian atmosphere was performed without exploiting aerodynamic lift.
Actually, Viking capsules had a center of gravity off-set with respect to the center of pressure, which led to a small lift generation, and so the Viking missions presented a ballistic-lifting configuration; however, the latter was not used to actively control the trajectory. Moving the center of mass was done only for attitude stability [19], and control actions were applied using a reaction control system (RCS) that used a three-axes stabilized attitude control technique. In the Viking missions, there was no guidance function inside the GNC system, so the entry trajectory was uncontrollable.
The other missions, until MSL, used unguided ballistic entry to land. Phoenix, for example, used it to achieve three-axes stabilized attitude control and improve accuracy in parachute deployment [19]. This control strategy during landing provided very large landing ellipses, in the order of hundreds of kilometers as anticipated in 1.1.
Ballistic open-loop control has the main advantage of simplicity and robustness: less active components means less risk of failures with lower aerodynamic heating and shorter flight time. However, the disadvantages are several as well: firstly, a larger peak heat flow, but the most important problem is that the uncertainty in the landing position is enormous and this force the landing area to be large and flat, with the lowest number of obstacles as possible. The accuracy provided by the ballistic today is not sustainable for future ambitions, for which advanced EDL techniques are needed in order to land in hazardous sites with higher scientific value, like could be in the Southern hemisphere of the red planet, which has not been reached yet.

The turning point happened with the NASA mission Mars Science Laboratory and its rover Curiosity. The MSL spacecraft had a ballistic-lifting configuration, with the difference from Viking missions being that this configuration was exploited to obtain a certain angle of attack through the lift produced by the offset between the center of gravity and the center of pressure. This was the first mission to successfully use a closed-loop control in order to track a pre-computed trajectory during the EDL phase. Curiosity took advantage of an aeroshell with a lifting body together with thrusters to control the attitude and therefore the lift direction during the landing sequence. The active guidance used in the MSL mission was derived from the Apollo program [20] [19], and the attitude control

was again performed, as for Viking and Phoenix, with a three-axes stabilized continuous feed-back controller to follow the reference trajectory; in particular, a PD controller and a pure D controller for rate damping [21].
The architecture of the entry attitude control is shown with the block scheme in Fig. 1.4:

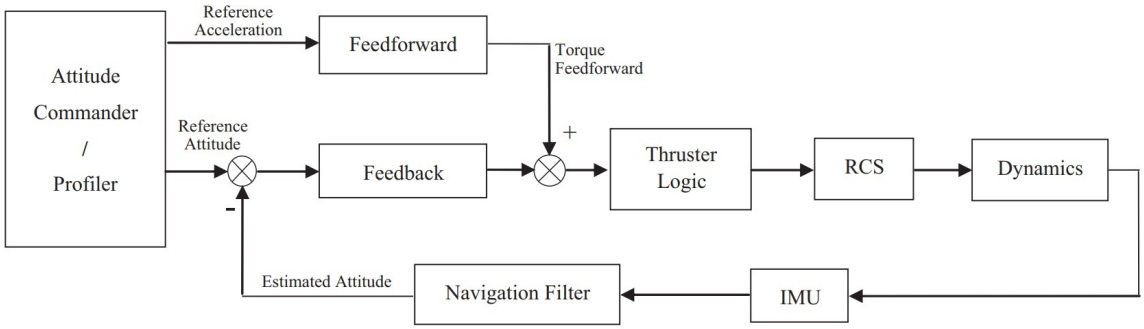


Figure 1.4: Architecture of MSL attitude control during entry [19].

With a ballistic-lifting configuration, landing accuracy improves significantly and the maximum dynamic pressure overload is relatively low with respect to the ballistic configuration.
The result is clear when looking at Tab. 1.1, where it is possible to observe that landing accuracy improved from ellipses of hundreds of kilometers to a $20 \times 7\,km$ ellipse.
Eventually, the lifting configuration can produce a large lift that allows long-distance maneuver to be performed in order to recover the horizontal error accumulated during the entry with respect to the target, being in this way more suitable for next-generation landing vehicles aiming at pinpoint landing [19].

As for the InSight mission, the entry phase of the EDL was ballistic and, therefore, unguided and not spin stabilized. Once the entry phase was complete, the heatshield was jettisoned, the lander legs were deployed, the radar altimeter was activated, and the lander separated from the backshell and began powered flight. During the descent part of the EDL, the lander performed a gravity turn maneuver to get closer to the landing site, where it descended at a constant velocity until touchdown, then the motor was turned off [22].
The last NASA mission regarding landing on the red planet is the already cited Mars 2020, with the rover Perseverance. The Perseverance EDL sequence and control technique were inherited directly by the MSL mission, with two new EDL features: Range Trigger and Terrain-Relative Navigation (TRN) [23].
The Range Trigger is a technology for which the parachute deployments event is related to the range and no more to the velocity as done in the MSL mission. This allows a

significant reduction of the error dispersion at deploy, and hence the landing ellipse is reduced as well. The TRN instead has the goal of choosing a precise landing position for a safe touchdown; for Mars 2020 it was made up of two subsystems: the Landing Vision System (LVS) and the Safe Target Selection (STS) [23].
In the following figure, it is possible to observe the different NASA landing mission on Mars configurations, with the related landing technology.

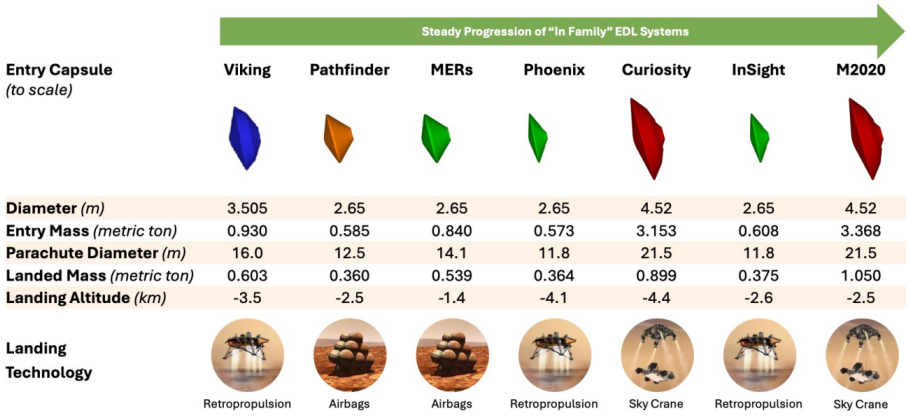


Figure 1.5: Evolution of Mars EDL systems [16].

Eventually, Tianwen-1 mission from China during the EDL phase implemented the GNC subsystem depicted in Fig. 1.6.

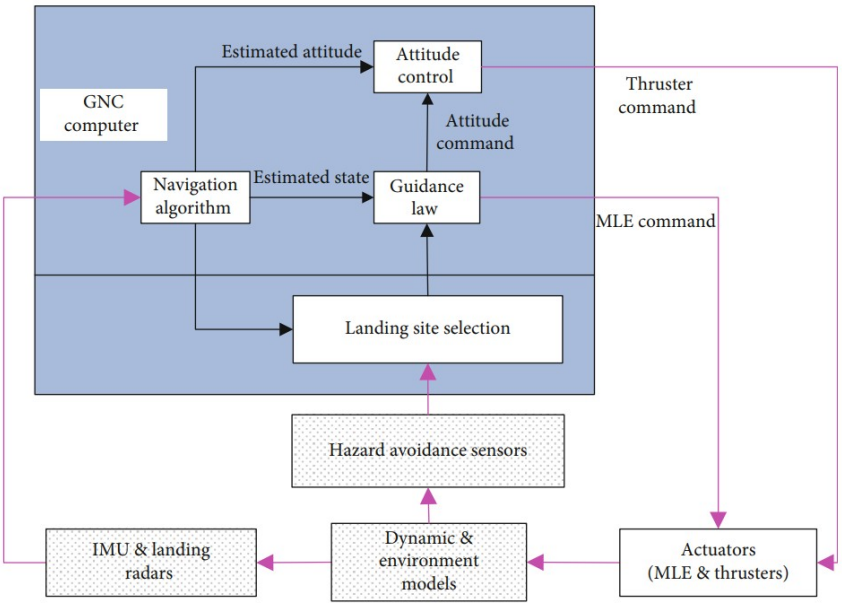


Figure 1.6: Tianwen EDL GNC scheme [24].

In particular, the attitude control is mainly built as a PID controller with Pulse-Width-

Modulation (PWD) technique. This control technique is strictly related to the need to handle hazard avoidance and hover mode in the final part of the landing.

In more recent years, algorithms for optimal guidance and control have been developed in order to perform a planetary landing. One of the best known is the G-FOLD technique (Guidance for Fuel Optimal Large Divert), which sets the problem as a constrained optimal control problem, solving it using convex optimization methods [25]. This is a technique developed by NASA JPL for future missions, such as Mars Sample Return.

One of the control techniques studied and analyzed recently is Model Predictive Control, which is indeed a matter of study of this thesis, and it is introduced in the next section.

## 1.6. Introduction to Model Predictive Control

The most recent missions to Mars could exploit a much more increased computational capability compared to the first missions in the 1960s. This increased capability has allowed the introduction of algorithmic strategies inherently more promising from the performance perspective, but also much heavier for the computation burden.

One of the fundamental problems to be solved to achieve a pinpoint landing is the planning of a trajectory, fulfilling the system constraints, able to reach the target from a quite wide range of initial positions and velocities. Enhanced computation capabilities allow solving algorithms including quadratic optimization and iterative approaches in relatively fast times, compatible with the kinematics of a landing capsule. Among these approaches, it can be highlighted, for instance, the one introduced in 2007-2008 by Behçet Açıkmeşe [26]. This approach exploits lossless convexification to obtain a convex problem and solve it through Second Order Conical Programming (SOCP). Hence, it is possible to plan divert maneuvers with the capability to recover errors to the target up to 3-4 km, making pinpoint landing feasible.

Thanks to the studies aforementioned, the classical optimal control, like fuel-optimal powered descent, that had already been formulated in past years, became computable and reliable hence exploitable in real missions.
In this scenario, Model Predictive Control (MPC) takes place, which is one of the control techniques that are going to be used in this thesis. In a few words, MPC is an optimal control solved repeatedly. It minimizes a cost function over a finite, receding horizon of time, so being able to compute an action not only locally optimized, but looking forward the future evolution of the system.
Receiving the estimations on the current state, the MPC computes the control actions

that have to be applied in order to minimize the cost function over the horizon by solving a constrained optimization problem, then only the first of these control actions is applied, and at the following step the procedure is repeated moving the horizon one step forward.

MPC was introduced in the literature during the late 1980s, with some first evidence also during the 1970s. Its first applications were industrial, focusing mainly on power plants and petroleum refineries, used to replace classical control techniques such as PID [27].
Today MPC is used in many different sectors, from HVAC systems (Heating, Ventilation, and Air Conditioning) [28] to the biomedical sector, for example for insulin infusion in artificial pancreas [29], as well as for automotive, power electronics, and others.

MPC applications also extend to the space sector, with studies being carried out in recent years. Some examples are MPC used for attitude tracking with reaction wheels actuators [30]; some studies focused on MPC used for spacecraft formation flying with sensing noise [31], others on a rendezvous maneuver performed with MPC [32] [33].

Regarding landing applications, MPC has also been studied for aeronautic applications such as in the thesis reported in [34]. Moreover, works related to space applications for planetary landings have already been done, studying how the GNC subsystem could take advantage of the MPC architecture for pinpoint landing [35], and studying how this control technique could be used to land reusable launchers [36] [37] [38].
In particular, studies related to model predictive control have also been done on its possible use in order to perform a Mars landing. Some of these studies are, for example, on how to take advantage of nonlinear model predictive control to land in high elevation sites [39], MPC used to trace a fuel-optimal trajectory [40] or to perform a glided entry with a spaceplane aiming at pinpoint landing [41].
To conclude, MPC is one of the most avant-garde control techniques studied recently, which could lead to significant technological improvements, also for the scientific purpose of pinpoint landing.

In the next section, the EDL sequence framework of this thesis is analyzed.

## 1.7. EDL sequence framework

The simulator used in this work is based on the ExoMars Rosalind-Franklin mission, and the improvements introduced with respect to the baseline model can be highlighted by comparing the respective Entry, Descent, and Landing (EDL) sequences.
In Fig. 1.7 the ExoMars EDL procedure is presented.

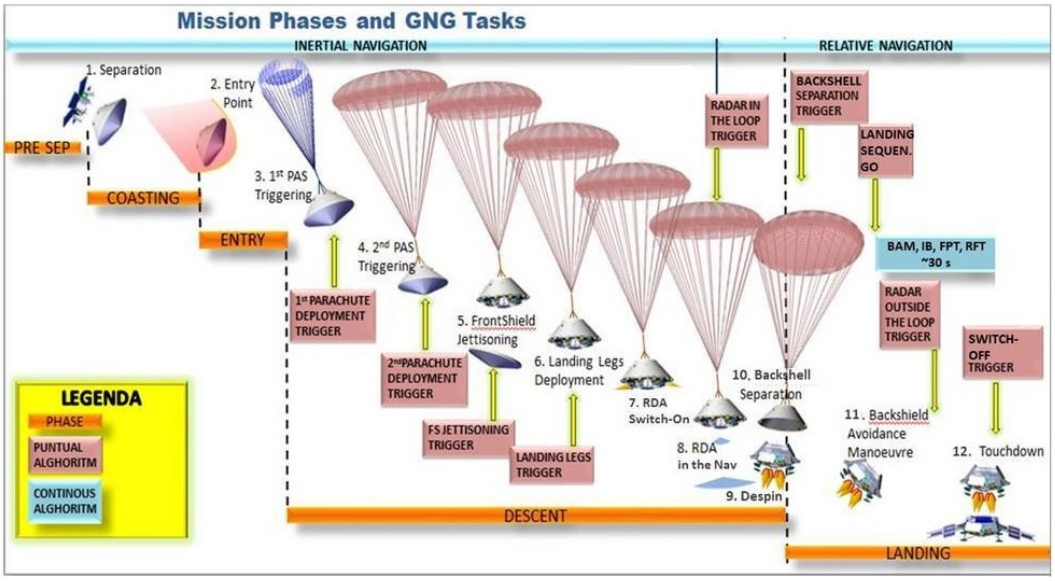


Figure 1.7: ExoMars EDL sequence. Credits: Thales Alenia Space - Italy.

The Entry phase utilizes a heat shield (or front shield) to protect the spacecraft from the high thermochemical load, and during this part of the landing, no control is adopted and it is defined as Ballistic Entry. Later on, during the descent part, the velocity slows down thanks to aerodynamic drag and the parachutes, that are two in the ExoMars mission, are deployed. Then the front shield is jettisoned, landing legs are deployed, the lander is de-spinned, and the backshield with the parachutes attached is separated, a backshield avoidance maneuver is performed and the lander touches ground.
The instruments used in GNC to perform the landing are the Radar Doppler Altimeter (RDA), which is activated at a certain altitude from the ground, to measure the velocity and altitude, and an IMU for the attitude. As already said, this landing strategy does not allow high accuracy in position, due to the error accumulated during the entry and descent phases.

Instead, in Fig. 1.8 the EDL for the simulator used in this work is depicted.
The main differences are the following.

- **Guided Entry assumption**: The initial horizontal error of the lander from the target is assumed to be in a radius of 3 km as achievable in case an active guided entry strategy controls the lander during the entry phase..
- **Instruments**: Not only RDA and IMU, but also a Landing Vision System (LVS) is used, which enables the localization of the lander with respect to the target by exploiting cameras that acquire images of the ground and compare them to a database.

- **Divert Maneuver**: Instead of a simple backshield avoidance maneuver, a divert maneuver is performed; hence the lander in the last 2 kilometers of altitude tries to recover the error in position with respect to the target accumulated during the previous phases of EDL.

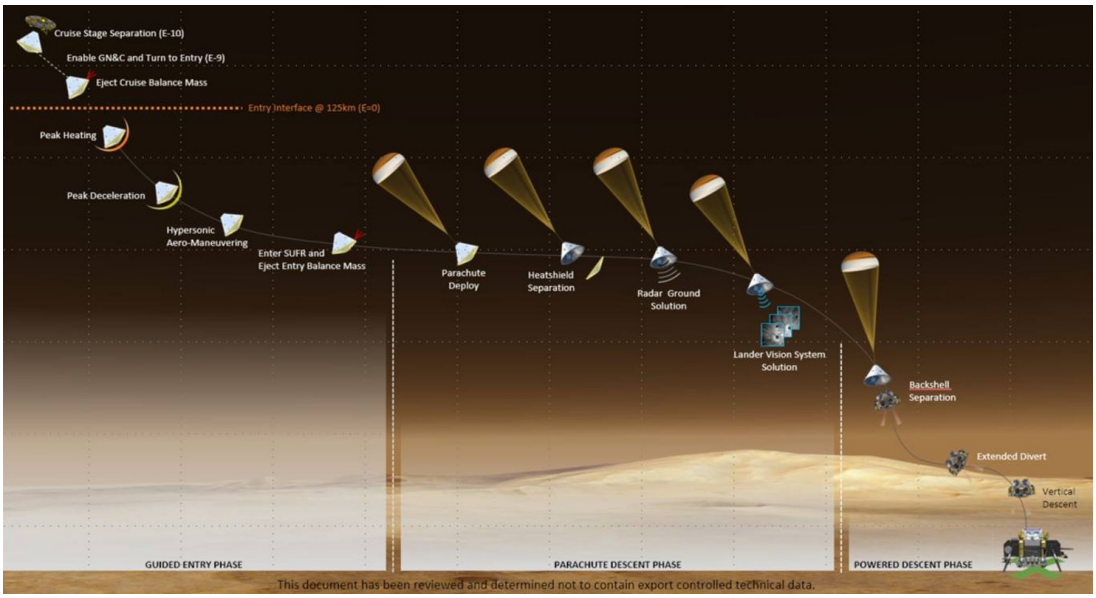


Figure 1.8: Mars Lander simulator EDL sequence. Credits: NASA JPL.

### 1.7.1. EDL sequence on Earth comparison

It is worth to spend some words to highlight the main difference between an EDL sequence on Mars and on Earth.
The first space missions to perform EDL on Earth, hence the Re-Entry from space, were using conic capsules with thermic shields, and the re-entry phase was essentially ballistic. The precision in landing was not a priority with respect to thermal and structural security. Missions like these were, for example, NASA's Project Mercury and Project Gemini during the late 1950s and early 1960s [42] [43].
With the Space Shuttle program, things changed, having a spaceplane with wings capable of exploiting Earth atmosphere to produce aerodynamic forces that lead to a controlled entry into Earth atmosphere and allow precise landing due to a long gliding phase in atmosphere. This was performed, for example, to precisely land in Kennedy Space Center in Florida or in Edwards Air Force Base in California [44]. It has to be noted that the Space Shuttle was manned and hence the guidance was not autonomous, but manual.

In recent years, the science of EDL sequence on Earth has seen further improvements, with the use of reusable launchers to perform reentry on Earth of some stages in order to

reuse them.
An emblematic example is the Falcon 9 rocket from SpaceX which performs the reentry of the first stage, after separating the second one. Falcon 9 achieves an autonomous landing dispersion of 10 meters on Autonomous Spaceport Drone Ship (ASDS) and 30 meters for landing at Cape Canaveral [45]. To obtain these results, different technologies are used, such as a Ground Positioning System (GPS) to precisely retrieve its position, aerodynamic surfaces as grid fins, which stabilize and correct the trajectory during the landing. It is worth mentioning that Falcon 9 uses convex optimization algorithms integrated in the onboard software to generate on-line customized flight codes during the landing sequence [45].
One of the most ambitious programs of SpaceX is the Starship program, which, inheriting some technologies from the Falcon 9 rocket, aims to perform precise landings on the Moon and Mars [46].

The main differences from the Mars scenario are that on Mars no GPS system can be used, as well as the atmosphere density, which, as already stated, is almost 100 times denser on Earth than on Mars. Therefore, an entry vehicle on Mars cannot count only on aerodynamic drag to decelerate with parachutes, but has to also use retropropulsion systems as described previously, or other technologies such as the aforementioned airbags or the sky crane.
Moreover, since no GPS system is available, the procedure to land on Mars has to be completely autonomous.
Another remarkable difference is the choice of the landing site, while on Earth it is possible to land in prepared landing zones, such as runways for the Shuttle or platforms for Falcon 9, on Mars the landing site is unknown and irregular.
All these differences highlight the challenges of landing on Mars and explain why achieving high landing precision for a mission towards the red planet is harder than on Earth.

## 1.8. GNC subsystem framework

The performance of an autonomous landing and its accuracy are mainly related to the Guidance, Navigation, and Control subsystem.
It is the software part that allows the lander to autonomously respond to unexpected changes in the trajectory and eventual perturbations.

In particular, the three parts of the subsystem operate as follows.

- **Guidance**: Determines online the trajectory to be followed (if not pre-computed offline) from the entry into Mars atmosphere until the touchdown, in order to per-

fectly reach the target, satisfying the requirements, and respecting the constraints. The trajectory computed is sent each time instant to the Control.

- **Navigation**: Continuously determines through sensors the spacecraft position, velocity, attitude, and angular rates that are sent both to Guidance in order to update the trajectory to be followed and to the Control part.
- **Control**: Taking as input both the trajectory to be followed and the estimations of the state, it computes at each time instant the control actions to be applied in order to follow the desired trajectory until the touchdown, and it sends the computed actions to the thrusters.

The focus of this work, as already previously stated, is on the **Control** block, which, in the scenario analyzed, takes the information on horizontal position from the LVS, while the estimation on altitude and velocity from the Radar Doppler Altimeter (RDA). This controller will be enhanced by testing LQR and MPC control techniques, while **Guidance** and **Navigation** blocks are used as provided by TAS-I.

Two configurations of the controller will be analyzed, the first one with RDA and so with both measurements of altitude and velocity, and the second one only with the altimeter, so the velocities are retrieved as states of the kinematic filter fed by measured accelerations (from IMU), measured altitude (from Altimeter), and measured horizontal distances from the target (from LVS). This configuration will be referred to as ALO (ALtimeter Only). This second configuration of course is more challenging, having fewer measurements, but still is very interesting to study since having one instrument less on board would lower the cost and the total weight to be carried in flight.

For what concerns the guidance section, as already told, the first part of the trajectory is the so called divert maneuver, from the entry in Mars atmosphere until about 300 meters of altitude, where the lander is supposed to be vertically positioned above the target.

The divert maneuver part of the guidance is precomputed offline and given to the software as input, while the second part of the descent has an online guidance system that provides to the controller the trajectory to be followed in the last vertical descent to the target.

# 2 | Simulator overview and performance requirements at touchdown

The simulator provided by TAS-I is presented in Fig. 2.1. A fundamental distinction in three main macro-blocks can be made.

1. The module devoted to the simulation of the real-world. It includes lander dynamics and kinematics, actuators, sensors, and environmental disturbances.
2. The module representative of the on-board software (SW) and avionic behaviors inside the on-board computer.
3. The performance section included to compare actual and estimated states in order to monitor the simulation

The Real-world module is called *Dynamics* in the Simulink model shown in the next page and it includes as said the actuators, that are eight MR80 thrusters displaced symmetrically on the bottom of the spacecraft, and sensors that are RDA, IMU and LVS.
The module representative of the on-board SW includes the following sections:

1. *Guidance*, that with a first phase of pre-computed trajectory for the divert maneuver and a second phase of online reference building, provides to the controller the trajectory to be followed.
2. *Navigation*, including specific post-processing blocks to elaborate the raw sensors' data in order to obtain valuable engineering measurements, and filters, for data fusion and state estimation.
3. *Control*, encompassing regulation algorithms to compute the forces and torques in the thruster space, in order to track the prescribed attitude and trajectory.

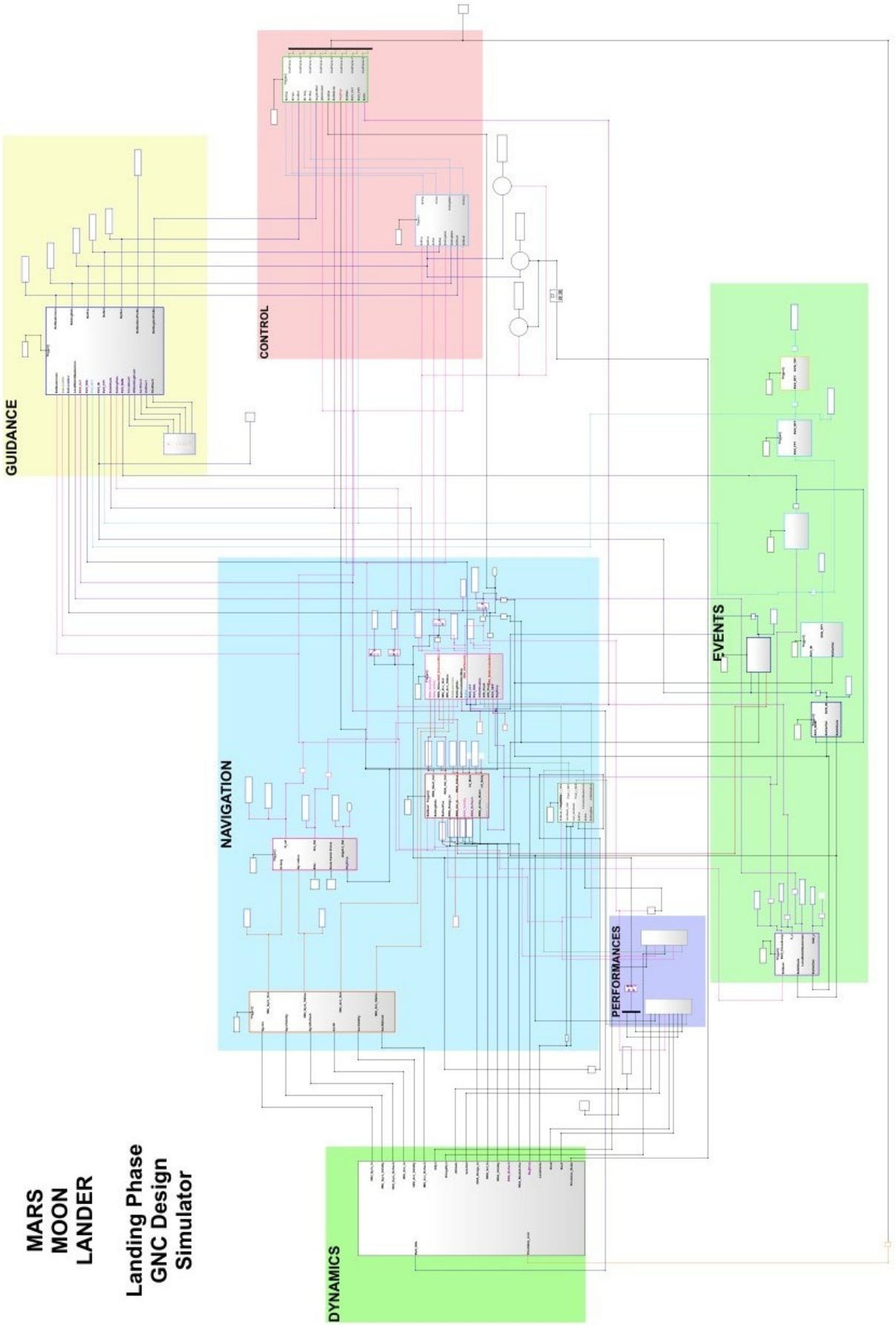


Figure 2.1: Simulink model of the GNC architecture. Credits: TAS-I.

4. The last section depicted in Fig. 2.1 is the one of the *Events* block. Conceptually part of the *Guidance* functionality, they are devoted to enable different landing modes, consistently with the monitored kinematic conditions. For example, the Radar Outside the Loop event disables the RDA operations at a certain altitude, when it is no longer able to give accurate estimations due to the too high proximity to ground.

The aim of the thesis is, as previously stated, to improve the performance given by the simulator by using more advanced control techniques. The baseline controller will be the benchmark for comparison with future results. Initially, the baseline controller was built as a Proportional-Integrative-Derivative (PID) Controller structured as follows:

- **Translation Controller** devoted to compute the forces to be applied in order to follow the trajectory both vertically and horizontally, following the planned profiles.
- **Attitude Controller** that aims to compute the torques needed to correct rotations and keep the lander in the correct orientation following the reference.
- **Dispatch block** to report the control actions (determined in the lander body frame) in the thrusters space. It also monitors the control actions amplitude and feasibility, and it applies protection against saturation, if necessary.
- **Gain manager** to compute, adjust, and tune the gains to be used.

The controller, each time instant, receives the errors in position, velocity, attitude and angular rate given by the difference between estimated state from Navigation and reference state from Guidance. These errors combined with the selected gains provide the control actions to be applied.

Before analyzing the baseline performances provided by the PID controller, it is important to state what the landing requirements and the initial conditions are.

## 2.1. Landing requirements at touchdown

The following requirements are from ExoMars experience by TAS-I, the aim is to perform a successful soft landing with a stable attitude at touchdown.

- Vertical velocity $< 2.5$ m/s
- Horizontal velocity $< 2$ m/s
- Off vertical angular displacement $< 7°$

- Off vertical rate of displacement $< 3.2°/s$
- With RDA configuration, horizontal distance from the target $< 60$ m
- With ALO configuration, horizontal distance from the target $< 100$ m
- Nice to have a propellant consumption lower than 300 kg

The distance requirements are computed in such a way that the lander can be reached by the rover in a reasonable time.

## 2.2. Initial conditions

Performance analysis will be performed in the next sections using a Monte Carlo simulation with a population of 100. Each case of the Monte Carlo simulation has different initial conditions and hence different trajectories to ground, always with the aim of reaching the axes origin with the aforementioned landing requirements.
Hence, there is a different mission profile for each case of the Monte Carlo, and each one is mainly composed of the divert maneuver (pre-computed) and the last part of the trajectory.

The most relevant initial conditions, directly provided by TAS-I and representative of the expected uncertainties and dispersions, are the following:

1. The initial horizontal position is uniformly distributed within a circle of radius 3000 m centered on the target, with random direction. This choice is made in order to encounter the abilities of the guided entry phase that should end with the parachute deployment with a precision of 3 km, in order to allow error recovery at landing during the divert maneuver.
2. The altitude is set to be within a minimum of around 2400 meters and a maximum of around 2700 meters.
3. The horizontal velocity reaches a maximum around 7 $m/s$ as initial condition.
4. The initial vertical velocity is between $80\,m/s$ and $100\,m/s$.

Regarding the rotational aspect, the lander is supposed to have almost stable initial conditions, with a maximum off vertical angular displacement (i.e. off vertical angle) around 5° between its body vertical axis and the local vertical direction, and, similarly, maximum transversal angular rate (i.e. off vertical rate of displacement) around $3.8°/s$.
The same set of 100 Monte Carlo initial conditions is used throughout this work; the population is the one related to Tab. 2.1 and kept fixed for all simulations.

The detailed initial conditions can be appreciated in Tab. 2.1.
Note that this type of table is going to be the reference table to present results and compare them among different types of controllers. The percentiles are useful for understanding, in first approximation, the distribution of values.
In the table also the data relative to angle at spin axis and angle rate at spin axis are reported for the sake of knowledge, even if there are no requirements on these parameters.

| | **Min** | **0.27%** | **1%** | **50%** | **99%** | **99.73%** | **Max** |
|---|---|---|---|---|---|---|---|
| **Altitude** [m] | 2412.700 | 2412.700 | 2420.800 | 2549.700 | 2674.800 | 2687.300 | 2687.300 |
| **Hor. Distance** [m] | 30.497 | 30.497 | 54.411 | 1245.600 | 2906.700 | 2959.900 | 2959.900 |
| **\|Vert Velocity\|** [m/s] | 80.070 | 80.070 | 80.270 | 89.920 | 99.430 | 99.724 | 99.724 |
| **Hor. Velocity** [m/s] | 0.444 | 0.444 | 0.563 | 4.189 | 6.802 | 7.040 | 7.040 |
| **Off Vert Angle** [°] | 0.005 | 0.005 | 0.008 | 1.161 | 4.575 | 4.980 | 4.980 |
| **Ang. @Spin Ax.** [°] | -0.979 | -0.979 | -0.973 | -0.081 | 0.938 | 0.961 | 0.961 |
| **Trans. Ang. Rate** [°/s] | 0.010 | 0.010 | 0.028 | 0.621 | 3.294 | 3.885 | 3.885 |
| **Rate @Spin Ax.** [°/s] | -0.626 | -0.626 | -0.576 | -0.017 | 0.674 | 0.751 | 0.751 |

Table 2.1: Initial conditions of the population for the 100 cases Monte Carlo simulation.

To perceive the dispersion of the initial conditions, the histograms of distribution for the parameters presented in the table above are reported.
In Fig. 2.2 and Fig. 2.3 are reported the histograms for the translational position, hence the altitude and the horizontal distance from the target.

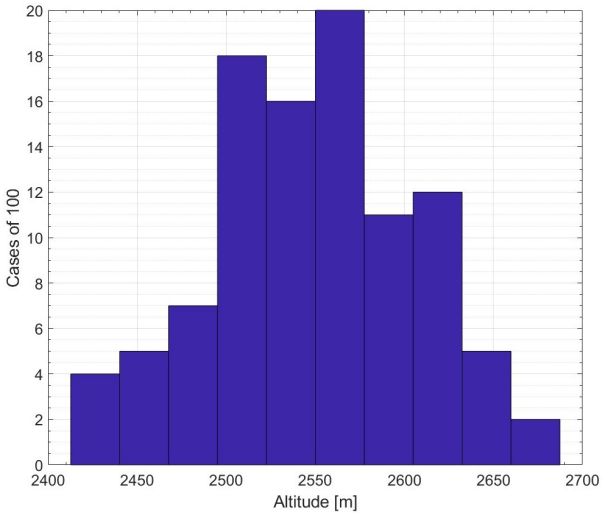


Figure 2.2: Altitude dispersion histogram - Initial conditions.

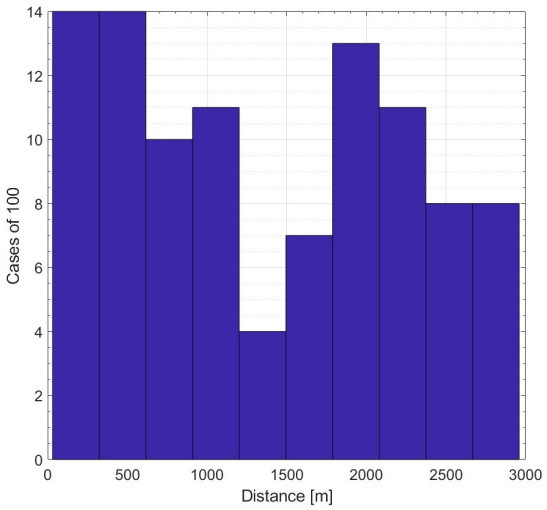


Figure 2.3: Horizontal distance from target dispersion histogram - Initial conditions.

In Fig. 2.4 and Fig. 2.5 are reported the histograms for the translational velocities, therefore, vertical and horizontal velocities distribution.

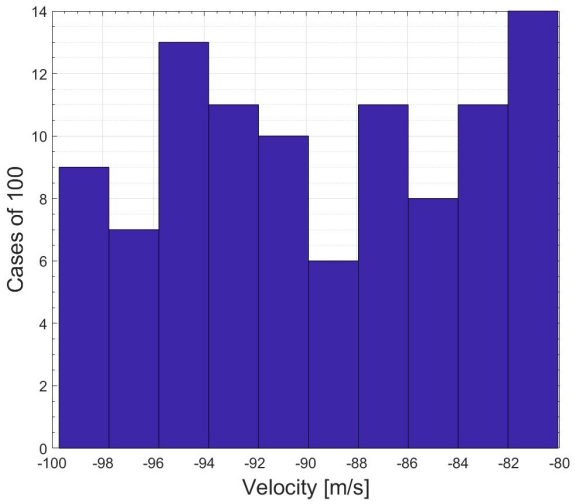


Figure 2.4: Vertical velocity dispersion histogram - Initial conditions.

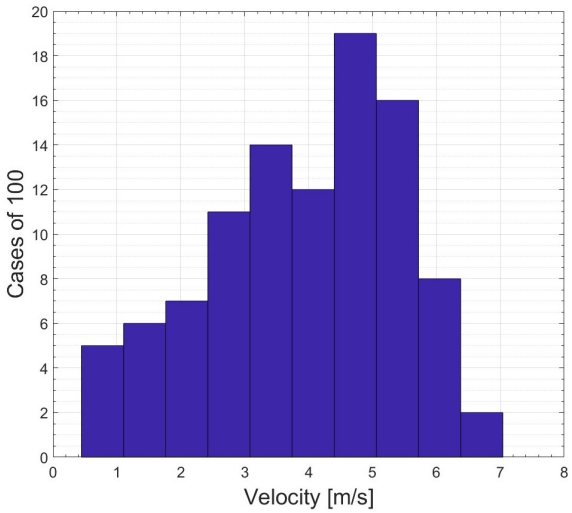


Figure 2.5: Horizontal velocity dispersion histogram - Initial conditions.

In Fig. 2.6 the dispersion of the initial conditions of the off-vertical angle is shown, while in Fig. 2.7 the one of the angle at the spin axis.

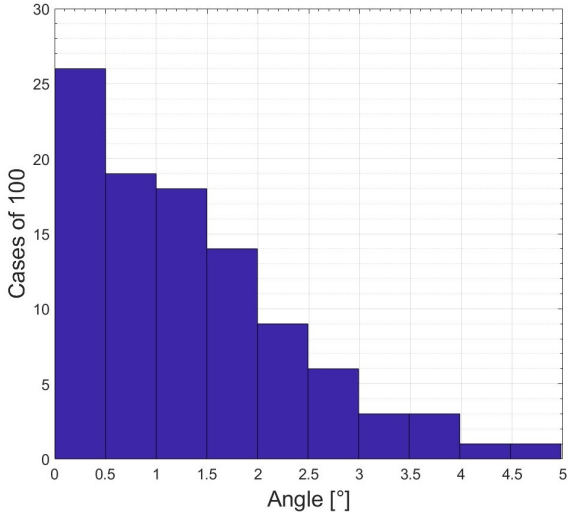


Figure 2.6: Off-vertical angle dispersion histogram - Initial conditions.

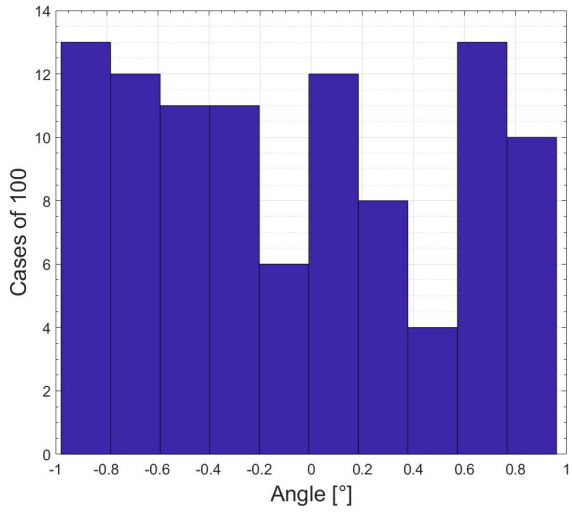


Figure 2.7: Angle at spin axis dispersion histogram - Initial conditions.

Eventually, in Fig. 2.8 and Fig. 2.9 is reported the distribution of initial values for the transversal angular rate and the rate at spin axis.

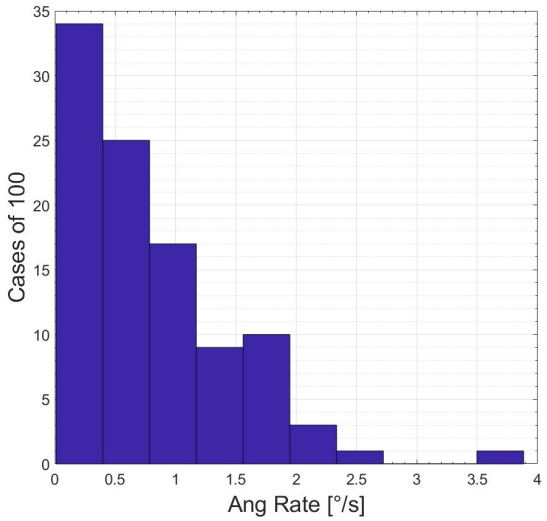


Figure 2.8: Transversal angular rate dispersion histogram - Initial conditions.

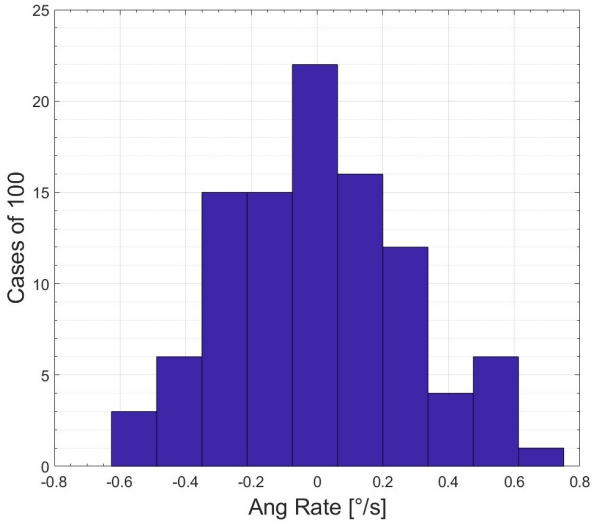


Figure 2.9: Rate at spin axis dispersion histogram - Initial conditions.

### 2.2.1. Navigation overview

Even if Navigation is not the focus of the work, it is important and interesting to understand how the process of retrieving measurements on the states works in this simulator. Since this simulator is built with the scope of scalable increase of complexity with respect to the ExoMars navigation, the approach followed for this Pinpoint Landing simulator has an incremented architecture compared to the ExoMars one.
In short, the navigation block first checks the measurements versus the predicted values, and it determines single input data for the filters in case two measurements are available (i.e. with gyroscopes and accelerometers).
After the measurements are prepared, they are fed to the filter composed of a kinematic predictor and a disturbance predictor. The kinematic predictor is responsible for the measurement process and the correction. The disturbance predictor structure is flexible in managing the disturbances acting on the system and easier to be tuned with respect to the approaches with Kalman filters.
The measurement process comes from the onboard sensors (IMU, Radar Doppler Altimeter, and LVS), but is different for the two sensor configurations, RDA and ALO, since the latter has no measurement coming from the Doppler of the RDA, therefore, it has no direct measurement on the velocities.

# 3 | Initial performances: PID control

In this chapter, the PID results are going to be presented, with both the configurations with RDA and ALO, which will be used as benchmark for future results.

## 3.1. Results - RDA configuration

In Tab. 3.1 the results are presented in terms of percentiles of the simulation, the minimum value and the 0.27% value are the same, as well as the maximum value and the 99.73%, due to the fact that the population for the simulation was exactly 100.
The touchdown (TD) values reported are all the ones aforementioned in the requirements, plus the last line regarding the propellant mass consumption and the last column for the requirements.

| | **Min** | **0.27%** | **1%** | **50%** | **99%** | **99.73%** | **Max** | **Req.** |
|---|---|---|---|---|---|---|---|---|
| **Hor. Distance** [m] | 0.385 | 0.385 | 2.484 | 29.147 | 56.059 | 61.023 | 61.023 | 60 |
| **\|Vert Velocity\|** [m/s] | 0.577 | 0.577 | 0.613 | 1.494 | 2.432 | 2.458 | 2.458 | 2.5 |
| **Hor. Velocity** [m/s] | 0.019 | 0.019 | 0.030 | 0.427 | 1.059 | 1.090 | 1.090 | 2 |
| **Off Vert Angle** [°] | 0.076 | 0.076 | 0.080 | 0.617 | 1.643 | 1.733 | 1.733 | 7 |
| **Ang. @Spin Ax.** [°] | -0.942 | -0.942 | -0.825 | 0.027 | 0.678 | 0.707 | 0.707 | |
| **Trans. Ang. Rate** [°/s] | 0.042 | 0.042 | 0.074 | 1.349 | 3.489 | 3.653 | 3.653 | 3.2 |
| **Rate @Spin Ax.** [°/s] | -0.501 | -0.501 | -0.494 | 0.004 | 0.528 | 0.531 | 0.531 | |
| **Prop. Mass** [Kg] | 183.797 | 183.797 | 184.977 | 230.731 | 291.043 | 292.035 | 292.035 | |

Table 3.1: PID touchdown values, RDA configuration.

As can be seen by comparing the maximum value to the required one from each line of the table, the requirements are not always perfectly fulfilled. The horizontal distance requirement, which characterizes the pinpoint landing, is almost satisfied having a maximum of around 61 meters of distance from the target. The requirements on velocities are met, even if on vertical velocity there is no margin of error since the highest one is really close

to 2.5 m/s.
With regard to the attitude, all the values are acceptable except for the transversal angular rate, i.e. the off vertical rate of displacement, which exceeds the requirement. This last result actually is not a big issue, since having a stable and reliable attitude should be enough, but the problem related to the transversal angular rate, which is related also to the horizontal touchdown velocity, is explained considering the risk of the lander tipping over when it lands in a steep terrain, so there is room for improvement in this aspect as well. Eventually, the propellant consumption meets the nice to have requirement of a mass lower than 300 kg.

To get a better idea of how the PID is performing, the figures on the next page can be analyzed. They show with plots and histograms how the touchdown values of the main landing parameters of interest are distributed with respect to the Monte Carlo simulation of 100 cases. The histograms are built with 10 bins that evenly space the range from the minimum value to the maximum value of the parameter considered.
In Fig. 3.1 there is the plot of the 100 cases trajectories, all starting from a different initial position and a different altitude, they aim to reach the origin of the axes.
On the right, in Fig. 3.2, the histogram showing the distribution of the landing positions is presented. As it is observable from Tab. 3.1 as well, there is only one case that exceeds the requirement, but five cases over 50 meters of distance and few cases under 10 meters of precision.

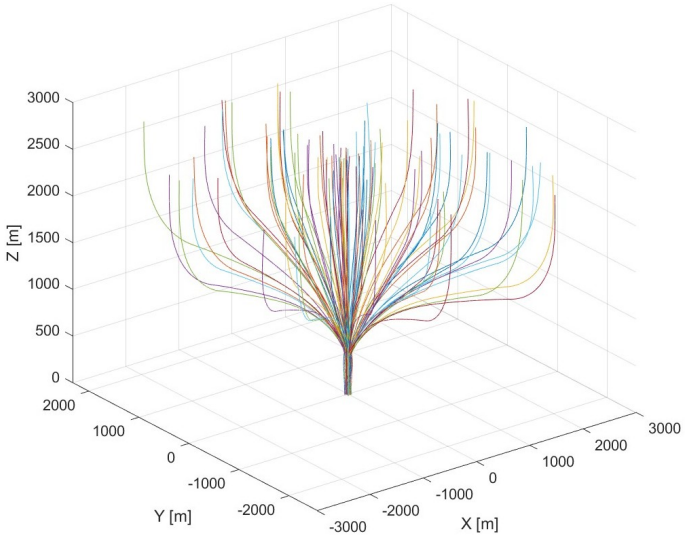


Figure 3.1: PID with RDA, Monte Carlo 3D trajectories.

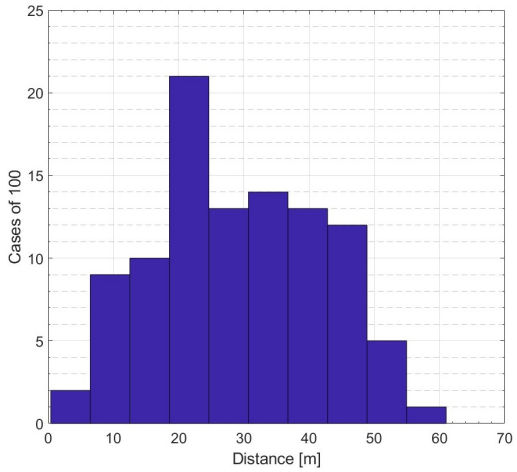


Figure 3.2: PID with RDA, horizontal distance at TD distribution histogram.

The horizontal velocity histogram is not of interest since, as in Tab. 3.1, the requirement

is largely satisfied. Instead, the vertical velocity plot and histograms are reported, since with a maximum vertical velocity of 2.458 m/s the requirement is only marginally satisfied and hence the distribution of the values is interesting.

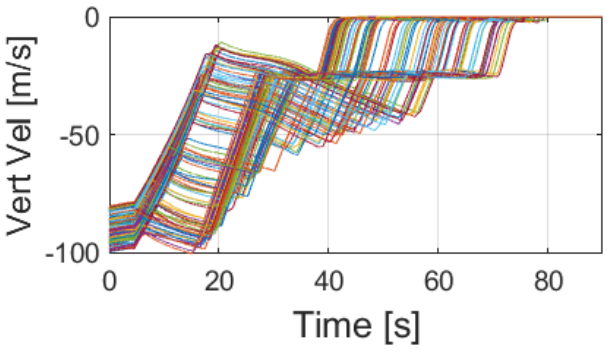


Figure 3.3: PID with RDA, vertical velocities plot.

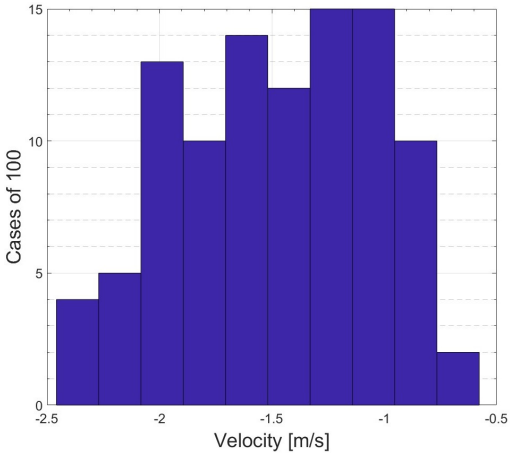


Figure 3.4: PID with RDA, vertical velocities at TD distribution histogram.

Fig. 3.3 shows the behavior of the 100 vertical velocities associated with the Monte Carlo analysis.

In Fig. 3.4 it can be seen that more than 10 cases have an absolute vertical landing velocity higher than 2 m/s, with a peak of 13 cases around 2 m/s. As for the horizontal distance from the target at touchdown, here there is room for improvement too.

## 3.2. Results - ALO configuration

It is also important to have a look at how the baseline simulator with PID performed in the configuration without radar doppler altimeter, but with only altimeter available. The results of the same Monte Carlo simulation of 100 cases are in the following table.

| | **Min** | **0.27%** | **1%** | **50%** | **99%** | **99.73%** | **Max** | **Req.** |
|---|---|---|---|---|---|---|---|---|
| **Hor. Distance** [m] | 5.218 | 5.218 | 5.723 | 33.349 | 93.353 | 101.168 | 101.168 | 100 |
| **\|Vert Velocity\|** [m/s] | 0.900 | 0.900 | 0.945 | 1.569 | 2.439 | 2.474 | 2.474 | 2.5 |
| **Hor. Velocity** [m/s] | 0.133 | 0.133 | 0.136 | 0.672 | 1.902 | 1.910 | 1.910 | 2 |
| **Off Vert Angle** [°] | 0.097 | 0.097 | 0.107 | 0.779 | 1.752 | 1.925 | 1.925 | 7 |
| **Ang. @Spin Ax.** [°] | -0.969 | -0.969 | -0.831 | 0.002 | 0.697 | 0.720 | 0.720 | |
| **Trans. Ang. Rate** [°/s] | 0.092 | 0.092 | 0.094 | 1.358 | 3.283 | 3.4454 | 3.445 | 3.2 |
| **Rate @Spin Ax.** [°/s] | -0.439 | -0.439 | -0.428 | 0.007 | 0.546 | 0.564 | 0.564 | |
| **Prop. Mass** [Kg] | 181.668 | 181.668 | 182.721 | 227.854 | 287.383 | 287.685 | 287.685 | |

Table 3.2: PID touchdown values, ALO configuration.

The performance is comparable to that of the previous configuration. Again, the PID struggles to satisfy the horizontal distance requirement, and the maximum vertical velocity at touchdown lies close to the requirement limit.

The horizontal velocity values have increased, as predictable as not having anymore a direct measurement of the velocities, and are close to the limit as well. Eventually, on the rotational aspect, the transversal angular rate again fails to accomplish the imposed requirement while the off vertical angle is still stable at touchdown.

In the figures Fig. 3.5 and Fig. 3.6, it is observable that again there are different cases with touchdown values of vertical velocities close to the limit, as well as 4 cases with a horizontal touchdown velocity near the limit requirement value of 2 m/s.

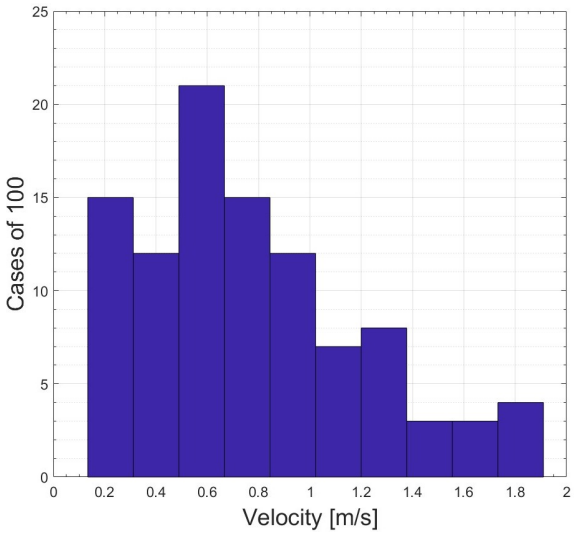


Figure 3.5: PID without RDA, horizontal velocities at TD distribution histogram.

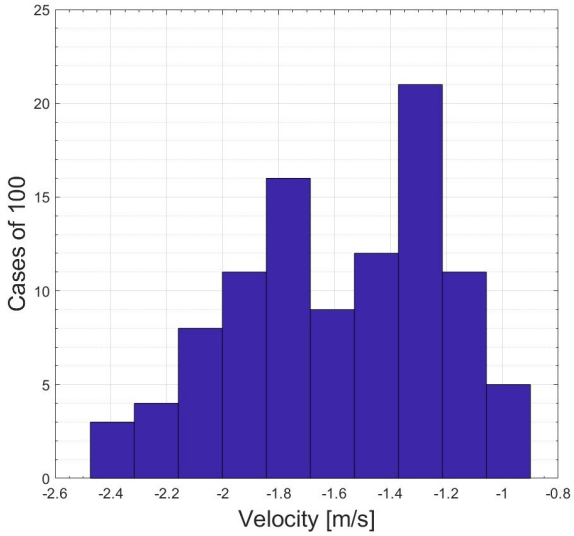


Figure 3.6: PID without RDA, vertical velocities at TD distribution histogram.

Figure 3.7 shows the landing distances from the target of the Monte Carlo analysis. As it was already clear from Tab. 3.2 there is actually only one case exceeding the 100 meters requirement, and from the histogram it is observable that only few cases land with a distance higher than 60 meters, with major concentration around 20-40 meters of distance.
This parameter is going to be one of the most relevant in the next analysis, since pinpoint landing is the focus of this work, and the landing distance from the target represents the most important performance index.

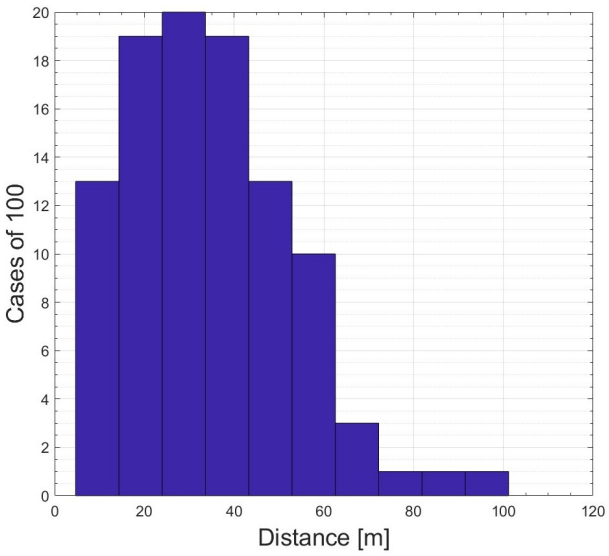


Figure 3.7: PID without RDA, horizontal distance from target at TD distribution histogram.

To conclude, this chapter has provided an overview of the initial performance, which will be useful in the following chapters, on LQR and MPC controllers, to assess how effectively the adopted controller performs and how significant the achieved improvements are with respect to the baseline starting results.

# 4 | LQR Controller

The first control strategy to be applied and tested, seeking an improvement of the PID performances, is a Linear-Quadratic-Regulator (LQR) Controller, and in this chapter the approach used, together with the mathematical formulation of the problem and the results obtained, are exposed. Starting from the LQR applied only on translation and moving to an integrated controller in both rotation and translation, the results of both the configurations, RDA and ALO, are analyzed for both the steps.

## 4.1. Mathematical formulation

The LQR controller is the most popular optimal control problem. The following formulation follows the approach presented in [47].
The general form for an optimal control problem of a nonlinear time-invariant system is as follows.

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{4.1}$$

Here $x \in R^n$ and $u \in R^m$ are the state variables and control inputs respectively, and $f(...)$ is the nonlinear function describing the system. The goal is to find a control that minimizes the following cost functional:

$$J(\mathbf{x}, t) = \int_t^{t_f} L(\mathbf{x}, \mathbf{u})\, d\tau \tag{4.2}$$

where $t$ is the current time, $t_f$ the final time, and $L(x, u)$ characterizes the cost objective. Applying the principle of optimality the problem can be solved with the Hamilton-Jacobi-Bellman equation, which is not always easy to solve.
Since the analyzed scenario is a tracking problem where state and input deviations from the reference trajectory are sufficiently small, the system can be linearized and the optimal control can be formulated as an LQR problem.

The error variables are defined as:

$$\delta\mathbf{x} = \mathbf{x} - \mathbf{x_r}, \quad \delta\mathbf{u} = \mathbf{u} - \mathbf{u_r} \tag{4.3}$$

Eq. 4.3 represents the difference between the state and its reference indicated by **r**. Later, it will be explained that the state considered is not the real state of the lander, since it is not possible to know it, but the estimated state provided by the on-board sensors.
The linearization is as follows.

$$\delta\dot{\mathbf{x}} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{(\mathbf{x}_r,\, \mathbf{u}_r)} \delta\mathbf{x} + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{(\mathbf{x}_r,\, \mathbf{u}_r)} \delta\mathbf{u} \tag{4.4}$$

The equivalent compact matrix form is the following.

$$\delta\dot{\mathbf{x}} = \mathbf{A}\,\delta\mathbf{x} + \mathbf{B}\,\delta\mathbf{u} \tag{4.5}$$

With

$$\mathbf{A} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{(\mathbf{x}_r,\, \mathbf{u}_r)}, \qquad \mathbf{B} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{(\mathbf{x}_r,\, \mathbf{u}_r)} \tag{4.6}$$

Equation 4.5 is the state space representation of the linear time-invariant system. The matrix **A** is the $n \times n$ dynamics matrix and the matrix **B** is the $n \times m$ control matrix. The cost function of the LQR problem, from Eq. 4.2, is quadratic and is expressed as follows.

$$J(\mathbf{x}, t) = \int_t^\infty (\delta\mathbf{x}^\mathbf{T}\mathbf{Q}\,\delta\mathbf{x} + \delta\mathbf{u}^\mathbf{T}\mathbf{R}\,\delta\mathbf{u})\,d\tau \tag{4.7}$$

Here $\mathbf{Q} = \mathbf{Q}^\mathbf{T} \geq 0$ is a symmetric and positive semi-definite matrix representing the state weights (cost) matrix, and $\mathbf{R} = \mathbf{R}^\mathbf{T} > 0$ is a symmetric and positive definite matrix representing the control weights (cost) matrix. The optimal control is obtained with the minimum of the cost function, which is given by:

$$\mathbf{u}^* = -\mathbf{R}^{-\mathbf{1}}\mathbf{B}^\mathbf{T}\mathbf{S}\mathbf{x} \tag{4.8}$$

The matrix **S** is unknown and is derived by the solution of the Riccati equation:

$$\mathbf{SA} + \mathbf{A^T S} + \mathbf{Q} - \mathbf{SBR^{-1}B^T S} = 0 \tag{4.9}$$

Once $\mathbf{S}$ is retrieved, the optimal control $\mathbf{u}^*$ can be computed, and Eq. 4.8 becomes:

$$\mathbf{u}^* = -\mathbf{Kx} \tag{4.10}$$

Here, matrix $\mathbf{K} = -\mathbf{R^{-1}B^T S}$ is the optimal state-feedback gain matrix, which concludes the LQR formulation.
First, the implementation of the LQR controller for translational dynamics is introduced. Subsequently, the integrated LQR approach addressing both translational and rotational dynamics is presented.

## 4.2. LQR on translation

Since the baseline PID controller had two different controllers for translation and rotation, the first step of the rationale of this work is to use the LQR controller only on translation, keeping the PID controller on rotation, so building a controller to compute the forces to be applied in order to keep the lander along the reference horizontal and vertical profiles.

### 4.2.1. Mathematical model

In this thesis the LQR problem is a discrete problem solved by using as state vector (in Eq. 4.3) the error vector between the estimated (measured) state given by the Navigation section and the reference state given by the Guidance section.
The command vector is the acceleration vector that is multiplied by the mass, which, since this quantity, now, is only used internally to compute the thrust command within the controller, it is assumed constant and set to its initial value $m = 1500\,kg$ in this specific operation. Then, the resulting forces are applied after being mapped to the thrusters space through the Dispatch block.

The implementation in Matlab/Simulink is done with a simulator which operates in the discrete-world with a sampling time for the controller of $\Delta t = 0.1\,s$. This control sampling time is motivated by the system dynamics and is constrained by the reaction velocity of the thruster control valve, the equipment devoted to change step-by-step the level of the thrust in order to track the GNC commands. In the following equations, the index $i + 1$ denotes the state at the next discrete-time step, replacing the derivative of the state, obtained by propagating the state at step $i$ over the sampling interval $\Delta t$.

Hence the state space representation, in Eq. 4.5, in the discrete-world becomes:

$$\mathbf{x}_{i+1} = \mathbf{A}\mathbf{x}_i + \mathbf{B}\mathbf{u}_i \tag{4.11}$$

With the following optimal control solution:

$$\mathbf{u}_i = \mathbf{K}\,\mathbf{x}_i \tag{4.12}$$

Note that usually the traditional form of the problem uses the difference between the estimated state and the reference state; here instead it is used the opposite, which is the reason for why instead of applying the gains as in Eq. 4.10 with a minus sign, they are applied without it.
Hence, the state vector is:

$$\mathbf{e}_{\mathbf{p}_i} = \mathbf{p}_{\mathbf{r}i} - \hat{\mathbf{p}}_i, \quad \mathbf{e}_{\mathbf{v}i} = \mathbf{v}_{\mathbf{r}i} - \hat{\mathbf{v}}_i, \quad \mathbf{x}_i = \begin{bmatrix} e_{p_x} \\ e_{v_x} \\ e_{p_y} \\ e_{v_y} \\ e_{p_z} \\ e_{v_z} \end{bmatrix}_i \tag{4.13}$$

with **r** standing for the reference profile and the hat for the estimated profile.
And the control vector is:

$$\mathbf{u}_i = \mathbf{a}_{\mathbf{c}i} = \begin{bmatrix} a_{c_x} \\ a_{c_y} \\ a_{c_z} \end{bmatrix}_i \tag{4.14}$$

The extended 4.12 equation in this framework is therefore:

$$\begin{bmatrix} a_{c_x} \\ a_{c_y} \\ a_{c_z} \end{bmatrix}_i = \begin{bmatrix} k_{x,x} & k_{x,v_x} & k_{x,y} & k_{x,v_y} & k_{x,z} & k_{x,v_z} \\ k_{y,x} & k_{y,v_x} & k_{y,y} & k_{y,v_y} & k_{y,z} & k_{y,v_z} \\ k_{z,x} & k_{z,v_x} & k_{z,y} & k_{z,v_y} & k_{z,z} & k_{z,v_z} \end{bmatrix} \begin{bmatrix} e_{p_x} \\ e_{v_x} \\ e_{p_y} \\ e_{v_y} \\ e_{p_z} \\ e_{v_z} \end{bmatrix}_i \tag{4.15}$$

This brings to the following set of equations for the $i-th$ step, with $\mathbf{a_c}$ being the com-

manded acceleration:

$$\begin{cases} a_{c_x} = k_{x,x}\, e_{p_x} + k_{x,y}\, e_{p_y} + k_{x,z}\, e_{p_z} + k_{x,v_x}\, e_{v_x} + k_{x,v_y}\, e_{v_y} + k_{x,v_z}\, e_{v_z} \\ a_{c_y} = k_{y,x}\, e_{p_x} + k_{y,y}\, e_{p_y} + k_{y,z}\, e_{p_z} + k_{y,v_x}\, e_{v_x} + k_{y,v_y}\, e_{v_y} + k_{y,v_z}\, e_{v_z} \\ a_{c_z} = k_{z,x}\, e_{p_x} + k_{z,y}\, e_{p_y} + k_{z,z}\, e_{p_z} + k_{z,v_x}\, e_{v_x} + k_{z,v_y}\, e_{v_y} + k_{z,v_z}\, e_{v_z} \end{cases} \tag{4.16}$$

Now, to build the **A** matrix and the **B** matrix, the equations of the evolution of the reference and estimated states have to be analyzed to derive the evolution of the errors. The reference evolution of the state, where $\mathbf{a_r}$ is the reference acceleration, i.e. the reference command, is ruled by the following equations for $\mathbf{p_r}$ and $\mathbf{v_r}$:

$$\mathbf{p_{r_{i+1}}} = \mathbf{p_{r_i}} + \Delta t\, \mathbf{v_{r_i}} + \frac{\Delta t^2}{2} \mathbf{a_{r_i}} \tag{4.17}$$

$$\mathbf{v_{r_{i+1}}} = \mathbf{v_{r_i}} + \Delta t\, \mathbf{a_{r_i}} \tag{4.18}$$

The kinematic evolution of the real system, with position $\mathbf{p}$ and velocity $\mathbf{v}$, is driven by $\mathbf{a_t}$, which is the total acceleration acting on the system, as follows:

$$\mathbf{p_{i+1}} = \mathbf{p_i} + \Delta t\, \mathbf{v_i} + \frac{\Delta t^2}{2} \mathbf{a_{t_i}} \tag{4.19}$$

$$\mathbf{v_{i+1}} = \mathbf{v_i} + \Delta t\, \mathbf{a_{t_i}} \tag{4.20}$$

Since it is only possible to know the estimations on the state, and not the real state itself, from equations 4.19 and 4.20 it is possible to retrieve the kinematic evolution of the estimated state (position $\hat{\mathbf{p}}$ and velocity $\hat{\mathbf{v}}$), which is analogous to the aforementioned equations but with *hat* notation.

$$\hat{\mathbf{p}}_{\mathbf{i+1}} = \hat{\mathbf{p}}_{\mathbf{i}} + \Delta t\, \hat{\mathbf{v}}_{\mathbf{i}} + \frac{\Delta t^2}{2} \hat{\mathbf{a}}_{\mathbf{t_i}} \tag{4.21}$$

$$\hat{\mathbf{v}}_{\mathbf{i+1}} = \hat{\mathbf{v}}_{\mathbf{i}} + \Delta t\, \hat{\mathbf{a}}_{\mathbf{t_i}} \tag{4.22}$$

Note that, with reference to the total acceleration, the "hat" notation is also applied, but with a different meaning. In this case, the notation indicates that the mathematical

formulation, implemented in the SW, cannot account for the total amount of the forces. Some lower dynamics are difficult to be precisely known (for instance, the internal propellant sloshing), and others are deliberately neglected because they are recognized to be magnitude orders lower than the ones linked to the thruster forces. By replacing in Eq. 4.13, the following system of equations is obtained:

$$\begin{cases} \mathbf{e_{p_{i+1}}} = \mathbf{e_{p_i}} + \Delta t\, \mathbf{e_{v_i}} + \dfrac{\Delta t^2}{2} \mathbf{a_{c_i}} \\ \mathbf{e_{v_{i+1}}} = \mathbf{e_{v_i}} + \Delta t\, \mathbf{a_{c_i}} \end{cases} \tag{4.23}$$

with $\mathbf{a_c}$ being the control action as in Eq. 4.14, obtained as:

$$\mathbf{a_{c_i}} = \mathbf{a_{r_i}} - \hat{\mathbf{a}}_{\mathbf{t_i}} \tag{4.24}$$

The system of equations (Eq. 4.23) can be written in matrix form with the state space representation (Eq 4.11).
Initially, the problem is treated with a decoupled axes approach with the following state and control matrices, retrieved from 4.23, for each axis:

$$\mathbf{A} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \dfrac{\Delta t^2}{2} \\ \Delta t \end{bmatrix} \tag{4.25}$$

This strategy brings to the following $\mathbf{Q}$ and $\mathbf{R}$ matrices, again for each axis, with the first one chosen to be diagonal and so without the correlation terms between position and velocity:

$$\mathbf{Q} = \begin{bmatrix} Q_{p_j} & 0 \\ 0 & Q_{v_j} \end{bmatrix}, \quad j = x, y, z. \tag{4.26}$$

$$\mathbf{R} = R_j, \quad j = x, y, z. \tag{4.27}$$

In addition to the decoupled axes approach, at the beginning the weights of the $\mathbf{Q}$ matrix are kept constant in time over the simulation. This first approach is used to validate the problem with its equations, but it is not effective in terms of results.

To obtain a better result, a coupled axes approach is designed in order to have a more general problem to analyze.

The extended **A** and **B** matrices are reported hereafter.

$$\mathbf{A} = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \end{bmatrix} \tag{4.28}$$

$$\mathbf{B} = \begin{bmatrix} \frac{\Delta t^2}{2} & 0 & 0 \\ \Delta t & 0 & 0 \\ 0 & \frac{\Delta t^2}{2} & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & \frac{\Delta t^2}{2} \\ 0 & 0 & \Delta t \end{bmatrix} \tag{4.29}$$

In this manner, the **Q** matrix becomes a $6x6$ matrix, which also includes the correlation terms between the weights, with $2x2$ submatrices from Eq. 4.26.

$$\mathbf{Q} = \begin{bmatrix} \begin{bmatrix} \mathbf{Q_{xx}} \end{bmatrix} & \begin{bmatrix} \mathbf{Q_{xy}} \end{bmatrix} & \begin{bmatrix} \mathbf{Q_{xz}} \end{bmatrix} \\ \begin{bmatrix} \mathbf{Q_{yx}} \end{bmatrix} & \begin{bmatrix} \mathbf{Q_{yy}} \end{bmatrix} & \begin{bmatrix} \mathbf{Q_{yz}} \end{bmatrix} \\ \begin{bmatrix} \mathbf{Q_{zx}} \end{bmatrix} & \begin{bmatrix} \mathbf{Q_{zy}} \end{bmatrix} & \begin{bmatrix} \mathbf{Q_{zz}} \end{bmatrix} \end{bmatrix} \tag{4.30}$$

The matrix **R** is instead kept diagonal, with no correlation terms.

$$\mathbf{R} = \begin{bmatrix} R_x & 0 & 0 \\ 0 & R_y & 0 \\ 0 & 0 & R_z \end{bmatrix} \tag{4.31}$$

### 4.2.2. Weights selection - RDA configuration

The weights in the two cost matrices are selected following the common guideline given by the Bryson's rule [48].
According to this rule, the diagonal terms of the weighting matrices are chosen as the inverse of the square of the maximum acceptable value of the corresponding state or control variable, as in Equation 4.32.

$$Q_{ii} = \frac{1}{x_{i,max}^2}, \quad R_{jj} = \frac{1}{u_{j,max}^2} \tag{4.32}$$

At this stage of the thesis the weights are chosen to be altitude-varying, and so no longer constant. Using a single run simulation of a worst-case scenario for the problem of this work, the maximum acceptable values, used to define the weights with the Bryson's rule, are derived by looking at the evolution of the errors as the altitude decreases during the simulation with the baseline simulator including the PID controller.
The errors analyzed (in position and velocity) are not the difference between estimated and reference state, but the difference between estimated and real state (taken from the dynamics block). This choice is made since the estimated-reference difference is directly linked to how precise the estimation of the real state is, hence this approach yields better performances.
The same procedure is used for the maximum acceptable accelerations to define the weights in the control cost matrix **R**, which are instead kept constant. The latter weights are chosen taking the maximum values of the acceleration on each of the three axes during the simulation with the baseline PID controller.

To have an altitude-varying behavior of the weights in the **Q** matrix, 1-D LookUp Tables (LUT) on Simulink are exploited. A LookUp Table takes as input a variable, in this case the estimated altitude, and gives as output the desired gain. The block-scheme of a LookUp Table on Simulink is presented in the following figure.

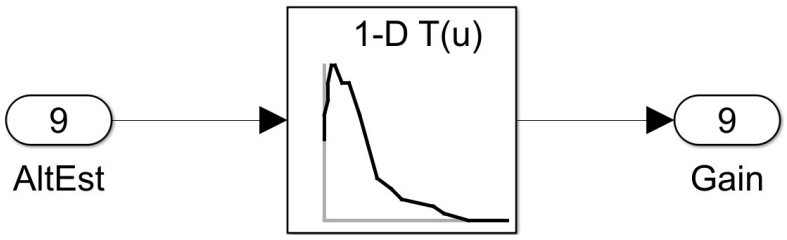


Figure 4.1: LUT block scheme.

The work principle of a LUT is the following: the altitude is partitioned into breakpoints, which are arbitrarily defined. At each breakpoint corresponds a chosen value of the gain, and a curve like the one in Fig. 4.2 is built.
With the decreasing of the altitude in time, the LUT provides the associated gain by linearly interpolating the curve.

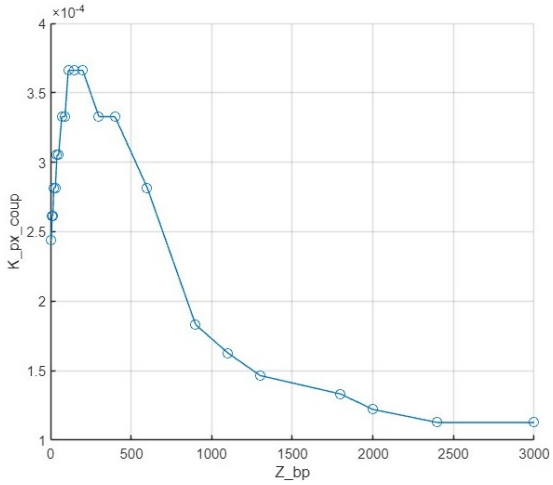


Figure 4.2: LUT curve plot example.

Now that the LUT operational principle is clear, the set of breakpoints and associated gain values can be presented.
The breakpoints have to be defined as a strictly monotonic increasing vector, which after several attempts is set as:

```
Z_bp = [0 10 20 50 100 150 200 300 400 600 900 1300 1800 2400 3000];
```

The unit of measure is meters. Note that the maximum value is 3000 meters to ensure it comprehends the initial altitude of all the cases in the Monte Carlo simulation. The number of breakpoints is denser close to ground, since the control is more critical in the last meters of descent, and having more breakpoints permits a more refined control.
After several iterations, the $\mathbf{R}$ matrix that provided the best performances is diagonal with the following weights, constants as already said, chosen according to the Bryson's rule with the aforementioned criteria:

$$\mathbf{R} = \begin{bmatrix} \frac{1}{0.03^2} & 0 & 0 \\ 0 & \frac{1}{0.03^2} & 0 \\ 0 & 0 & \frac{1}{1.5^2} \end{bmatrix} \tag{4.33}$$

The $\mathbf{Q}$ matrix structure obtained, again after refining the tuning, which yielded the best results, is reported hereafter.
The index $k$ goes from 1 to 13, that is, indeed, the length of the altitude breakpoints vector `Z_bp`.

$$\mathbf{Q}(k) = \begin{bmatrix} \frac{1}{\sigma_{p_x}(k)^2} & 0 & 0 & 0 & 0 & \frac{1}{\sigma_{p_x}(k)^2} \\ 0 & \frac{1}{\sigma_{v_x}(k)^2} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sigma_{p_y}(k)^2} & 0 & 0 & \frac{1}{\sigma_{p_y}(k)^2} \\ 0 & 0 & 0 & \frac{1}{\sigma_{v_y}(k)^2} & 0 & \frac{0.1}{\sigma_{v_y}(k)^2} \\ 0 & 0 & 0 & 0 & \frac{1}{\sigma_{p_z}(k)^2} & 0 \\ \frac{1}{\sigma_{p_x}(k)^2} & 0 & \frac{1}{\sigma_{p_y}(k)^2} & \frac{0.1}{\sigma_{v_y}(k)^2} & 0 & \frac{1}{\sigma_{v_z}(k)^2} \end{bmatrix}, \; k = 1...13. \quad (4.34)$$

Initially a diagonal $\mathbf{Q}$ matrix was used, then some correlation terms have been added to enhance the performance. In particular:

- $\mathbf{Q_{16}}$ (and for sym. $\mathbf{Q_{61}}$) represents the correlation between the position in $x$ and the velocity in $z$, imposed to be equal to the diagonal weight related to the position in $x$ itself.
- The same applies for $\mathbf{Q_{36}}$ and $\mathbf{Q_{63}}$ for the correlation between the position in $y$ and the velocity in $z$.
- $\mathbf{Q_{46}}$ (and for sym. $\mathbf{Q_{64}}$) instead represents the correlation between the velocity in $y$ and the velocity in $z$, chosen after tuning to be the 10% of the weight on $y$ velocity.

The weights on $\mathbf{Q_{26}}$ and $\mathbf{Q_{62}}$, which represent the correlation between velocity in $x$ and velocity in $z$, were initially imposed in the same way as $\mathbf{Q_{46}}$ and $\mathbf{Q_{64}}$. However, these terms were omitted since the performance change was not providing any benefit, while on the other hand a matrix with larger number of elements is more computationally demanding.
The $\sigma$ sets chosen for each variable, after a refined tuning process, are:

$\sigma_{p_x}$ = `[10 20 30 40 50 50 50 60 60 60 70 80 90 100 100];`
$\sigma_{p_y}$ = `[10 20 30 40 50 50 50 60 60 60 70 80 90 100 100];`
$\sigma_{p_z}$ = `[40 40 40 40 40 40 40 40 40 40 40 40 50 50 50];`
$\sigma_{v_x}$ = `[0.5 1 2 3 4 5 5 6 6 6 6 7 8 10 10];`
$\sigma_{v_y}$ = `[0.5 1 2 3 4 5 5 6 6 6 6 7 8 10 10];`
$\sigma_{v_z}$ = `[0.3 0.5 1.1 2.2 3 3 3 4 5 5 5 5 5 5 5];`

With everything defined, the algorithm to retrieve the gains (matrix **K**) can be exposed.

## Gains computation

With the chosen forms of **Q** and **R** matrices, the matrix of gains **K** from Equation 4.15 has only some terms that are non-zero, as shown in Eq. 4.35.

$$\mathbf{K} = \begin{bmatrix} k_{x,x} & k_{x,v_x} & 0 & 0 & 0 & k_{x,v_z} \\ 0 & 0 & k_{y,y} & k_{y,v_y} & 0 & k_{y,v_z} \\ k_{z,x} & k_{z,v_x} & k_{z,y} & k_{z,v_y} & k_{z,z} & k_{z,v_z} \end{bmatrix} \tag{4.35}$$

For each non-zero term of the matrix, a vector of 13 elements has to be defined (i.e. equal to the length of the breakpoints vector), which is going to be passed to the associated LUT. This vector will define the $y$-axis of the plot, as the one in Fig. 4.2, to build the linear interpolation curve through which the LUT provides the correct gain associated to the estimated altitude. Hence, there are a total of 12 LookUp Tables, one for each non-zero term of the gain matrix.
The vector of values for each $k$ of the gain matrix is obtained using the Matlab function `dlqr()` iteratively, as shown in Algorithm 4.1. This function is used to solve the LQR problem for discrete systems, it directly provides as output the **K** matrix.

Algorithm 4.1 Gains set computation

1: Empty gains vector definition, one for each non-zero term of the gain matrix **K**.
2: **for** $i$ from 1 to $length(Z_{bp})$ **do**
3: Define $\mathbf{Q}(i)$ with the $i-th$ set of $\sigma$
4: Compute $\mathbf{K}(i)$ matrix with `K = dlqr(A,B,Q,R)`
5: Allocate the non-zero terms of **K** to the previously created vectors.
6: **end for**
7: Gains vector computed, to be passed to the associated LUT together with the altitude breakpoint vector.

With the gains given by the LookUp Tables, the control action is computed in terms of accelerations from Equation 4.16, which, by deleting the terms that are zeros, becomes:

$$\begin{cases} a_{c_x} = k_{x,x}\, e_{p_x} + k_{x,v_x}\, e_{v_x} + k_{x,v_z}\, e_{v_z} \\ a_{c_y} = k_{y,y}\, e_{p_y} + k_{y,v_y}\, e_{v_y} + k_{y,v_z}\, e_{v_z} \\ a_{c_z} = k_{z,x}\, e_{p_x} + k_{z,v_x}\, e_{v_x} + k_{z,y}\, e_{p_y} + k_{z,v_y}\, e_{v_y} + k_{z,z}\, e_{p_z} + k_{z,v_z}\, e_{v_z} \end{cases} \tag{4.36}$$

These accelerations are then multiplied by the mass obtaining the force, which is then transformed into the space of the thrusters using the Dispatch block, and applied to the lander to perform its control to follow the planned trajectory.
With the control procedure in mind, it is now possible to present the results obtained for the RDA configuration.

### 4.2.3. Results - RDA configuration

To have an immediate perception of the performances, the same table used for the PID is used to present the results. Comparing the values in Tab. 4.1 and Tab. 3.1 for the PID, an enhancement of the performances in the translational touchdown values is observable. The best result is a gain of 10 meters in landing accuracy, which brings all the cases of the Monte Carlo simulation to satisfy the requirement. Another important result is given by the general reduction in the vertical velocity touchdown values, with a maximum of 1.807 $m/s$ that allows for a softer landing and a safer margin compared to the one given by the PID. Eventually, there is also a significant improvement in propellant consumption, with around 15-18 kg of consumed mass less for each percentile.

| | **Min** | **0.27%** | **1%** | **50%** | **99%** | **99.73%** | **Max** | **Req.** |
|---|---|---|---|---|---|---|---|---|
| **Hor. Distance** [m] | 3.662 | 3.662 | 3.879 | 26.404 | 50.537 | 50.658 | 50.658 | 60 |
| **\|Vert Velocity\|** [m/s] | 0.691 | 0.691 | 0.707 | 1.132 | 1.786 | 1.807 | 1.807 | 2.5 |
| **Hor. Velocity** [m/s] | 0.066 | 0.066 | 0.068 | 0.484 | 1.160 | 1.174 | 1.174 | 2 |
| **Off Vert Angle** [°] | 0.219 | 0.219 | 0.238 | 2.526 | 5.314 | 5.450 | 5.450 | 7 |
| **Ang. @Spin Ax.** [°] | -0.937 | -0.937 | -0.844 | 0.017 | 0.630 | 0.632 | 0.632 | |
| **Trans. Ang. Rate** [°/s] | 0.041 | 0.041 | 0.078 | 1.197 | 3.837 | 4.317 | 4.317 | 3.2 |
| **Rate @Spin Ax.** [°/s] | -0.570 | -0.570 | -0.533 | 0.007 | 0.639 | 0.663 | 0.663 | |
| **Prop. Mass** [kg] | 165.059 | 165.059 | 165.828 | 215.905 | 274.875 | 275.360 | 275.360 | |

Table 4.1: LQR on translation touchdown values, RDA configuration.

The performance in rotational touchdown values, as expected, is degraded. This is due to the presence of two different controllers in translation and rotation, with the first one being an optimal controller, which dominates the overall control action reducing the effectiveness of the second one. The consequences are higher values for the off vertical angles at TD and the transversal angular rate that still fails to respect the requirement, as with the PID controller, but with even higher values. This problem is solved later, in Sec 4.3, with the integration of an LQR controller also on rotation. The main plots of interest are hereafter reported, showing the distribution of the horizontal distance from the target at touchdown and the vertical velocity at touchdown through histograms.

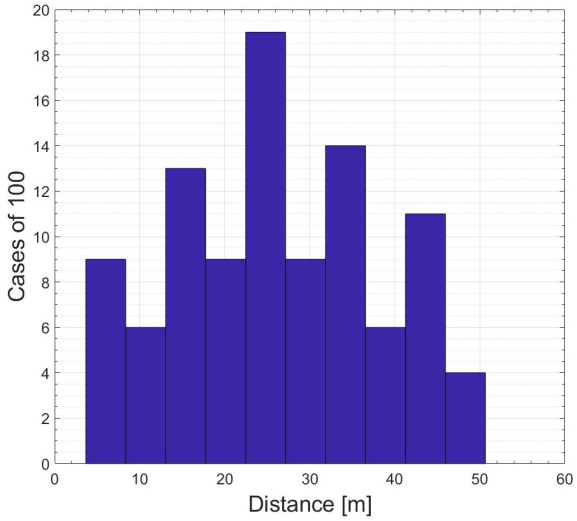


Figure 4.3: LQR on translation with RDA, horizontal distances at TD.

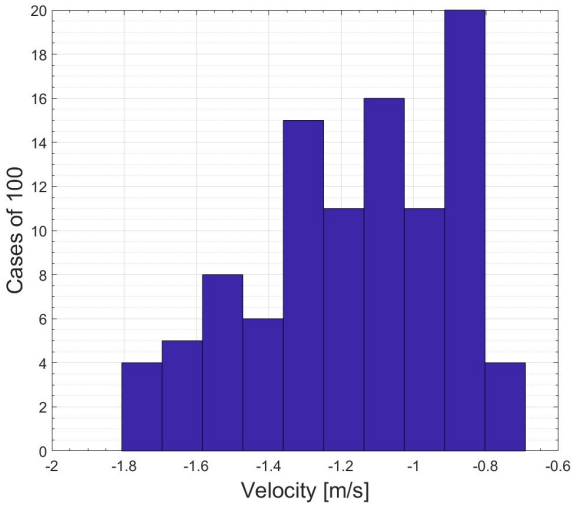


Figure 4.4: LQR on translation with RDA, vertical velocities at TD.

Comparing Fig. 4.4 with Fig. 3.4 it is clear that the general behavior has improved, having most of the cases concentrated at low velocity values with respect to a high number of cases over 2 $m/s$ (absolute value) in the PID simulation. The horizontal velocity is not exposed since the results are comparable to those with PID, and both are fully respecting the requirement.
To better visualize the results improvement, here is reported a summarizing table comparing the maximum values obtained, for the relevant parameters, of PID and the mixed configuration just discussed, with LQR on translation and PID on rotation (LQR+PID).

| **Max Parameter** | **PID** | **LQR+PID** |
|---|---|---|
| Horizontal Distance [m] | 61.023 | 50.658 |
| \|Vertical Velocity\| [m/s] | 2.458 | 1.807 |
| Off vertical Angle [°] | 1.733 | 5.450 |
| Transversal Ang. Rate [°/s] | 3.653 | 4.317 |
| Propellant Consumption [kg] | 292.035 | 275.36 |

Table 4.2: RDA configuration: PID - LQR on translation max. values comparison.

### 4.2.4. Weights selection - ALO configuration

Using only the altimeter and, therefore, no longer having direct measurements on velocities, the weights in terms of altitude breakpoints and $\sigma$ set for the matrix $\mathbf{Q}$ are differently chosen, as well as the matrix $\mathbf{Q}$ structure itself and the weights for the control cost matrix $\mathbf{R}$. Different settings have been tested, the tuning that yielded the best results is the one

that follows. For the **Q** matrix weights and the breakpoints needed for the LUTs:

```
Z_bp = [0 5 10 15 20 30 40 50 70 90 110 150 200 300 400 600 900 1100
        1300 1800 2000 2400 3000];
```

$\sigma_{p_x}$ = `[30 30 28 28 26 26 24 24 22 22 20 20 20 22 22 26 40 45 50 55 60 65 65];`

$\sigma_{p_y}$ = `[30 30 28 28 26 26 24 24 22 22 20 20 20 22 22 26 40 45 50 55 60 65 65];`

$\sigma_{p_z}$ = `[40 40 40 40 40 40 40 40 50 50 50 50 60 60 60 70 70 70 80 90 90 100 100];`

$\sigma_{v_x}$ = `[1.8 1.8 1.9 2.0 2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 2.9 2.7 2.5 2.2 2.0 1.9 1.9 1.9 1.9 1.9];`

$\sigma_{v_y}$ = `[1.8 1.8 1.9 2.0 2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 2.9 2.7 2.5 2.2 2.0 1.9 1.9 1.9 1.9 1.9];`

$\sigma_{v_z}$ = `[0.14 0.16 0.18 0.20 0.24 0.28 0.38 0.50 0.70 0.90 1.10 1.30 1.60 2.00 2.50 3.20 3.80 4.50 5.00 5.50 6.00 6.50 7.00];`

Note that the estimated altitude breakpoints vector has more items, especially near the ground, in order to allow finer tuning since, in this scenario, a higher control accuracy is needed.
Moreover, the weights related to the horizontal position are generally higher with respect to the weights in the RDA configuration, particularly for low altitudes. The weights on horizontal velocities present small variations and are again higher when approaching the soil. On the other hand, the weights for the vertical velocity are much lower, suggesting an acceptable error around $0.1 - 0.2 m/s^2$ in the last tenth of meters of descent.
All these choices are made since the Monte Carlo 100 cases simulation, for the LQR on translation with ALO, using the weights chosen for the RDA configuration, presented low performances and some failing simulations due to a too high vertical landing velocity. The solution has been to have a vertical velocity that weights more than the other parameters.
The **R** control cost matrix, which yielded the best results after tuning, is:

$$\mathbf{R} = \begin{bmatrix} \frac{1}{0.022^2} & 0 & 0 \\ 0 & \frac{1}{0.022^2} & 0 \\ 0 & 0 & \frac{1}{1.2^2} \end{bmatrix} \tag{4.37}$$

With slightly lower values of maximum acceptable accelerations with respect to the RDA configuration.

The **Q** matrix structure is changed as well, and after tuning is chosen to be diagonal, so the same as in Eq. 4.34 but only with diagonal terms. The gains computation procedure is the same shown for the RDA configuration with Algorithm 4.1, then LookUp Tables on Simulink are exploited and the control action is computed with the accelerations equations.
The only difference is that, having a diagonal **Q**, the matrix of gains **K** has a different structure with fewer terms that are non-zero:

$$\mathbf{K} = \begin{bmatrix} k_{x,x} & k_{x,v_x} & 0 & 0 & 0 & 0 \\ 0 & 0 & k_{y,y} & k_{y,v_y} & 0 & 0 \\ 0 & 0 & 0 & 0 & k_{z,z} & k_{z,v_z} \end{bmatrix} \tag{4.38}$$

With this structure of **K** there is no correlation between the axes and so each acceleration depends on position and velocity of its own axis, as the following system of equations shows.

$$\begin{cases} a_{c_x} = k_{x,x}\, x + k_{x,v_x}\, v_x \\ a_{c_y} = k_{y,y}\, y + k_{y,v_y}\, v_y \\ a_{c_z} = k_{z,z}\, z + k_{z,v_z}\, v_z \end{cases} \tag{4.39}$$

### 4.2.5. Results - ALO configuration

The results of the Monte Carlo simulation are presented with Tab. 4.3, the same as the other results to simplify the comparison.

The improvements in the translational performances are almost the same of the RDA configuration: by comparing Tab. 4.3 with Tab. 3.2 an improvement of around 10 meters of horizontal landing accuracy can be observed, together with lower and safer vertical velocities at touchdown, with a maximum value that is almost half of the one obtained with the PID controller.
The horizontal velocities are slightly better as well, whereas the maximum consumed propellant has increased by some kilograms even if, comparing the other percentiles, the general behavior has improved.

| | Min | 0.27% | 1% | 50% | 99% | 99.73% | Max | Req. |
|---|---|---|---|---|---|---|---|---|
| **Hor. Distance** [m] | 2.260 | 2.260 | 2.575 | 30.521 | 85.324 | 92.553 | 92.553 | 100 |
| **\|Vert Velocity\|** [m/s] | 0.765 | 0.765 | 0.768 | 0.888 | 1.243 | 1.262 | 1.262 | 2.5 |
| **Hor. Velocity** [m/s] | 0.122 | 0.122 | 0.123 | 0.643 | 1.738 | 1.765 | 1.765 | 2 |
| **Off Vert Angle** [°] | 0.061 | 0.061 | 0.075 | 0.823 | 1.753 | 1.878 | 1.878 | 7 |
| **Ang. @Spin Ax.** [°] | -1.007 | -1.007 | -0.833 | 0.054 | 0.678 | 0.730 | 0.730 | |
| **Trans. Ang. Rate** [°/s] | 0.033 | 0.033 | 0.042 | 1.195 | 5.400 | 4.613 | 4.613 | 3.2 |
| **Rate @Spin Ax.** [°/s] | -0.399 | -0.399 | -0.393 | 0.006 | 0.609 | 0.616 | 0.616 | |
| **Prop. Mass** [Kg] | 174.505 | 174.505 | 174.801 | 224.799 | 291.249 | 293.202 | 293.202 | |

Table 4.3: LQR on translation touchdown values, ALO configuration.

The transversal angular rate, as before, does not meet the requirement of a maximum of $3.2°/s$, but again the rotational behavior is not of interest in this first step and will be treated in the next section.
To complete the results comparison, the same histograms reported in Sect. 3.2 (PID Results - ALO configuration) regarding horizontal landing distance and vertical velocity at touchdown are presented hereafter.

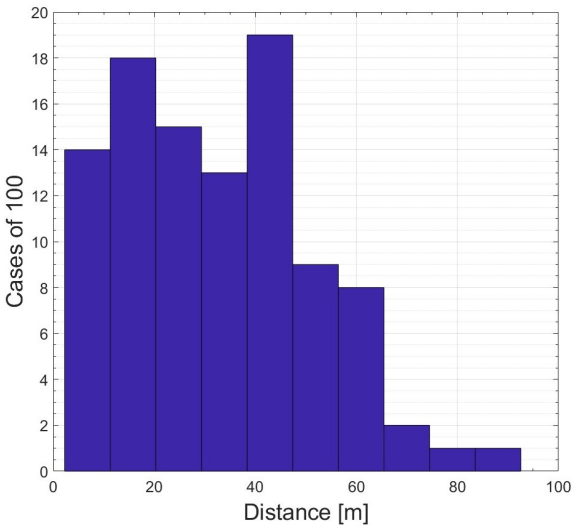


Figure 4.5: LQR on translation, ALO, horizontal distances at TD.

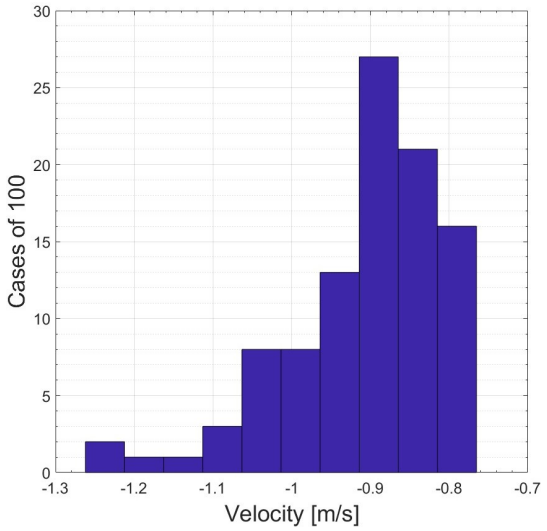


Figure 4.6: LQR on translation, ALO, vertical velocities at TD.

It is observable that the behavior in the horizontal distance is similar to the one with PID controller, except for the case where the requirement boundary was exceeded. Regarding the vertical velocities, instead, there is a significant improvement having only around 15 cases exceeding a vertical landing absolute velocity of 1 $m/s$.

Again, a comparison table as Tab. 4.2, is presented to compare the PID with the LQR

on translation, in the ALO configuration in this case.

| Max Parameter | PID | LQR+PID |
|---|---|---|
| Horizontal Distance [m] | 101.168 | 92.553 |
| \|Vertical Velocity\| [m/s] | 2.474 | 1.262 |
| Horizontal Velocity [m/s] | 1.910 | 1.765 |
| Off vertical Angle [°] | 1.925 | 1.878 |
| Transversal Ang. Rate [°/s] | 3.445 | 4.613 |
| Propellant Consumption [kg] | 287.685 | 293.202 |

Table 4.4: ALO configuration: PID - LQR on translation max. values comparison.

The next step is to integrate the LQR controller also in rotation, keeping the two controllers (translational and rotational) separated, in order to enhance the general performances of the simulator.

## 4.3. Integrated translational and rotational LQR

The LQR on rotation is designed in the same identical way as the translational one. Using state space representation, with the **A** and **B** matrices to be retrieved, the gains are derived again following Algorithm 4.1, after a tuning process of the weights for the cost matrices **Q** and **R**. Then LookUp Tables are exploited in Simulink by defining an estimated altitude breakpoints vector, as done for the translational control.
The only difference is that the angular acceleration is expressed in terms of control torque and inverse of the suitable diagonal term of the matrix tensor, so the control actions are the torques to be transposed in the thrusters space after passing through the dispatch block. This is the reason why, in the **R** matrix, the maximum acceptable values are related to the maximum acceptable torques (according again to the Bryson's rule) and in the **B** matrix the inertia matrix appears, as shown through the following mathematical formulation.

### 4.3.1. Mathematical model

The state is defined as follows:

$$\mathbf{x}_i = \begin{bmatrix} \mathbf{e_a} \\ \boldsymbol{e_\omega} \end{bmatrix}_i = \begin{bmatrix} e_{a_x} & e_{a_y} & e_{a_z} & e_{\omega_x} & e_{\omega_y} & e_{\omega_z} \end{bmatrix}_i^T \tag{4.40}$$

where $\mathbf{e_{a}}_i$ is the angular error and $\mathbf{e}_{\omega i}$ is the error in angular velocity. The error is again computed as the difference between reference and estimated state, so the gains are applied without the minus sign. The discretized evolution of the errors is analogous to the translational evolution in Eq. 4.23, but with rotational variables:

$$\begin{cases} \mathbf{e}_{a_{i+1}} = \mathbf{e}_{a_i} + \Delta t\, \mathbf{e}_{\omega \mathbf{i}} + \dfrac{\Delta t^2}{2}\, \boldsymbol{\alpha}_i \\ \mathbf{e}_{\omega \mathbf{i+1}} = \mathbf{e}_{\omega \mathbf{i}} + \Delta t\, \boldsymbol{\alpha}_i \end{cases} \tag{4.41}$$

The difference, as said, is that the angular accelerations are expressed as:

$$\boldsymbol{\alpha}_i = \mathbf{M^{-1}}\, \boldsymbol{\tau}_i \tag{4.42}$$

Where $\mathbf{M^{-1}}$ is the inverse of the inertia matrix, indicated with M (and not I) not to be confused with the matrix $\mathbf{I_3}$ which is the diagonal identity matrix $3x3$. The control action in this case is expressed as follows:

$$\mathbf{u}_i = \boldsymbol{\tau}_i = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix}_i \tag{4.43}$$

In first approximation, considering that $\Delta t$ is the sampling time associated to the controller in the discretized Simulink model which is $\Delta t = 0.1\, s$, the following assumption in Eq. 4.44 is made, since the results are not largely affected by the second order term.

$$\frac{\Delta t^2}{2}\, \mathbf{M}^{-1} \boldsymbol{\tau}_k \approx \mathbf{0} \tag{4.44}$$

The system of equations in Eq. 4.41 thus becomes:

$$\begin{cases} \mathbf{e}_{a_{i+1}} = \mathbf{e}_{a_i} + \Delta t\, \boldsymbol{e_{\omega_i}} \\ \boldsymbol{e_{\omega_{i+1}}} = \boldsymbol{e_{\omega_i}} + \Delta t\, \mathbf{M^{-1}}\, \tau_i \end{cases} \tag{4.45}$$

Hence, the cost matrices $\mathbf{A}$ and $\mathbf{B}$ are retrieved as:

$$\mathbf{A} = \begin{bmatrix} \mathbf{I}_3 & \Delta t\, \mathbf{I}_3 \\ \mathbf{0}_{3\times 3} & \mathbf{I}_3 \end{bmatrix} \tag{4.46}$$

$$\mathbf{B} = \begin{bmatrix} \mathbf{0}_{3\times3} \\ \Delta t\, \mathbf{M}^{-1} \end{bmatrix} \tag{4.47}$$

The inertia matrix $\mathbf{M}$ is diagonal with the following values, directly provided by TAS-I:

$$\mathbf{M} = \begin{bmatrix} 605.32 & 0 & 0 \\ 0 & 1020.49 & 0 \\ 0 & 0 & 1368.11 \end{bmatrix} \cdot 1.122 \quad [kg \cdot m^2] \tag{4.48}$$

The procedure for building the $\mathbf{Q}$ and $\mathbf{R}$ matrices is the same as for translational control. It is used the Bryson's rule looking at the behavior of the errors in the PID attitude baseline controller, in this way the set of $\sigma$ for the state cost matrix are defined as well as the weights for the control cost matrix.
The gains are then computed with the usual Algorithm 4.1 and LookUp Tables are exploited to make the gains altitude-varying. Eventually, the control actions (torques in this case) are computed as always with Eq. 4.12 and they are transposed in thrusters space and applied to control the attitude. The mathematical framework has been presented, and so the weights selection together with the results for both configurations is exposed.

### 4.3.2. Weights selection - RDA configuration

The $\mathbf{Q}$ matrix is chosen to be diagonal as well as the $\mathbf{R}$ matrix. Their tuning is here after reported, with the set of $\sigma$ chosen and the associated estimated altitude breakpoints vector:

$$\mathbf{R} = \begin{bmatrix} \frac{1}{1200^2} & 0 & 0 \\ 0 & \frac{1}{1200^2} & 0 \\ 0 & 0 & \frac{1}{120^2} \end{bmatrix} \tag{4.49}$$

```
Z_bp = [0 10 20 50 100 150 200 300 400 600 900 1300 1800 2400 3000];
σ_ax = [3   3   3   3   3.5 4   4.5 5   6   8   12  16  20  24  28];
σ_ay = [3   3   3   3   3.5 4   4.5 5   6   8   12  16  20  24  28];
σ_az = [15   15   15   15   15   15 15 15 15  15 15 15 15 15 15];
σ_w  = [25 25 25 25 25 25 25 25 25 25 25 25 25 25 25];
```

Note that the maximum allowable value for the angular error on $x$ and $y$ axes is smaller than on $z$ and for the angular velocities. This is due to the accuracy of the attitude

control needed, especially close to ground, for the off vertical angle. The control on $z$ axis and on angular velocities can be more relaxed. The diagonal $\mathbf{Q}$ matrix, with $i$ from 1 to the length of `Z_bp`, i.e. 15, is:

$$\mathbf{Q}(k) = \begin{bmatrix} \frac{1}{\sigma_{a_x}(k)^2} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{\sigma_{a_y}(k)^2} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sigma_{a_z}(k)^2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{\sigma_w(k)^2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{\sigma_w(k)^2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{\sigma_w(k)^2} \end{bmatrix}, \quad k = 1...15. \tag{4.50}$$

### 4.3.3. Results - RDA configuration

The results are presented here with the tuning for the LQR rotational control exposed in the previous section. The tuning of the LQR translational control instead is the same as used in Sect. 4.2.2.

| | **Min** | **0.27%** | **1%** | **50%** | **99%** | **99.73%** | **Max** | **Req.** |
|---|---|---|---|---|---|---|---|---|
| **Hor. Distance** [m] | 3.190 | 3.190 | 3.680 | 35.295 | 52.626 | 52.768 | 52.768 | 60 |
| **\|Vert Velocity\|** [m/s] | 0.672 | 0.672 | 0.694 | 1.184 | 1.778 | 1.833 | 1.833 | 2.5 |
| **Hor. Velocity** [m/s] | 0.076 | 0.076 | 0.078 | 0.401 | 1.185 | 1.201 | 1.201 | 2 |
| **Off Vert Angle** [°] | 0.320 | 0.320 | 0.322 | 3.407 | 5.242 | 5.277 | 5.277 | 7 |
| **Ang. @Spin Ax.** [°] | -0.947 | -0.947 | -0.867 | 0.053 | 0.677 | 0.719 | 0.719 | |
| **Trans. Ang. Rate** [°/s] | 0.043 | 0.043 | 0.045 | 0.377 | 0.821 | 0.828 | 0.828 | 3.2 |
| **Rate @Spin Ax.** [°/s] | -0.070 | -0.070 | -0.056 | 0.005 | 0.057 | 0.059 | 0.059 | |
| **Prop. Mass** [Kg] | 165.048 | 165.048 | 165.097 | 214.121 | 274.682 | 275.030 | 275.030 | |

Table 4.5: Integrated LQR touchdown values, RDA configuration.

The translation performances are more or less the same as those shown in Tab. 4.1 in Sect. 4.2.3. The main improvement is in the transversal angular rate values, which now are lower, significantly below the limit imposed by the requirement.
The other parameters, such as propellant consumption, horizontal distance, and velocities at touchdown, remained similar to the previous ones. The off vertical angle still exhibits relatively high values; however, they remain below the prescribed requirement and there-

fore do not represent an issue.
The result obtained in the transversal angular rate can be seen in the following plots.

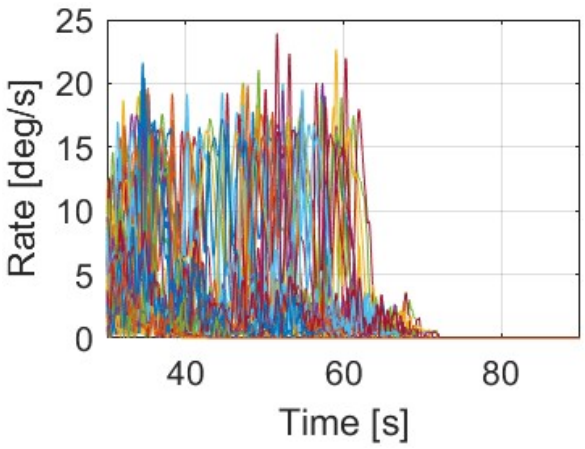


Figure 4.7: Integrated LQR with RDA, lateral rate in time MC simulation.

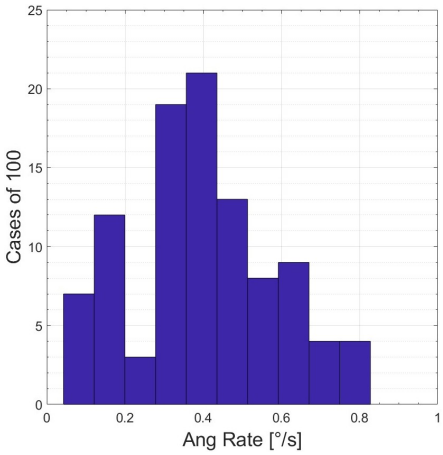


Figure 4.8: Integrated LQR with RDA, lateral rate TD values histogram.

Analyzing the plot of the transversal angular rate of the 100 cases from the Monte Carlo (MC) simulation in Fig. 4.7, it is possible to see a generally stable behavior, with peaks around $20 - 25\ °/s$, much lower than those obtained with the previous analysis with the LQR controller on translation and the PID controller on rotation, which were around $50 - 55\ °/s$.
A comparison table between PID, LQR on translation and integrated LQR is here presented to perceive the evolution of the performances among the different controllers tested so far.

| **Max Parameter** | **PID** | **LQR+PID** | **LQR** |
|---|---|---|---|
| Horizontal Distance [m] | 61.023 | 50.658 | 52.768 |
| \|Vertical Velocity\| [m/s] | 2.458 | 1.807 | 1.833 |
| Horizontal Velocity [m/s] | 1.090 | 1.174 | 1.201 |
| Off vertical Angle [°] | 1.733 | 5.450 | 5.277 |
| Transversal Ang. Rate [°/s] | 3.653 | 4.317 | 0.828 |
| Propellant Consumption [kg] | 292.035 | 275.36 | 275.030 |

Table 4.6: RDA configuration: PID - LQR on translation - Integrated LQR max. values comparison

With these results, the implementation of a full LQR controller, integrated both in rotation and translation, is validated for the RDA configuration.

### 4.3.4. Weights selection - ALO configuration

For the configuration with only altimeter the following set of $\sigma$ has been found to yield the best performance, with the **Q** matrix still diagonal.

```
Z_bp = [0 10 20 50 100 150 200 300 400 600 900 1300 1800 2400 3000];
```
$\sigma_{a_x}$ = [10 10 10 11 12 13 13 15 15 15 18 20 22 24 28];
$\sigma_{a_y}$ = [10 10 10 11 12 13 13 15 15 15 18 20 22 24 28];
$\sigma_{a_z}$ = [180 180 180 180 180 180 180 180 180 180 180 180 180 180 180];
$\sigma_w$ = [180 180 180 180 180 180 180 180 180 180 180 180 180 180 180];

The breakpoints vector is the same as before, and the controller is generally more relaxed with higher acceptable values for the angular errors in $x$ and $y$.
On the other hand, the control on the angular error on $z$ and the angular velocity control is almost null, considering that this angle is not interested in the divert maneuver and the linked kinematics are limited..
This choice has been made since it has been noted that the control on these variables was not effecting the rotational performances, hence relaxing the acceptable errors provided better results also in the translational aspect, as shown in the next paragraph.

The matrix **R** is tuned as follows:

$$\mathbf{R} = \begin{bmatrix} \frac{1}{1800^2} & 0 & 0 \\ 0 & \frac{1}{1800^2} & 0 \\ 0 & 0 & \frac{1}{1800^2} \end{bmatrix} \tag{4.51}$$

This present a more relaxed tuning as well, with same maximum acceptable torques for all the axes.

### 4.3.5. Results - ALO configuration

The results obtained with this configuration in the usual 100 cases Monte Carlo simulation present different improvements. Note that the tuning for the LQR translational controller is the same as presented in Sect. 4.2.4.
To fully perceive the enhancements in the performances obtained, the following table (Tab. 4.7), and in particular some of its values, have to be compared with the ones in Tab. 3.2 for the full PID controller and with Tab. 4.3 for the mixed LQR and PID controller, respectively, on translation and rotation.

| | Min | 0.27% | 1% | 50% | 99% | 99.73% | Max | Req. |
|---|---|---|---|---|---|---|---|---|
| **Hor. Distance** [m] | 0.475 | 0.475 | 1.719 | 21.328 | 58.875 | 65.423 | 65.423 | 100 |
| **\|Vert Velocity\|** [m/s] | 0.740 | 0.740 | 0.744 | 0.994 | 1.986 | 2.116 | 2.116 | 2.5 |
| **Hor. Velocity** [m/s] | 0.152 | 0.152 | 0.163 | 0.725 | 1.970 | 1.976 | 1.976 | 2 |
| **Off Vert Angle** [°] | 0.061 | 0.061 | 0.071 | 0.952 | 2.055 | 2.084 | 2.084 | 7 |
| **Ang. @Spin Ax.** [°] | -0.913 | -0.913 | -0.806 | 0.048 | 0.678 | 0.716 | 0.716 | |
| **Trans. Ang. Rate** [°/s] | 0.009 | 0.009 | 0.017 | 0.202 | 0.530 | 0.579 | 0.579 | 3.2 |
| **Rate @Spin Ax.** [°/s] | -0.080 | -0.080 | -0.075 | 0.000 | 0.227 | 0.233 | 0.233 | |
| **Prop. Mass** [Kg] | 158.279 | 158.279 | 159.076 | 205.325 | 283.683 | 289.531 | 289.531 | |

Table 4.7: Integrated LQR touchdown values, ALO configuration.

Immediately can be perceived the most important result: the maximum horizontal distance from the target at touchdown is around 65 $m$.
This is a huge improvement with respect to the previous simulations, since around 27 meters of landing precision are gained compared to the last result obtained, largely meeting the requirement.

Another important result relates to the propellant mass consumption. Comparing the percentiles in Tab. 4.7 with Tab. 4.3, a lower propellant consumption is appreciated for each percentile, in particular only some kilograms for the maximum value, but $15-20$ $kg$ less in the lower percentiles.

The latter difference gets even greater when comparing Tab. 4.7 with the table of PID results (Tab. 3.2).
Regarding the horizontal and vertical velocities, the values are acceptable, since they are still below the limit imposed by the requirement. However, it should be noted that these values have generally increased with respect to the one of 4.3, when using LQR only on translation. Especially the horizontal velocity reaches a maximum value that lies very close to the limit.
Comparing them with the PID controller instead, the values are similar, with slightly higher horizontal velocities at TD and slightly lower vertical velocities at TD. On the rotational aspect, the behavior meets the requirements as well, with values largely below the limit for the transversal angular rate and the off vertical angle, having a stable attitude controller.

To better present these results, the behavior of some of the aforementioned parameters is analyzed with plots and histograms.

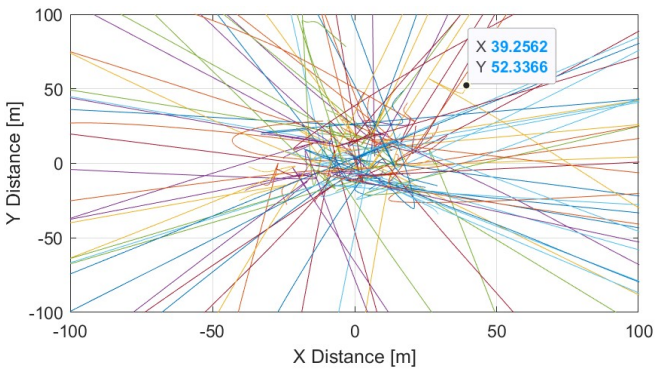


Figure 4.9: Integrated LQR, ALO, traj. projections on ground.

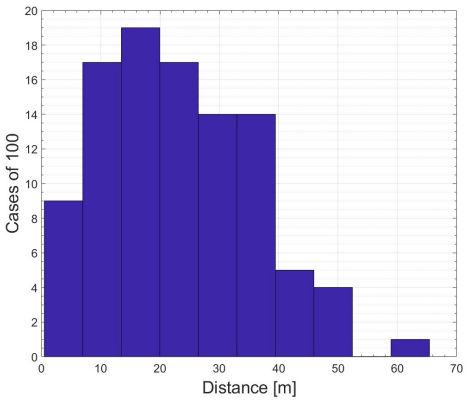


Figure 4.10: Integrated LQR, ALO, histogram of landing distances.

From Fig. 4.10 it is observable that only one case exceeds 60 meters of distance from the target at touchdown, and that 90 cases out of 100 have a landing distance lower than 40 meters. The trajectory projection on ground of the 100 cases simulation is shown in Fig. 4.9, where the one with the maximum horizontal distance at touchdown is highlighted with a marker on its landing point.

In Fig. 4.11 and Fig. 4.12 the histograms for the horizontal and vertical landing velocity are presented. From the vertical velocity histogram, it is shown what is reported by the percentiles in Tab. 4.7. More than half of the Monte Carlo simulation cases have absolute vertical landing velocities lower than 1 $m/s$, and only two cases have values close to 2 $m/s$. On the other hand, in Fig. 4.12 it is depicted that 8 cases have a horizontal touchdown velocity close to the limit.

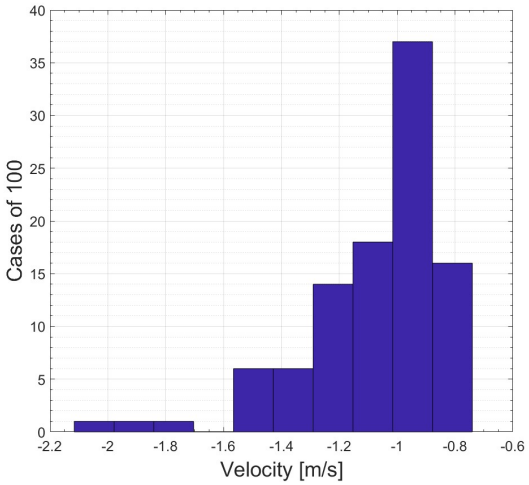


Figure 4.11: Integrated LQR, ALO, vertical distances at TD.

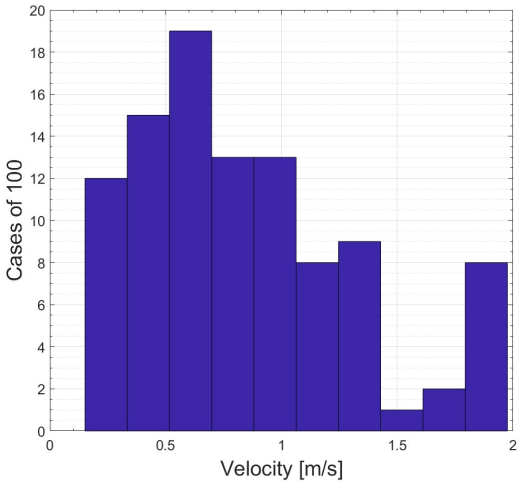


Figure 4.12: Integrated LQR, ALO, horizontal velocities at TD.

The last plot shown is related to the propellant mass consumption for the 100 cases. The trend of consumption is similar for all the simulations, also for the previous types of controller analyzed. It is interesting to see in this particular scenario that the consumption is mainly evenly distributed between $160\,kg$ and $270\,kg$, with a couple of peaks in $280 - 290\,kg$.
The high spread of propellant consumption of more than 100 kg is due to the length difference of the divert maneuver among the various simulations, having some simulations starting from an initial horizontal distance close to the target and others starting instead with kilometers of distance to be recovered through the divert maneuver.

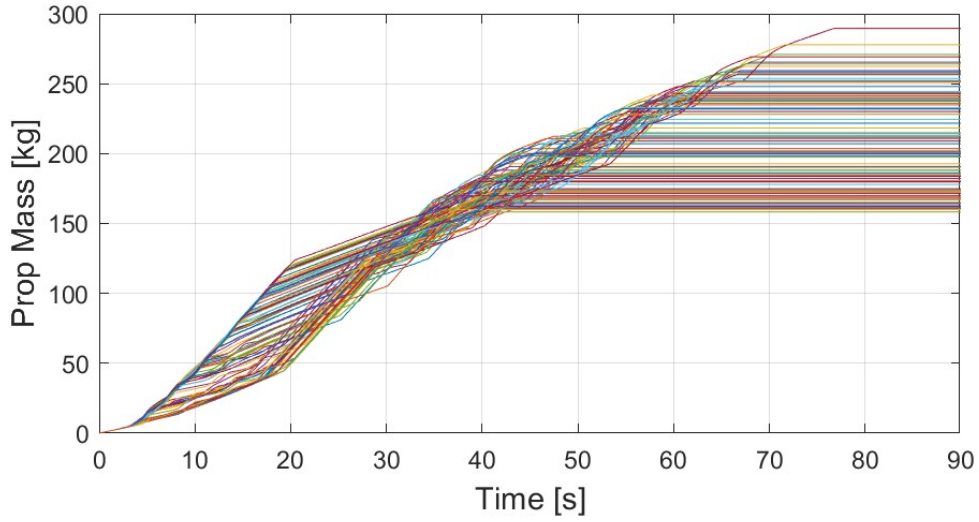


Figure 4.13: Propellant consumption trend for 100 cases simulation with integrated LQR.

Finally, the comparison table among PID, LQR on translation and integrated LQR is presented to perceive the difference between the main parameters' maximum values also for the ALO configuration.

| **Max Parameter** | **PID** | **LQR+PID** | **LQR** |
|---|---|---|---|
| Horizontal Distance [m] | 101.168 | 92.553 | 65.423 |
| \|Vertical Velocity\| [m/s] | 2.474 | 1.262 | 2.116 |
| Horizontal Velocity [m/s] | 1.910 | 1.765 | 1.976 |
| Off vertical Angle [°] | 1.925 | 1.878 | 2.084 |
| Transversal Ang. Rate [°/s] | 3.445 | 4.613 | 0.579 |
| Propellant Consumption [kg] | 287.685 | 293.202 | 289.531 |

Table 4.8: ALO configuration: PID - LQR on translation - Integrated LQR max. values comparison

# 5 | Model Predictive Control

The Model Predictive Control (MPC) is part, as the LQR, of the big family of optimal controllers. MPC combines the benefits of feedback control and constrained optimization. Firstly, the mathematical formalization of the general MPC problem is presented, then the mathematical modeling and the implementation approach are exposed together with the simulation architecture. Eventually, after the weights selection, the results are presented for the two usual configurations, with RDA and with ALO. The following formalization follows the approach presented in [49].

## 5.1. Problem formalization

The MPC is, as already introduced in Sect. 1.6, a constrained receding-horizon optimization framework in which control actions are determined by solving optimal control problems sequentially and iteratively at each time step. The optimization covers a period of $N_s$ time steps, while the control horizon is set to $N_m$. In particular, $N_s$ is the length of the prediction (or output) horizon, $N_m$ is the length of the control (or input) horizon, with $N_m \leq N_s$. If $N_s$ is infinite, the problem is called an infinite horizon problem, otherwise a finite horizon problem.

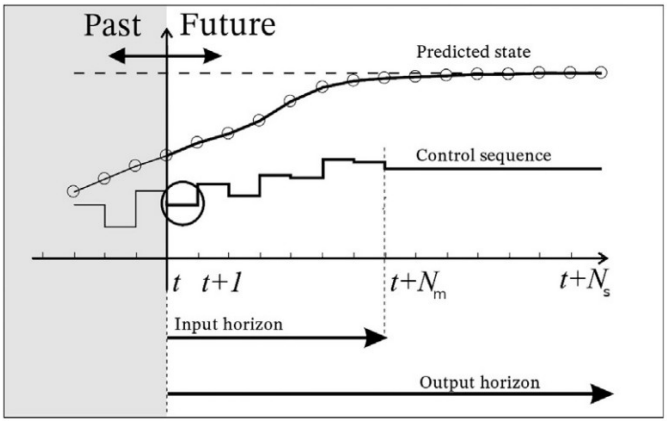


Figure 5.1: Prediction horizon scheme.

In Fig. 5.1 a scheme is shown that presents prediction (output) and control (input) horizon. For the MPC working principle, only a total of $N_m$ computed control actions are executed by the plant. At the next time step, the horizon is moved one step forward and the solver solves the optimization problem again in the span of the prediction horizon.

Hence, the three fundamental parameters of the MPC to be carefully selected are:

- **Prediction Horizon.** The prediction horizon, also known as receding horizon, is computed as $p_H = N_s \Delta t$. It defines how far in the future the controller is able to predict the behavior of the states looking for its optimal evolution. A low value of the prediction horizon results in a controller not able to exploit the benefits of this control strategy. On the other hand, if the prediction horizon is too large, the computational time could become excessive in terms of on-board implementation, and this would jeopardize the feasibility of the mission.
- **Sampling Time.** The sampling time $\Delta t$ is the same as previously used, $\Delta t = 0.1\,s$, related to the operational time discretization of the controller in the Matlab/Simulink simulator, as in the previous control applications. In the MPC context, it defines the time span between two successive optimizations.
- **Control Horizon.** The control horizon defines how many of the control actions computed by the MPC (result of the solved optimization problem) are applied. The best solution is to have $N_m = 1$, so to recompute the control actions for each time step in order to act immediately on external disturbances, and this is the strategy adopted in this study.

Hence, the optimal control problem to be solved by the MPC is formalized as follows. Considering a discrete-time dynamical system:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) \tag{5.1}$$

where, as for the LQR, $\mathbf{x}$ and $\mathbf{u}$ are, respectively, the state and control vectors. The MPC solves, at each time step $k$ of the simulation, the following finite-horizon optimal problem:

$$min_{\mathbf{u}} J(\mathbf{x_i}, \mathbf{u_i}), \;\; i = 0, ..., N_s - 1 \tag{5.2}$$

With $J$ cost function to be minimized, and $i$ being the $i-th$ step, called *stage*, in the prediction horizon length $N_s$. Eq. 5.2 is subject to:

$$\mathbf{x}_{i+1} = f(\mathbf{x}_i, \mathbf{u}_i), \;\; \mathbf{x}_0 = \mathbf{x}(k) \tag{5.3}$$

$$\mathbf{x}_i \in \mathbf{X}, \;\; \mathbf{u}_i \in \mathbf{U} \tag{5.4}$$

Where in Eq. 5.3 it is clear that the initial condition is given by the state at the $k-th$ step in which the MPC has to solve the optimal problem, while in Eq. 5.4 the constraints for the state vector and the control vector are defined, again related to the $k-th$ time instant. The function $f(...)$ represents the dynamical evolution of the state in function of the control actions, it is analogous to Eq. 5.1 and Eq. 5.3.
As was predictable, being both MPC and LQR optimal controllers, the MPC resembles the LQR scheme; indeed, MPC can be seen as an LQR with constraints and finite time horizon, solved iteratively. It is important to note that the constraints sets $\mathbf{X}$ and $\mathbf{U}$ represent one of the biggest advantages of MPC over LQR, allowing to handle explicitly physical limitations, actuator saturation, and other constraints.

The similarity between MPC and LQR is observable also looking at the form of the cost function:

$$J(\mathbf{x_k}, \mathbf{u_k}) = \sum_{i=0}^{N_s-1} (\mathbf{x}_{k+i}^T \mathbf{Q} \mathbf{x}_{k+i} \; + \; \mathbf{u}_{k+i}^T \mathbf{R} \mathbf{u}_{k+i}) \; + \; \mathbf{x}_{k+N_s}^T \mathbf{S} \mathbf{x}_{k+N_s} \tag{5.5}$$

This is essentially Eq. 4.7 but in discrete form, with $\mathbf{Q}$ and $\mathbf{R}$ being the usual cost matrices for states and controls. The terminal weight is represented by matrix $\mathbf{S}$ which is obtained, as for the unconstrained LQR, from the algebraic Riccati equation. Therefore, the cost function is the summation of the cost objective throughout the prediction horizon time.
Before presenting the mathematical modeling and the implementation approach, it is important to define the landing scenario with the description of the operational modes design, in order to clarify how the different control strategies are employed throughout the descent.

### 5.1.1. Operational modes design

As already stated, the landing maneuver can be divided into two main different phases: first the divert maneuver, then the final vertical descent on the target.
The strategy adopted to control the lander during the landing maneuver in this scenario is

to switch the controller mode from MPC to LQR at the end of the divert maneuver. This is done because as the vehicle approaches ground, as explained in [36], a standard receding-horizon MPC may continue to "pushing" the terminal event forward: the predicted landing time is repeatedly postponed, so the optimizer continues to generate commands that achieve landing in the future rather than committing to touchdown. This behavior can be solved with a decreasing-horizon MPC, which is used to prevent the aforementioned problem by reducing the prediction horizon near the end of the maneuver; in this work, switching to LQR in the terminal phase plays an equivalent role, in order to practically reduce the horizon to a local feedback law which drives the vehicle to a safe soft landing. The LQR controller adopted is the same as that discussed in its dedicated chapter.

The main difference between the two phases of the landing maneuvers is that the trajectory profile of reference provided by the guidance in the first part is pre-computed off-line, while during the vertical descent it is computed online. This is the reason why the switch moment is chosen to be at the end of the divert maneuver, since simultaneous switching of the controller and the guidance mode yielded reliable performances.
This hybrid control architecture allows the strengths of MPC to be exploited during the most demanding phase of the maneuver, while relying on the simplicity and robustness of LQR control during the final descent.

## 5.2. Mathematical Modeling

In this section, the system dynamics modeling is exposed, which consists of a detailed description of the $f(...)$ function in Eq. 5.1. Then the cost function adopted for the pinpoint landing on Mars scenario is presented.

### 5.2.1. Dynamics and state function

The implemented controller is an integrated translational and rotational controller, whose dynamics uses as a state the translational and rotational variables together with the mass of the lander, and as input the forces and torques computed to control the vehicle.
The state vector is defined as:

$$\mathbf{x} = \begin{bmatrix} x & y & z & v_x & v_y & v_z & q_x & q_y & q_z & q_w & \omega_x & \omega_y & \omega_z & m \end{bmatrix}^T \tag{5.6}$$

Where $\mathbf{r} = [x\ y\ z]^T$ is the position, $\mathbf{v} = [v_x\ v_y\ v_z]^T$ is the velocity, $\mathbf{q} = [q_x\ q_y\ q_z\ q_w]^T$ is the attitude quaternion, $\boldsymbol{\omega} = [\omega_x\ \omega_y\ \omega_z]^T$ is the angular velocity, and $m$ is the mass of the

vehicle.
The control input is:

$$\mathbf{u} = \begin{bmatrix} F_x & F_y & F_z & \tau_x & \tau_y & \tau_z \end{bmatrix}^T \tag{5.7}$$

where $\mathbf{F} = [F_x\ F_y\ F_z]^T$ is the thrust force and $\boldsymbol{\tau} = [\tau_x\ \tau_y\ \tau_z]^T$ is the control torque.
The translational dynamics is given by:

$$\dot{\mathbf{r}} = \mathbf{v}. \tag{5.8}$$

The NMPC requires an accurate nonlinear prediction model, hence Coriolis and centrifugal effects are included explicitly, to propagate the state over the horizon. The centrifugal and Coriolis terms are:

$$\mathbf{a}_{cf} = \boldsymbol{\omega}_M \times (\boldsymbol{\omega}_M \times \mathbf{r}), \qquad \mathbf{a}_{cor} = 2\,\boldsymbol{\omega}_M \times \mathbf{v}. \tag{5.9}$$

where $\boldsymbol{\omega}_M$ is the Mars rotation rate.
Considering the landing site with latitude $\theta = 30°$, and the reference system centered on the target ENU, with $i$ oriented to East, $j$ oriented to North, and $k$ oriented Upwards along the nadir-zenith direction; Mars angular velocity is expressed as:

$$\boldsymbol{\omega}_M = \begin{bmatrix} 6.139 \cdot 10^{-5} \\ 0 \\ 3.544 \cdot 10^{-5} \end{bmatrix} \quad [rad/s] \tag{5.10}$$

The translational acceleration is modeled as follows:

$$\dot{\mathbf{v}} = \frac{1}{m}\mathbf{F} - \mathbf{a}_{cf} - \mathbf{a}_{cor} + \mathbf{g}_M \tag{5.11}$$

where $\mathbf{g}_M$ is the gravitational acceleration vector on Mars, which, expressed in ENU reference frame, is:

$$\mathbf{g}_M = \begin{bmatrix} 0 \\ 0 \\ -3.72 \end{bmatrix} \quad [m/s^2] \tag{5.12}$$

As for the attitude equations, the quaternion dynamics is given by:

$$\dot{\mathbf{q}} = \frac{1}{2}\Omega(\boldsymbol{\omega})\,\mathbf{q} \tag{5.13}$$

With the matrix $\Omega(\boldsymbol{\omega})$:

$$\Omega(\boldsymbol{\omega}) = \begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix}. \tag{5.14}$$

The angular rate dynamics, in the approximation of out of diagonal terms of inertia much lower than the diagonal ones, is::

$$\dot{\boldsymbol{\omega}} = \begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} = \begin{bmatrix} \dfrac{\tau_x - (I_{zz} - I_{yy})\omega_y\omega_z}{I_{xx}} \\ \dfrac{\tau_y - (I_{xx} - I_{zz})\omega_z\omega_x}{I_{yy}} \\ \dfrac{\tau_z - (I_{yy} - I_{xx})\omega_x\omega_y}{I_{zz}} \end{bmatrix} \tag{5.15}$$

where $(I_{xx}, I_{yy}, I_{zz})$ are the principal inertia moments, numerically taken from Eq. 4.48. Eventually, the mass evolution is modeled as follows. With the thrust magnitude being:

$$T = \|\mathbf{F}\|_2 = \sqrt{F_x^2 + F_y^2 + F_z^2} \tag{5.16}$$

The mass flow rate model is retrieved from the Tsiolkovsky equation as follows.

$$\dot{m} = -\frac{T}{I_{sp} g_0} \tag{5.17}$$

Here $I_{sp} = 205\,s$ is the specific impulse for the MR80 thrusters and $g_0 = 9.806\ m/s^2$ is the standard gravitational acceleration on Earth.
The state function which represents the evolution of the system is expressed as the derivative of the state $\mathbf{x}$ in function of the control actions $\mathbf{u}$, as shown in the previous equations, and can be summarized with:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{r}} \\ \dot{\mathbf{v}} \\ \dot{\mathbf{q}} \\ \dot{\boldsymbol{\omega}} \\ \dot{m} \end{bmatrix} \tag{5.18}$$

The system of equations is non-linear, as can be seen from the terms of translational acceleration in Eq. 5.11 where there is the dependence from $1/m$, the mass $m$ being a state. In the expression of the mass flow rate as well, there is a second order term inside the thrust magnitude in Eq. 5.16. The Coriolis and centrifugal accelerations are non-linear as well, since they present the cross-product in Eq. 5.9.
Eventually, also in the rotational dynamics, there are non-linear terms. In Eq. 5.13 the quaternion dynamics sees the product between the matrix $\Omega$, which depends on the angular velocity which is a state, and the quaternion vector itself. Finally, the angular rate dynamics presents non-linearity too having the product between different components of the angular velocity vector.
All these non-linear terms, which identify the function $f(...)$ for the dynamics in Eq. 5.1 as non-linear, bring to the extension of the classical MPC problem to a **Non-Linear Model Predictive Control (NMPC)** problem.

### 5.2.2. Cost function

For the application of this work, the decision has been taken not to include in the cost J the contribution related to the terminal cost given by the **S** matrix in Eq. 5.5. Terminal weights are commonly introduced to obtain formal stability guarantees; however, deriving a meaningful matrix **S** as terminal weight for the full nonlinear landing dynamics could require repeated local linearizations or solving a nonlinear Riccati-like equation each time step, significantly increasing the computational burden.
This approach is also justified because a cost term, depending on the final errors with respect to the target at touchdown, is already included in the guidance algorithm. Hence, omitting terminal weights simplifies the problem and reduces computational effort, while still providing reliable results. The cost function for the $k-th$ time instant of the simulation, after these considerations, is expressed as:

$$J_k = \sum_{i=1}^{N_s} \left( \mathbf{e}_i^T \mathbf{Q} \mathbf{e}_i \, + \, \mathbf{u}_i^T \, \mathbf{R} \, \mathbf{u}_i \right) \tag{5.19}$$

The summation index is shifted to start from $i = 1$, corresponding to the first predicted state after the initial condition. This choice is equivalent to the standard formulation and is adopted here for notational convenience.
The analyzed scenario is still a tracking problem, therefore the state vectors is represented as tracking error with respect to the reference profile.
In particular, the state vector is:

$$\mathbf{e_i} = \mathbf{x}_i^p - \mathbf{x}_i^{ref} \tag{5.20}$$

Where $\mathbf{x}_i^p$ is the $i-th$ predicted state obtained through propagation performed directly by the NMPC, through numerical integration using the implicit trapezoidal rule, with the right hand side provided by the non-linear state function presented in Sect. 5.2.1.

With $N_s$ being the length of the prediction horizon, so the number of steps in which control actions are optimized, a total of $N_s$ input $u_1, ..., u_{N_s}$ are defined. Starting from the estimated initial state $\hat{\mathbf{x}}_\mathbf{k}$, provided as input by the Navigation section, an integrated integrator in the NMPC propagates, using the state function, the dynamics forward by applying the $N_s$ sequence of control actions. Since the initial state is included in the prediction, the resulting predicted state sequence $\mathbf{x^p}$ contains $N_s + 1$ states.
The predicted terminal state corresponds to the result obtained after the last optimized control action. Since no control action is associated with this state, it is not included in the stage cost, as can be seen by the indexing $i$ of the summation in the cost function in Eq. 5.19, which runs from $i = 1$ to $i = N_s$.

The state $\mathbf{x}_i^{ref}$ is the $i-th$ vector of the set of reference states $\mathbf{x}^{ref}$ built by propagating for the same time span the reference state provided by the Guidance section at the time step $k$. This propagation is performed using a dedicated *Reference Builder* function later presented.
Both the aforementioned vectors are composed of the same elements shown in Eq. 5.6 with the exception of the vehicle mass. The mass is included in the state vector to perform proper propagation of the states, taking into account the mass flow rate, in order to have a correct model of the system dynamics. However, it is not included in the optimization problem, since there is no reference evolution of the mass state. Therefore, the tracking error is defined only for the states for which a reference profile is provided, while the mass of the vehicle is treated as an internal variable for the system dynamics and it is excluded from the cost function.

Regarding the term relative to the control action in Eq. 5.19, $\mathbf{u}_i$ is indeed the $i-th$ control action built as in Eq. 5.7. The complete set of control actions $\mathbf{u}$ of length $N_s$ is the output of the optimization problem solved by the solver at the time-step $k$. It should be remembered that only the first element of this set is going to be applied to control the lander, and at the next time step $k+1$ the optimization process is repeated.

## 5.3. Implementation approach

The working principle of the Non-Linear Model Predictive Control discussed before is summarized with the block scheme in Fig. 5.2, in order to facilitate the presentation of the implementation approach.
In the following block scheme, it is possible to appreciate all the elements introduced in the previous section. The NMPC takes in input the estimated state and the set of $N_s + 1$ reference states. The estimated state is propagated with the state function, and the difference between this propagation and the set of reference states provides the state error. The output of the NMPC is the set of controls **u**, and only the first element of this set is applied, then the process is repeated.

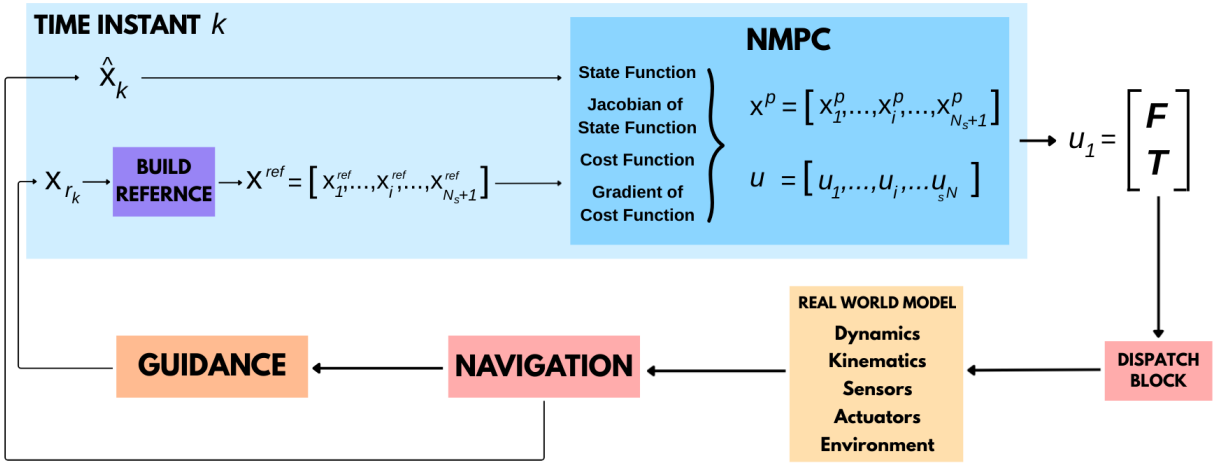


Figure 5.2: NMPC working principle block scheme.

In the NMPC block, it is possible to see two more functions not cited yet: the jacobian of the state function and the gradient of the cost function. These two are provided to the solver in order to speed up the runtime of the simulation, and they will be discussed later.
The design and implementation choices are now presented.

### 5.3.1. MPC fundamental parameters

The fundamental parameters, introduced in Sect. 5.1, are the prediction horizon, the sampling time, and the control horizon.
In this work, the following considerations are made:

– *Control Horizon.* The control horizon is set to be, as already stated, equal to one to have a more robust solution. This was already clear from the previous treatment

about the NMPC working principle, where it was stated that only the first element of the computed control actions is applied.

– *Sampling Time.* The sampling time is the same as that used in the LQR controller: $\Delta t = 0.1\,s$.

– *Prediction Horizon.* The prediction horizon $p_H$ is chosen in order to have a prediction of 12 steps, so $N_s = 12$ and $p_H = 1.2\,s$. A prediction horizon of 12 steps is sufficient to foresee the future behavior of the dynamics of the landing phase, while avoiding excessively long horizons that would increase computational complexity without providing substantial performance advantages. This choice is also driven by the already small sample time $\Delta t$; increasing $p_H$ leads to an excessive computational burden for a realistic on-board implementation.

### 5.3.2. Solving algorithm and block scheme

All the elements presented in Fig. 5.2, which can be taken as a block-scheme reference, have to be implemented in the Matlab/Simulink environment. The implementation is performed exploiting the Simulink block *Nonlinear Model Predictive Control Multiple Stages*. This block works exactly as explained before for the NMPC, and provides as output $u_1$. The *Multiple Stages* refers to the different $i-th$ instants, which are indeed called stages. It is specified since this block allows to assign different weights and penalize differently the states and the controls (using different matrices $\mathbf{Q}$ and $\mathbf{R}$) for different stages. However, this property is not exploited in this study, as shown later in this document. The Simulink NMPC Multistage block is presented in Fig. 5.3.

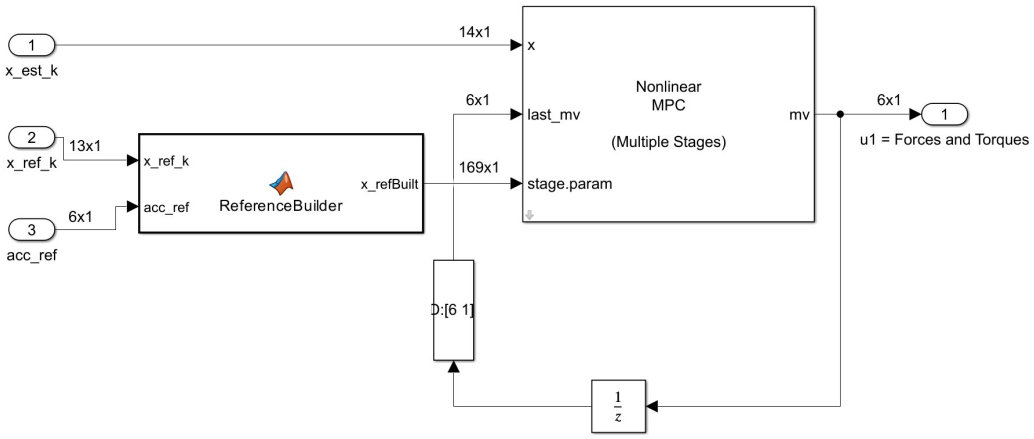


Figure 5.3: NMPC Simulink block.

The input `x_est_k` is $14 \times 1$ since it includes the mass estimation, modeled as shown in

the dynamics equations with Eq. 5.17, while the input `x_ref_k` is only $13 \times 1$ since the mass depletion is not subject to optimization.
The `stage.param` input is the reference profile built inside the `ReferenceBuilder` function, later discussed. This input is $169 \times 1$ since the NMPC block requires to have a column vector presenting the $N_s + 1$, hence 13 in this case, reference states sequence; being the state length equal to 13, the column vector becomes a $13 \times 13 = 169$ elements vector.
The last input depicted, `last_mv`, is simply the previous output of the NMPC block that goes through a Unit Delay block which holds the value until the next time instant, so at instant $k$ the `last_mv` input is the output `mv` of instant $k-1$. The initial condition $\mathbf{u_0}$ of the Unit Delay is set to be equal to the initial hovering force, hence:

$$\mathbf{u}_0 = \begin{bmatrix} 0 & 0 & m\,g_M & 0 & 0 & 0 \end{bmatrix}^T \tag{5.21}$$

With $g_M$ Mars gravitational acceleration, and $m = 1500\,kg$ the initial mass of the vehicle, provided by TAS-I.
The output of the Unit Delay block goes through a Signal Specification block used only to force the vector to have $6 \times 1$ dimensions.

The output `mv` stands for Manipulated Variables, which are indeed the variables that the solver directly manipulated in order to guide the dynamics, and corresponds to the translational forces and rotational torques applied to control the vehicle, hence the $6 \times 1$ control vector $\mathbf{u}_1$ as in Eq. 5.7, computed each time step.

Inside the NMPC block, the main parameter required is a Multistage Nonlinear MPC Controller object, which is called `lander` in this study. The `lander` object is defined in Matlab, using the `nlmpcMultistage` Matlab function. To build this object, different parameters are addressed to it: the sample time, the prediction horizon, the state function with its jacobian, the constraints sets $\mathbf{X}$ and $\mathbf{U}$, as well as the cost function with its gradient for each stage.
Before looking at all these parameters and how they are implemented and defined in the `lander` object, the overview of the NMPC Simulink block in Fig. 5.3 is concluded by analyzing the `ReferenceBuilder` function, used to build `x_refBuild`.

## Reference Builder function

As shown in Fig. 5.3, this Matlab function needs in input both the reference state provided by the guidance at the time instant $k$ and the accelerations, both translational

and angular, of the reference profile at the same time instant. The function performs a discretized propagation over the prediction horizon, thus providing a sequence of $N_s + 1$ reference states, stored in a column vector of dimensions $169 \times 1$.
For each stage $i$ the following vector is built:

$$\mathbf{s}_i = \begin{bmatrix} \mathbf{p}_i \\ \mathbf{v}_i \\ \mathbf{q}_i \\ \boldsymbol{\omega}_i \end{bmatrix} \in \mathbb{R}^{13} \tag{5.22}$$

Passing to the solver the full set as:

$$\mathbf{s} = \left[\mathbf{s}_1^T, ..., \mathbf{s}_\mathbf{i}^\mathbf{T}, ..., \mathbf{s}_{N_s+1}^T\right] \tag{5.23}$$

For each step, the translational propagation is performed as a constant-acceleration preview:

$$\mathbf{p}_{i+1} = \mathbf{p}_i + \mathbf{v}_i \, \Delta t + \frac{\Delta t^2}{2} \mathbf{a}_{r_i} \tag{5.24}$$

$$\mathbf{v}_{i+1} = \mathbf{v}_i + \Delta t \, \mathbf{a}_{r_i} \tag{5.25}$$

where $\Delta t$ is the usual sample time of the controller $\Delta t = 0.1\,s$ and $\mathbf{a}_{r_i}$ is the reference acceleration. The rotational propagation, instead, under the assumption of small angular variations, can be approximated by:

$$\mathbf{q}_{i+1} = \mathbf{q}_i + \frac{1}{2} \Omega(\bar{\boldsymbol{\omega}}_i) \mathbf{q}_i \Delta t \tag{5.26}$$

$$\boldsymbol{\omega}_{i+1} = \boldsymbol{\omega}_i + \boldsymbol{\alpha}_{r_i} \Delta t \tag{5.27}$$

In Eq. 5.26, which represents the quaternion kinematics, instead of the continuous expression in Eq. 5.13, a discretized integration is used. The matrix $\Omega(\boldsymbol{\omega})$ is the same as presented in Eq. 5.14, with the difference that instead of using the initial angular velocity, it is used the average angular velocity over the sampling interval. This choice corresponds to an approximation of the middle point of the continuous-time kinematic equation and provides a more accurate and numerically robust discretization.

Before going to the next time step, quaternion normalization is performed at the end of each step to preserve the unit-norm constraint.

$$\mathbf{q}_{i+1} = \frac{\mathbf{q}_{i+1}}{\|\mathbf{q}_{i+1}\|} \tag{5.28}$$

The purpose of this propagation is not to provide an accurate prediction of the lander trajectory, but to generate a consistent reference preview for the NMPC Multiple Stage block. The controller dynamics are ruled by the full nonlinear model, while the reference propagation is used only as an ideal trajectory preview.

## Lander NMPC object

The `lander` object is the main protagonist of the modeling and implementation of the NMPC. It is implemented in the main Matlab script that exploits the Matlab function `nlmpcMultistage` as:

```
lander = nlmpcMultistage(pH,14,MV=1:6);
```

This function creates the proper object to be provided to the Nonlinear MPC Multiple Stages block in Simulink. The input `pH` is the number of prediction steps, hence 12. The second input indicates the length of the state, and the third input the length of the manipulated variables output vector.
The constraints are addressed directly to this object by using:

```
lander.States(i).Min = ...,  lander.States(i).Max = ...
lander.MV(i).Min = ...,  lander.MV(i).Max = ...
```

The first two lines define the constraints relative to the states, while the second two lines are relative to the constraints on the control actions; hence, respectively, the systems $\mathbf{X}$ and $\mathbf{U}$ in Eq. 5.4.
The state constraints chosen in this scenario are the following:

$$\begin{cases} z \geq 0 \\ 0 \leq m \leq 1500 \end{cases} \tag{5.29}$$

The first one to ensure the physical constraint of not having a planned trajectory which sees an altitude lower than zero meters at any time instant. The second one is imposed for consistency, but it is superfluous, being impossible to have a mass higher than the initial one and having a propellant consumption that has maximum values around 300 kg, hence

with a minimum mass usually around 1200 kg. The control constraints imposed are:

$$\begin{cases} -1500 \leq F_x \leq 1500 \\ -1500 \leq F_y \leq 1500 \\ 0 \leq F_z \leq 15000 \\ -2500 \leq \tau_x \leq 2500 \\ -2500 \leq \tau_y \leq 2500 \\ -1000 \leq \tau_z \leq 1000 \end{cases} \tag{5.30}$$

These values are chosen by looking at the trend of the forces $F$ and torques $\tau$ in the simulations with the LQR controller.
The force on the $z$ axis has a lower boundary of zero newtons, since the lander is not supposed to gain altitude, therefore $F_z$ should not become negative at any time instant. The upper boundary of $F_z$ instead is imposed to be large, $15000\,N$, in order to give more room of operation to the solver, since if some solutions exceed reasonable values, there is still the Dispatch Block which handles saturation and thrusters limits.
As for the torques constraints, the boundaries of $\tau_z$ are narrower, since from the simulations with the LQR controller it was observable that generally a lower torque was required on the $z$ axis, which is understandable since the lander is symmetric on that axis and the main problems of the attitude controller are related to the off vertical displacement.

In addition to the constraints, as stated before, the functions are allocated to the lander object too. In particular, the state function described in Sect. 5.2.1 is stored in a Matlab function called `MMLstatefcn`, the same is done for its jacobian, which is stored in `MMLstatejac`. These two are assigned to the lander object as follows, in order to make them available to the NMPC Simulink block.

```
lander.Model.StateFcn = @MMLstatefcn;
lander.Model.StateJacFcn = @MMLstatejac;
```

The same procedure is performed for the cost function, stored in `MMLcostfcn`, and its gradient, whose associated function is `MMLcostgrad`. The difference with the state function is that in the multistage NMPC formulation, the cost function has to be defined for each stage, therefore it must be explicitly assigned to each prediction stage, even when the same cost function is used during all the horizon. Hence, the cost function is assigned to the object with the following *for* cycle:

```
for i=1:pH+1
```

```
    lander.Stages(i).CostFcn = @MMLcostfcn;
    lander.Stages(i).CostJacFcn = @MMLcostgrad;
    lander.Stages(i).ParameterLength = 13;
end
```

Therefore, for each of the 13 stages, the cost function is the same. The last line defines the `ParameterLength` variable, which is needed by the solver to know the length of the reference state for each stage $i$.
Now, the modeling equations for the jacobian of the state function and the gradient of the cost function can be presented.

## Jacobian of the state function

This jacobian is provided to the solver in order to speed up the simulation and to lower the computational burden; since without providing it, the solver would retrieve the jacobian numerically each time step, significantly adding computational calculations.
For a system subject to a generic state function:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \tag{5.31}$$

The jacobian is expressed with the following two matrices:

$$\mathbf{A} = \left.\frac{\partial f}{\partial x}\right|_{(x,u)}, \quad \mathbf{B} = \left.\frac{\partial f}{\partial u}\right|_{(x,u)} \tag{5.32}$$

Having $\mathbf{x} \in \mathbb{R}^{14}$ as in Eq. 5.6 and $\mathbf{u} \in \mathbb{R}^{6}$ as in Eq. 5.7, matrix $\mathbf{A}$ has dimensions of $14 \times 14$ and matrix $\mathbf{B}$ has dimensions of $14 \times 6$. The 14 rows for each of the matrices is due to the length of the derivative of the state vector $\dot{\mathbf{x}}$ in Eq. 5.18, representing the state function $f$.
The resulting matrices obtained after the partial derivations are hereafter reported. Starting from the $\mathbf{A}$ matrix, it assumes the following form:

$$\mathbf{A} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_{3\times4} & \mathbf{0}_3 & \mathbf{0}_{3\times1} \\ \mathbf{A}_{ar} & \mathbf{A}_{av} & \mathbf{0}_{3\times4} & \mathbf{0}_3 & \mathbf{A}_{am} \\ \mathbf{0}_{4\times3} & \mathbf{0}_{4\times3} & \mathbf{A}_{\dot{q}q} & \mathbf{A}_{\dot{q}\omega} & \mathbf{0}_{4\times1} \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_{3\times4} & \mathbf{A}_{\dot{\omega}\omega} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & \mathbf{0}_{1\times4} & \mathbf{0}_{1\times3} & 0 \end{bmatrix} \tag{5.33}$$

All the zero terms are due to the non-dependency of the component of the state function from the relative state. This can be checked with the equations presented in Sect. 5.2.1. The non-zero terms are:

- $\mathbf{A}(1{:}3,\ 4{:}6) = \mathbf{I}_3$. This is the derivative of the velocity vector with respect to itself, being equal to the identity matrix.
- $\mathbf{A}(4{:}6,\ 1{:}3) = \mathbf{A}_{ar}$. This is the derivative of the translational acceleration with respect to the position vector:

$$\mathbf{A}_{vr} = \frac{\partial \dot{\mathbf{v}}}{\partial \mathbf{r}} = -\left(\boldsymbol{\omega}_M \boldsymbol{\omega}_M^T - \|\boldsymbol{\omega}_M\|^2 \mathbf{I}_3\right) \tag{5.34}$$

  with $\boldsymbol{\omega}_M$ Mars angular velocity vector as in Eq. 5.10.

- $\mathbf{A}(4{:}6,\ 4{:}6) = \mathbf{A}_{av}$. This is the derivative of the translational acceleration with respect to the velocity vector:

$$\mathbf{A}_{vv} = \frac{\partial \dot{\mathbf{v}}}{\partial \mathbf{v}} = -2[\boldsymbol{\omega}_M]_\times \tag{5.35}$$

  With

$$[\boldsymbol{\omega}]_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{5.36}$$

- $\mathbf{A}(4{:}6,\ 14) = \mathbf{A}_{am}$. Derivative of the translational acceleration with respect to the mass:

$$\mathbf{A}_{vm} = \frac{\partial \dot{\mathbf{v}}}{\partial m} = -\frac{1}{m^2}\mathbf{F} \tag{5.37}$$

- $\mathbf{A}(7{:}10,\ 7{:}10) = \mathbf{A}_{\dot{q}q}$. Represents the partial derivative of $\dot{\mathbf{q}}$, derivative of the quaternion vector, with the quaternion itself.

$$\mathbf{A}_{\dot{q}q} = \frac{\partial \dot{\mathbf{q}}}{\partial \mathbf{q}} = \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega}) \tag{5.38}$$

  with $\Omega(\boldsymbol{\omega})$ as in Eq. 5.14.

- $\mathbf{A}(7{:}10,\ 11{:}13) = \mathbf{A}_{\dot{q}\omega}$. Derivative of $\dot{\mathbf{q}}$ with respect to the angular velocity vector.

$$\mathbf{A}_{\dot{q}\omega} = \frac{\partial \dot{\mathbf{q}}}{\partial \boldsymbol{\omega}} = \frac{1}{2}\mathbf{E}(\mathbf{q}) \tag{5.39}$$

With

$$\mathbf{E}(\mathbf{q}) = \begin{bmatrix} q_w & -q_z & q_y \\ q_z & q_w & -q_x \\ -q_y & q_x & q_w \\ -q_x & -q_y & -q_z \end{bmatrix} \tag{5.40}$$

- $\mathbf{A}(11{:}13,\ 11{:}13) = \mathbf{A}_{\dot{\omega}\omega}$. This is the derivative of the angular acceleration with respect to the angular velocity.

$$\mathbf{A}_{\dot{\omega}\omega} = \frac{\partial \dot{\boldsymbol{\omega}}}{\partial \boldsymbol{\omega}} = -\mathbf{M}^{-1}\left([\boldsymbol{\omega}]_\times \mathbf{M} - [\mathbf{M}\boldsymbol{\omega}]_\times\right) \tag{5.41}$$

  with $\mathbf{M}$ being the matrix of inertia, as in Eq. 4.48.

- $\mathbf{A}(14,\ 1{:}14) = 0_{1\times 14}$. Note that the row relative to the mass presents all zero-terms, since $\dot{m}$ does not depend on the states.

$$\frac{\partial \dot{m}}{\partial \mathbf{x}} = \mathbf{0}_{1\times 14} \tag{5.42}$$

Now the $\mathbf{B}$ matrix is presented, again in compact form with an overview of the non-zero terms.

$$\mathbf{B} = \frac{\partial \mathbf{f}}{\partial \mathbf{u}} = \begin{bmatrix} \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} \\ \frac{1}{m}\mathbf{I}_3 & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{4\times 3} & \mathbf{0}_{4\times 3} \\ \mathbf{0}_{3\times 3} & \mathbf{M}^{-1} \\ \mathbf{B}_{mF} & \mathbf{0}_{1\times 3} \end{bmatrix} \tag{5.43}$$

Here, the non-zero terms are:

- $\mathbf{B}(4{:}6,\ 1{:}6)$. This is the derivative of the velocity vector with respect to the forces:

$$\frac{\partial \dot{\mathbf{v}}}{\partial \mathbf{F}} = \frac{1}{m}\mathbf{I}_3 \tag{5.44}$$

- $\mathbf{B}(11{:}13, 4{:}6)$. Derivative of the angular accelerations with respect to the torques:

$$\frac{\partial \dot{\boldsymbol{\omega}}}{\partial \boldsymbol{\tau}} = \mathbf{M}^{-1} \tag{5.45}$$

- $\mathbf{B}(14, 1{:}3)$. This term represents the derivative of the mass flow rate with respect to the forces:

$$\frac{\partial \dot{m}}{\partial \mathbf{F}} = -\frac{1}{I_{sp} g_0} \frac{\mathbf{F}^T}{\|\mathbf{F}\|}, \qquad \|\mathbf{F}\| > 0 \tag{5.46}$$

  with $I_{sp} = 205\, s$ and $g_0 = 9.806\, m/s^2$ as already stated after Eq. 5.17.

## Gradients of the cost function

The gradients of the cost function are retrieved in the same way as the jacobian matrices for the state function, so performing the partial derivatives of the cost function, presented in its dedicated section with Eq 5.19, with respect to the state and the control. Note that the state in this context is the error vector $\mathbf{e}$.
The derivative with respect to the state provides:

$$\frac{\partial J}{\partial \mathbf{x}_{1:13}} = 2\mathbf{Q}\mathbf{e} \tag{5.47}$$

$$\frac{\partial J}{\partial x_{14}} = 0 \tag{5.48}$$

Note that the mass is not a state subject to optimization and, hence, the derivative with respect to the mass is zero.
Therefore, the full gradient of $J$ with respect to the state is:

$$\nabla_{\mathbf{x}} J = \begin{bmatrix} 2\mathbf{Q}\mathbf{e} \\ 0 \end{bmatrix} \tag{5.49}$$

As for the gradient computed with respect to the control actions:

$$\nabla_{\mathbf{u}} J = 2\mathbf{R}(\mathbf{u} - \mathbf{u}^{ref}) \tag{5.50}$$

These are the gradients provided to the solver in order to speed up the simulation.

With the discussion of the lander object with its constraints, the jacobian of the state and the gradient of the cost, now the problem formulation and implementation approach is completely presented.
The remaining aspect to be addressed, before presenting the results obtained, is the selection of the weights in the weighting matrices **Q** and **R** of the cost function.

## 5.4. Weights selection - RDA configuration

After a refined tuning, the matrix **Q** is chosen to be diagonal and with constant weights:

$$\mathbf{Q} = diag\left(\begin{bmatrix} 10 & 10 & 40 & 30 & 30 & 70 & 130 & 130 & 130 & 130 & 80 & 80 & 80 \end{bmatrix}\right) \tag{5.51}$$

The proportionality is clear, having the weights of the attitude controller being higher than the weights on the translational controller. This is because the rotational loop must be inherently faster than the translational loop in order to guarantee, in advance, the off-vertical angle consistent with the needed vertical component of the thrust.

In the translation weights, the position and velocity on $z$ are weighted more than the horizontal ones, since the $v_z$ parameter is more critical, especially considering the soft landing requirement, and needs higher weights to ensure the satisfaction of the requirements.
Regarding the **R** matrix, it is tuned as:

$$\mathbf{R} = diag\left(\begin{bmatrix} 5 \cdot 10^{-3} & 5 \cdot 10^{-3} & 5 \cdot 10^{-5} & 5 \cdot 10^{-6} & 5 \cdot 10^{-6} & 5 \cdot 10^{-6} \end{bmatrix}\right) \tag{5.52}$$

The higher an element of **R** is, the more expensive it is to use that control action. In the selected tuning for the matrix, rotational torques are penalized less than translational forces, allowing tight attitude control.
The lateral thrust components, instead, have the highest penalties to discourage unnecessary horizontal motion. The usage of NMPC during the divert maneuver allowed the LQR controller in the vertical descent phase to be more relaxed with less stringent weights on the attitude control part, enhancing the performances regarding the horizontal distance at touchdown.

The new set of $\sigma$ (the altitude-varying weights of **Q** matrix for the LQR controller) for both translation and rotation are reported in the next subsection.

### 5.4.1. New LQR weights

The only changes in the LQR translational weights, presented in Sect. 4.2.2, are in the $\sigma$ relative to the position in $x$ and $y$, where the second element is changed from 20 to 10:

$\sigma_{p_x}$ = [10 10 20 30 40 50 50 60 60 60 70 80 90 100 100];
$\sigma_{p_y}$ = [10 10 20 30 40 50 50 60 60 60 70 80 90 100 100];

In this way, the solver is allowed to spend more propellant in order to land closer to the target. The other weights are the same. The new rotational set of $\sigma$ is:

$\sigma_{a_x}$ = [7 7 7 10 10 10 10 10 10 8 12 16 20 24 28];
$\sigma_{a_y}$ = [7 7 7 10 10 10 10 10 10 8 12 16 20 24 28];
$\sigma_{a_z}$ = [180 180 180 180 180 180 180 180 180 180 180 180 180 180 180];
$\sigma_w$ = [180 180 180 180 180 180 180 180 180 180 180 180 180 180 180];

As anticipated before, the rotational control is generally more relaxed with relatively higher weights with respect to the ones previously chosen in Sect. 4.3.2.

## 5.5. Results - RDA configuration

The results are presented here after with the usual table used before in this thesis.

| | Min | 0.27% | 1% | 50% | 99% | 99.73% | Max | Req. |
|---|---|---|---|---|---|---|---|---|
| **Hor. Distance** [m] | 0.395 | 0.395 | 0.817 | 24.852 | 38.940 | 40.385 | 40.385 | 60 |
| **\|Vert Velocity\|** [m/s] | 0.581 | 0.581 | 0.615 | 0.952 | 1.179 | 1.189 | 1.189 | 2.5 |
| **Hor. Velocity** [m/s] | 0.144 | 0.144 | 1.038 | 1.038 | 1.700 | 1.722 | 1.722 | 2 |
| **Off Vert Angle** [°] | 0.028 | 0.028 | 0.055 | 0.929 | 1.960 | 1.995 | 1.995 | 7 |
| **Ang. @Spin Ax.** [°] | -5.333 | -5.333 | -4.915 | 0.471 | 7.299 | 7.959 | 7.596 | |
| **Trans. Ang. Rate** [°/s] | 0.021 | 0.021 | 0.023 | 0.167 | 0.618 | 0.761 | 0.761 | 3.2 |
| **Rate @Spin Ax.** [°/s] | -1.042 | -1.042 | -0.991 | 0.002 | 1.886 | 1.914 | 1.914 | |
| **Prop. Mass** [Kg] | 188.150 | 188.150 | 189.153 | 239.750 | 302.878 | 303.151 | 303.151 | |

Table 5.1: MPC touchdown values, RDA configuration.

It is immediately possible to see that an improvement in landing accuracy is achieved at the expense of an increase in the propellant mass consumption.
In particular, a gain of around 10-12 meters of accuracy is obtained for each percentile of the horizontal distance magnitude with respect to the results provided by the LQR; whilst the propellant consumption as well is grown of around 20-25 kg for each percentile. The other parameters are stable and similar to the previous results obtained: the horizontal and vertical velocity present a high margin under the limit imposed by the requirement,

as well as for the rotational control part with the off-vertical angle and the transversal angular rate.
The plots regarding the horizontal distance and the propellant consumption are reported, as these two parameters are the most interesting in the results presented.

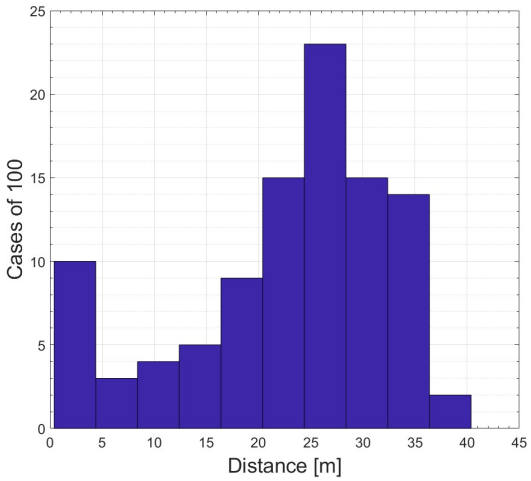


Figure 5.4: MPC with RDA - horizontal distance magnitude at TD, MC simulation.

As it is observable, only two cases have horizontal distance from the target which lies close to a distance of 40 meters; all the others present instead a lower distance, with a peak around 25-30 meters.
This accuracy pays the price of a generally higher propellant consumption, with 8 cases having a consumed mass around 300 kg, as depicted in the following histogram.

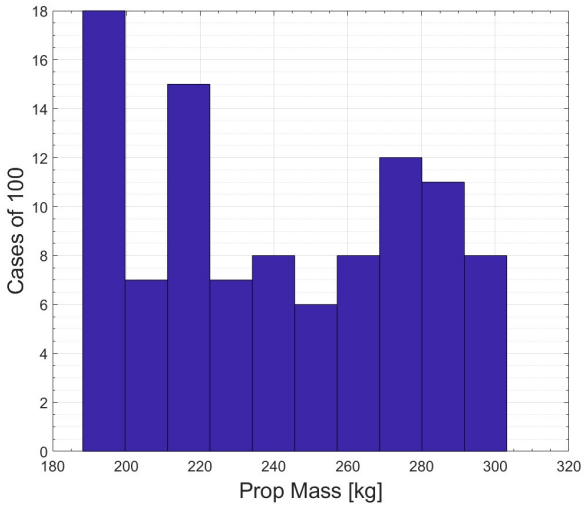


Figure 5.5: MPC with RDA - propellant mass consumption at TD, MC simulation.

Eventually, the comparison table with all the other controllers with RDA configuration is presented.
Note that the values reported are the maximum of the touchdown values of the Monte Carlo analysis performed for each controller, as done before; and the last column "MPC" is related to the configuration with MPC used for the divert maneuver, and LQR used for the final vertical descent.

| **Max Parameter** | **PID** | **LQR+PID** | **LQR** | **MPC** |
|---|---|---|---|---|
| Horizontal Distance [m] | 61.023 | 50.658 | 52.768 | 40.385 |
| \|Vertical Velocity\| [m/s] | 2.458 | 1.807 | 1.833 | 1.189 |
| Horizontal Velocity [m/s] | 1.090 | 1.174 | 1.201 | 1.722 |
| Off vertical Angle [°] | 1.733 | 5.450 | 5.277 | 1.995 |
| Transversal Ang. Rate [°/s] | 3.653 | 4.317 | 0.828 | 0.761 |
| Propellant Consumption [kg] | 292.035 | 275.36 | 275.030 | 303.151 |

Table 5.2: RDA configuration: comparison of the results provided by the different controllers implemented and tested.

The NMPC controller succeeds in further lowering the horizontal distance with respect to the previous controllers. Moreover, also the vertical velocity value is lower, providing a safer margin for the requirements of soft landing.
However, the horizontal velocity is slightly higher than the velocities presented with PID and LQR.
As for the rotational control, the performances are improved as well, having an off-vertical angle which is lower than the one obtained with LQR, and close to one obtained with PID. The transversal angular rate again presents a better value with respect to the PID, comparable to that of the LQR, respecting the boundary imposed by the requirement.
As mentioned above, the main problem is related to the propellant consumption, which sees an increase both with respect to the PID, of about 10-11 kg, and to the LQR.

These results will be further discussed in the last chapter, where a trade-off analysis is performed in order to draw the conclusions about the benefits and disadvantages provided by each controller.
For the sake of knowledge, the histograms related to vertical and horizontal touchdown velocities of the Monte Carlo analysis are reported, to allow the visualization of the values distribution.

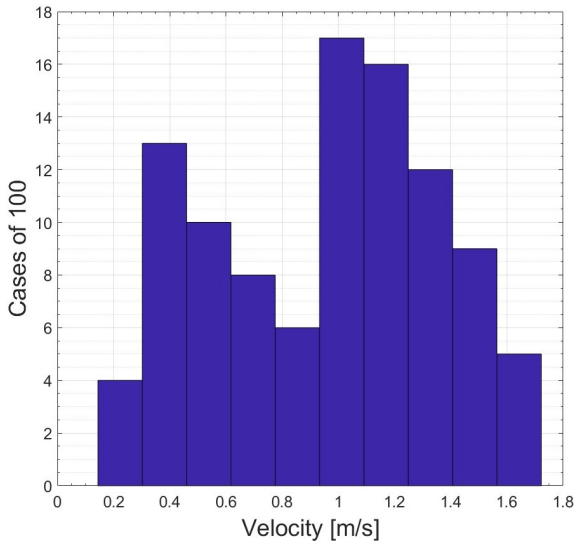


Figure 5.6: MPC with RDA, horizontal velocities at TD.

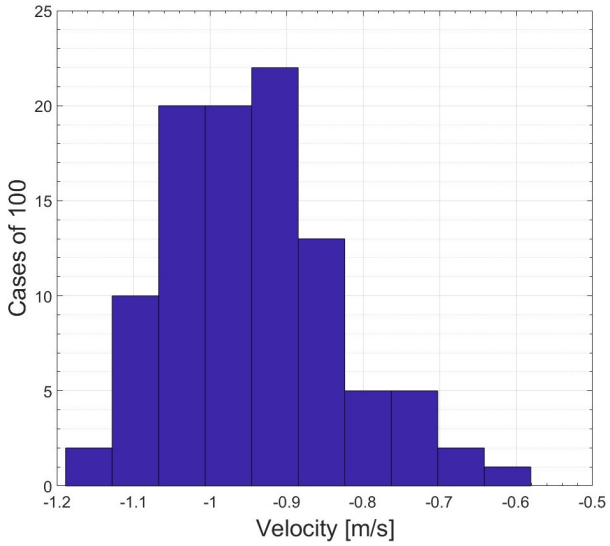


Figure 5.7: MPC with RDA, vertical velocities at TD.

## 5.6. ALO configuration discussion

With the only-altimeter configuration, the NMPC structure described above and used with the RDA do not provide completely satisfactory performances.
Different tuning strategies have been used to obtain successful results in terms of soft landing, distance from the target, and propellant consumption; however, none of them yielded a Monte Carlo analysis with 100 out of 100 successful cases in terms of landing velocities.

The tuning proposed for the RDA configuration provides a Monte Carlo simulation where around 1/3 of the cases are landing with horizontal velocity at touchdown higher than the required threshold at 99% probability 90% confidence level.
This leads to the implementation of a tuning which has to be less stringent in the horizontal distance requirement, in order to guarantee a safer landing with a lower discrepancy in the horizontal velocity performance a touchdown.
Keeping the prediction horizon equal to $p_H = 12$, the tuning that provides the best result is now presented. The translational LQR tuning has changed as follows, with respect to the one presented in Sect. 5.4.1. Again, the only differences with respect to the tuning originally chosen for the LQR presented in Sect. 4.2.4 are in the $\sigma$ related to the position in $x$ and $y$.

$\sigma_{p_x}$ = [140 140 130 120 120 120 90 90 80 80 80 80 80 80 80 80 80 80 80 80

```
      80 80 80];
σ_py = [140 140 130 120 120 120 90 90 80 80 80 80 80 80 80 80 80 80 80 80
      80 80 80];
```

The altitude breakpoints vector for the translational control is the same as before:

```
Z_bp = [0 5 10 15 20 30 40 50 70 90 110 150 200 300 400 600 900 1100 1300
        1800 2000 2400 3000];
```

The last 8-10 elements of the $\sigma_{p_x}$ and $\sigma_{p_y}$ are actually superfluous since the divert maneuver ends around at 200-300 meters of altitude; therefore, the LQR comes into play at that altitude and the breakpoints related to higher altitudes become not useful. However, they are still kept in the architecture of the simulation since it is not known exactly at what altitude the controller switch occurs.
The most important thing to note is that, in order to obtain a better result, the weights in $x$ and $y$ position (i.e. the precision in landing accuracy requested to the controller) have been significantly raised. In particular, almost 5 times higher for values closer to ground and 4 times higher for altitude values.
The weights of the rotational control, instead, are the same as chosen for the LQR control without RDA, in Sect. 4.3.4.
Eventually, also the $\mathbf{Q}$ matrix of the cost function of the NMPC controller sees a different tuning with respect to the one shown in Sect. 5.4 for the RDA configuration.

$$\mathbf{Q} = diag\left(\begin{bmatrix} 10 & 10 & 40 & 80 & 80 & 70 & 200 & 200 & 200 & 200 & 100 & 100 & 100 \end{bmatrix}\right) \quad (5.53)$$

The driving change in this matrix is the increase in the weights related to the horizontal velocities on $x$ and $y$, from 30 to 80. This also brings to a necessary increase in the rotational weights for quaternions and angular velocity, respectively from 130 to 200 and from 80 to 100. The weights in position, especially in $x$ and $y$ are instead kept equal as before, but now proportionally lower than the others, again in such a way to have a controller which prefers to handle the thrusters in order to lower the velocities instead of landing closer to the target.
Despite all these design choices, the Monte Carlo simulation still provides some cases with too high horizontal landing velocity. In particular, there are 6 cases with a horizontal velocity between 2.002 $m/s$ and 2.530 $m/s$. Therefore, the 6% of the simulation is successful in everything except for the horizontal velocity, which exceeds the limit of about $0.5 m/s^2$. Although these velocities could be accepted, considering that they exceed the limit by a small margin, the main problem remains in the general performances of the simulation: landing accuracy and propellant consumption do not provide completely

satisfactory results. Even by considering only the 94 successful cases out of 100, in the next histogram it is possible to see how the landing accuracy has gotten worse than the one obtained with LQR and with PID, with the ALO configuration.

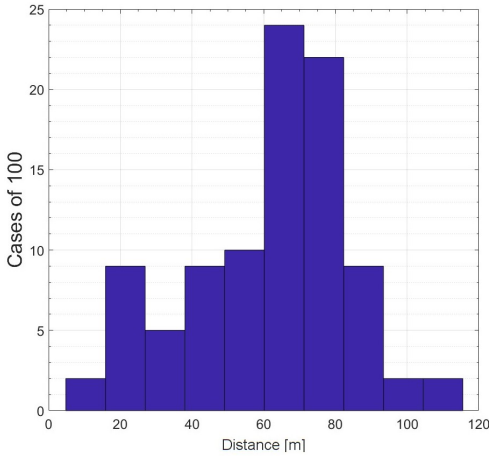


Figure 5.8: MPC with ALO, $p_H = 12$, horizontal distance magnitude at TD, MC simulation successful cases.

The propellant consumption as well, as shown in Fig. 5.9, is higher than with LQR and PID, with a maximum of around 296 kg of consumption.

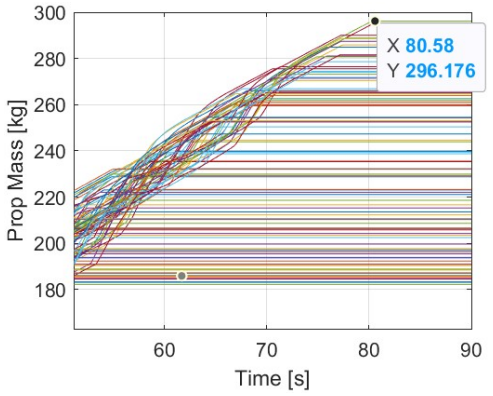


Figure 5.9: MPC with ALO, $p_H = 12$, propellant consumption.

One possible solution to this problem could be to reduce the prediction horizon length from $p_H = 12$ to $p_H = 5$. This choice is driven by the arguments already treated in Sect. 5.1.1, for which a decreasing horizon could be a solution to enhance MPC performance; therefore, this strategy is investigated (tuning and weights selection are not reported here, to avoid overloading the analysis).
However, the results provided do not yet satisfy all the requirements, presenting 4 cases

with a horizontal landing velocity exceeding the limit up to 2.567 $m/s$.
Again, the performances are generally not completely satisfactory, the same figures shown before are presented for this new case in Fig. 5.10 and Fig. 5.11, respectively the histogram of the landing distances for the successful cases and the propellant consumption.

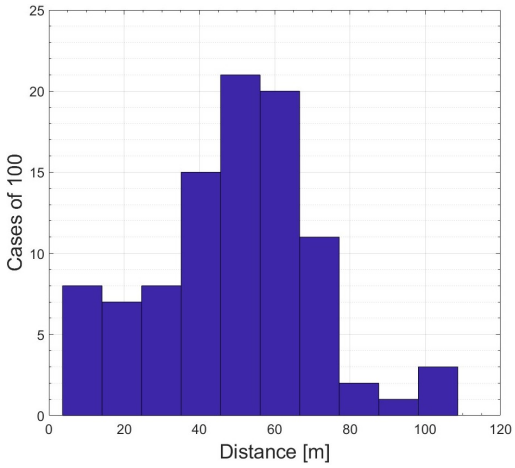


Figure 5.10: MPC with ALO, $p_H = 5$, TD distances successful cases.

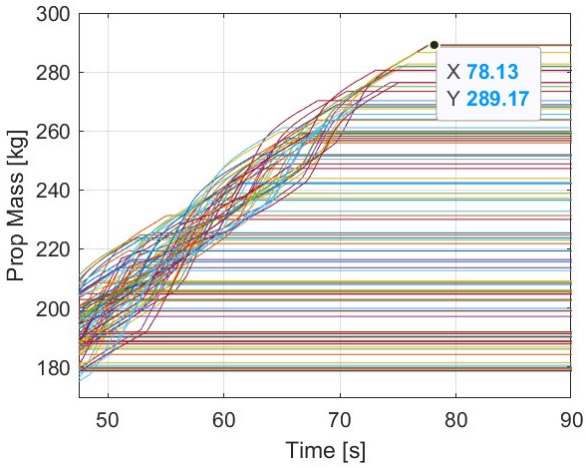


Figure 5.11: MPC with ALO, $p_H = 5$, propellant consumption.

As can be seen, the landing distances of successful cases are slightly decreased with respect to the previous case, as well as the general propellant consumption; however, the results are still not comparable with the ones obtained with the other controllers, above all because of the cases presenting too high horizontal velocities at tocuhdown.
The reason why this is happening is because, not having direct measurements on the velocities provided by the doppler, the performances of the MPC during the divert maneuver are degraded. This is understandable since the MPC have to perform a propagation forward in the future, and not having a precise estimation of the velocities does not allow an accurate predicted state, hence, the predicted control actions are not suitable in order to follow the reference profile.

To perceive this limitation of the MPC, a performance analysis of the horizontal velocities estimation is performed. In particular, simulation number 16 of the Monte Carlo analysis with $p_H = 12$ is subject to analysis, being part of realizations with horizontal velocity higher than the requirement, introduced before.
In Fig. 5.12 and Fig. 5.13 the trends of the estimation error for the horizontal velocity on $x$, both with and without the RDA, are shown; in order to perceive the behavior of both configurations. In Fig. 5.14 and Fig. 5.15 the same plots are presented for the $y$ axis.

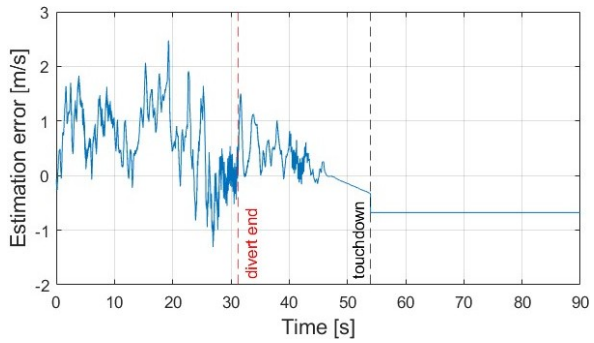


Figure 5.12: Velocity on $x$ estimation error, with RDA.

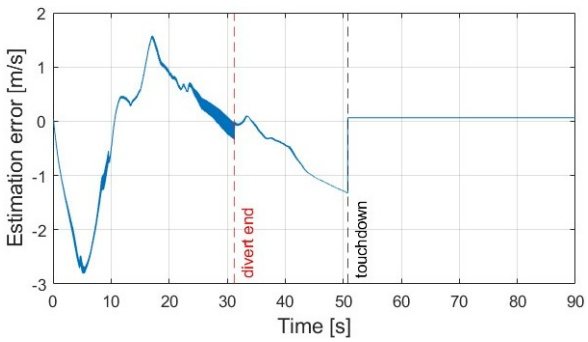


Figure 5.13: Velocity on $x$ estimation error, with ALO.

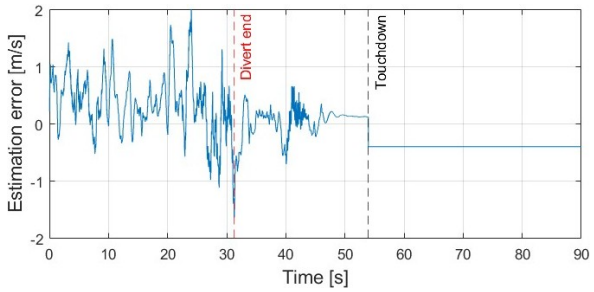


Figure 5.14: Velocity on $y$ estimation error, with RDA.

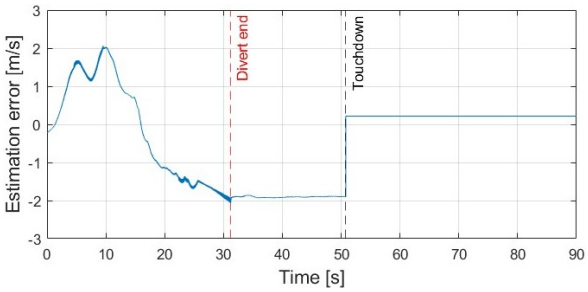


Figure 5.15: Velocity on $y$ estimation error, with ALO.

Analyzing these plots it is possible to understand that without the RDA, the velocities estimation is less accurate, and hence the controller struggles more to succeed in following the reference profiles.
During the MPC phase, as observable from Fig. 5.13 and Fig. 5.15, the error is almost always far from zero, and this leads to an inaccurate control of the spacecraft.

Figures from Fig. 5.16 to 5.19 present the trend of both real and estimated velocities for both axes in both configurations. It is possible to see that the error goes to zero different times during the MPC phase, therefore the controller gets the right estimation sporadically, which is sufficient to have a successful landing.
In particular, this behavior can be described as a systematic trend of the errors in the ALO configuration whereas. This is different from the RDA configuration behavior, where the error does not display any dominant systematic component and instead behaves in a noise-like manner, lacking any significant deterministic trend.

The tuning of such a complex system is a time-consuming process; due to the limited time available within the scope of this thesis, it was therefore not investigated further.

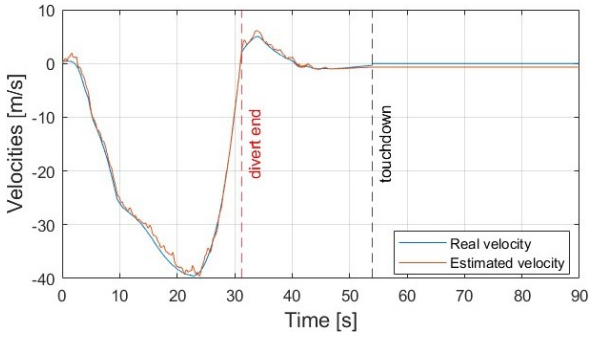


Figure 5.16: Real and estimated velocities on $x$, with RDA.

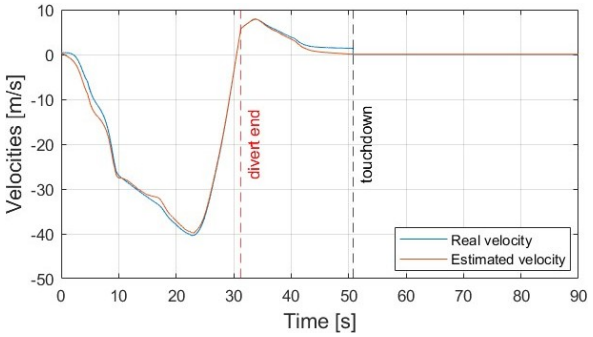


Figure 5.17: Real and estimated velocities on $x$, with ALO.

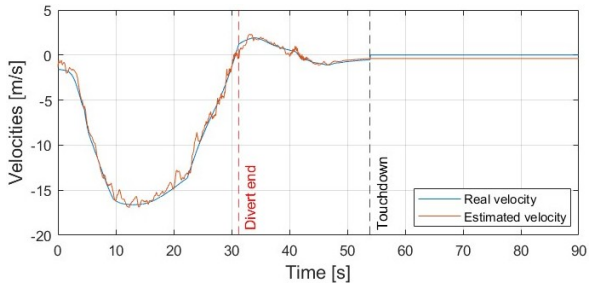


Figure 5.18: Real and estimated velocities on $y$, with RDA.

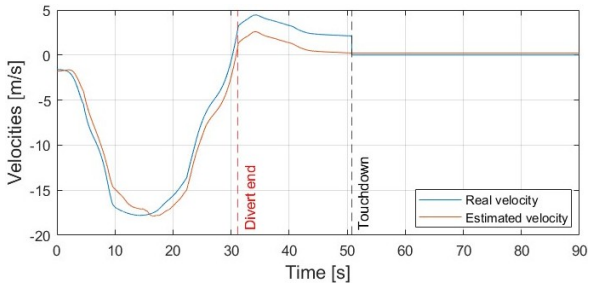


Figure 5.19: Real and estimated velocities on $y$, with ALO.

# 6 | Conclusions

In this work, the LQR and MPC controllers were analyzed to assess their suitability for a pinpoint landing on Mars. The controllers were tested by performing a Monte Carlo analysis on a population of 100 cases. Two different configurations have been analyzed for each control scenario, the first one with RDA, the second one without radar doppler, using only the altimeter (ALO).

In conclusion, the LQR manages to perform the pinpoint landing with good performances for both configurations. Moreover, the performance is improved with respect to that provided by the PID controller, which was used as a benchmark solution. In particular, LQR demonstrates the ability to land closer and consume less propellant than PID with both configurations.
The results with the LQR controller improved when a full integrated LQR controller was implemented in both translation and rotation, especially using altitude-varying gain to tune the control actions applied in function of the distance from ground.

The MPC was used only during the divert maneuver, after which the controller was switched to LQR for the final vertical descent. It succeeded in providing reliable and acceptable results with the three-sensor configuration, providing a landing distance from the target even better than that obtained with LQR, but significantly increasing propellant consumption. With the ALO configuration, the MPC struggled to obtain a comfortable solution, with cases of distance from the target and horizontal velocity at touchdown higher than the requirements.
The latter results are mainly due to the inherent looking-forward nature of the Model Predictive Control itself, that highlights a limit in the accurate command definition when the navigation solution is affected by larger uncertainties.

## 6.1. Future developments

The performance of the simulator could be further improved by integrating the proposed control solutions with enhanced navigation strategies for velocity estimation when the

Radar Doppler input is not available. Using more advanced navigation techniques would provide more accurate measurements, and therefore improve the final performances at touchdown for the altimeter only (ALO) configuration with Model Predictive Control.
Another improvement which would be beneficial is to modify the phase of vertical approach to the target at the conclusion of the divert maneuver. Right now, when the divert maneuver ends, the lander performs a simple vertical descent to the ground, since in that moment the lander is supposed to be at some hundreds of meters of altitude, vertically aligned above the target. In this last phase, a final terrain monitoring could be introduced, performing a trajectory planning and control devoted to hazard detection and avoidance for the last vertical descent. The last proposed future development is to investigate the integration of the proposed controller with a final approach to terrain based on Sky Crane technology, to allow for the extension toward large and heavier platforms.

# Bibliography

[1] Suzanne Deffree. Edn - mars 2 crash lands on the red planet, 2019. URL https://www.edn.com/mars-2-crash-lands-on-the-red-planet-november-27-1971/.

[2] The New York Times. Soviet says mars signal lasted 20 seconds, 1971. URL https://www.nytimes.com/1971/12/20/archives/soviet-says-mars-signal-lasted-20-seconds.html.

[3] NASA. Viking project - viking 1 and viking 2, . URL https://science.nasa.gov/mission/viking/.

[4] NASA. Mars pathfinder, . URL https://science.nasa.gov/mission/mars-pathfinder/.

[5] NASA. Mars exploration rovers: Spirit and opportunity, . URL https://science.nasa.gov/mission/mars-exploration-rovers-spirit-and-opportunity/.

[6] NASA JPL. Six things to know about nasa's opportunity mars rover, 2019. URL https://www.jpl.nasa.gov/news/six-things-to-know-about-nasas-opportunity-mars-rover/.

[7] NASA. Mars phoenix, . URL https://science.nasa.gov/mission/mars-phoenix/.

[8] NASA Astrobiology. Mars science laboratory. URL https://astrobiology.nasa.gov/missions/msl/.

[9] NASA. Here's how curiosity's sky crane changed the way nasa explores mars, 2024. URL https://www.nasa.gov/missions/mars-science-laboratory/curiosity-rover/heres-how-curiositys-sky-crane-changed-the-way-nasa-explores-mars/.

[10] David W. Way et al. Mars science laboratory: Entry, descent, and landing system performance, citation from page 5., March 2006. URL https://ntrs.nasa.gov/api/citations/20090007730/downloads/20090007730.pdf.

[11] NASA. Insight lander, . URL https://science.nasa.gov/mission/insight/.

[12] NASA. Mars 2020: Perseverance rover, . URL https://science.nasa.gov/mission/mars-2020-perseverance/.

[13] NASA. Ingenuity mars helicopter, . URL https://science.nasa.gov/mission/mars-2020-perseverance/ingenuity-mars-helicopter/.

[14] The Planetary Society. Tianwen-1 and zhurong, china's mars orbiter and rover. URL https://www.planetary.org/space-missions/tianwen-1.

[15] Xinli Li et al. Trim flap system design for improving ballistic-lifting entry performance of the tianwen-1 mars probe., May 2022. URL https://www.mdpi.com/2226-4310/9/6/287.

[16] NASA. Mars entry, descent, and landing challenges for human missions., 2024. URL https://www.nasa.gov/wp-content/uploads/2024/12/acr24-mars-edl-challenges.pdf?emrc=d99c3e.

[17] ESA. Exomars trace gas orbiter and schiaparelli mission, . URL https://exploration.esa.int/web/mars/-/46124-mission-overview.

[18] ESA. Mars sample return, . URL https://science.nasa.gov/mission/mars-sample-return/.

[19] Shuang Lin and Xiuqiang Jiang. Review and prospect of guidance and control for mars atmospheric entry, 2014. URL https://www.sciencedirect.com/science/article/pii/S0376042114000517.

[20] Thomas Antony. Deriving the msl/apollo entry guidance algorithm, 2021. URL https://www.thomasantony.com/posts/2021/msl-apollo-guidance/.

[21] Miguel San Martin et al. Mars science lander gnc design for entry, descent, and landing, 2011. URL https://pdfs.semanticscholar.org/36d9/add9ce34611efce59a5ee4f56d3f72b3e24b.pdf.

[22] Christopher D. Karlgaard et al. Mars insight entry, descent, and landing trajectory and atmosphere reconstruction. URL https://ntrs.nasa.gov/api/citations/20200002910/downloads/20200002910.pdf.

[23] David Way et al. Edl simulation results for the mars 2020 landing site safety assessment, 2020. URL https://ntrs.nasa.gov/api/citations/20200002978/downloads/20200002978.pdf.

[24] Xiangyu Huang et al. The tianwen-1 guidance, navigation, and control for mars entry, descent, and landing, 2021. URL https://pdfs.semanticscholar.org/c1e0/40c9afe7d059f6bd563eaac77e855ab4c1b1.pdf.

[25] Behçet Açıkmeşe et al. Flight testing of trajectories computed by g-fold: Fuel optimal large divert guidance algorithm for planetary landing, 2012. URL https://www.researchgate.net/publication/236334043_Flight_Testing_of_Trajectories_Computed_by_G-FOLD_Fuel_Optimal_Large_Divert_Guidance_Algorithm_for_Planetary_Landing.

[26] Behçet Açıkmeşe and Scott R. Ploen. Convex programming approach to powered descent guidance for mars landing., 2007. URL https://www.researchgate.net/profile/Adhithya-Babu/post/Lossless_convexification_vs_Successive_convexification/attachment/5f02dbe44597f90001fdb0ce/AS%3A910206900908032%401594021644439/download/_Convex+programming_pdg+for+Mars+landing.pdf.

[27] S. Joe Qin et al. A survey of industrial model predictive control technology, 2002. URL https://cepac.cheme.cmu.edu/pasilectures/darciodolak/Review_article_2.pdf.

[28] David Blum et al. Field demonstration and implementation analysis of model predictive control in an office hvac system, 2022. URL https://www.sciencedirect.com/science/article/pii/S0306261922004895.

[29] Chiara Toffanin et al. Artificial pancreas: model predictive control design from clinical experience, 2013. URL https://pubmed.ncbi.nlm.nih.gov/24351173/.

[30] Alberto Guiggiani et al. Constrained model predictive control of spacecraft attitude with reaction wheels desaturation, 2015. URL https://cse.lab.imtlucca.it/~bemporad/publications/papers/ecc15-reaction-wheels.pdf.

[31] Louis Breger, Jonathan How, and Arthur Richards. Model predictive control of spacecraft formations with sensing noise, 2005. URL https://skoge.folk.ntnu.no/prost/proceedings/acc05/PDFs/Papers/0422_ThB04_1.pdf.

[32] S. Di Cairano et al. Model predictive control approach for guidance of spacecraft rendezvous and proximity maneuvering, 2012. URL https://www.merl.com/publications/docs/TR2012-099.pdf.

[33] E. N. Hartley et al. Model predictive control system design and implementation

for spacecraft rendezvous, 2011. URL https://eprints.whiterose.ac.uk/id/eprint/90483/1/orcsatpaper_final.pdf.

[34] Talha Ulukir. Automatic landing with model predictive control, 2022. URL https://www.researchgate.net/publication/379832453_Automatic_Landing_with_Model_Predictive_Control.

[35] Carlo Alberto Pascucci et al. Model predictive control for powered descent guidance and control, 2015. URL https://cse.lab.imtlucca.it/~bemporad/publications/papers/ecc15-descent.pdf.

[36] Guillermo Zaragoza Prous, Enric Grustan-Gutierrez, and Leonard Felicett. A decreasing horizon model predictive control for landing reusable launch vehicles, citations from sec. 3.2, page 6., January 2025. URL https://doi.org/10.3390/aerospace12020111.

[37] Jacopo Guadagnini, Michèle Lavagna, and Paulo Rosa. Model predictive control for reusable space launcher guidance improvement, 2022. URL https://www.sciencedirect.com/science/article/pii/S0094576521005567.

[38] Ki-Wook Jung et al. Model predictive guidance for fuel-optimal landing of reusable launch vehicles, 2024. URL https://arxiv.org/abs/2405.01264.

[39] Xiaodong Yan and Shi Lyu. Mars entry guidance based on nonlinear model predictive control with disturbance observer, 2019. URL https://www.sciencedirect.com/science/article/pii/S0016003219306313.

[40] Wang Ting et al. Model predictive control guidance for constrained mars pinpoint landing, 2016. URL https://www.sciencedirect.com/science/article/pii/S0016003219306313.

[41] Jhonathan Murcia-Piñeros et al. Model predictive control for gliding descent on mars, 2025. URL https://www.mdpi.com/2076-3417/15/19/10400.

[42] NASA. Project mercury, . URL https://www.nasa.gov/project-mercury/.

[43] NASA. Project gemini, . URL https://www.nasa.gov/gemini/.

[44] NASA. Landing the space shuttle orbiter, 2006. URL https://www3.nasa.gov/centers/kennedy/pdf/167415main_LandingatKSC06.pdf.

[45] Lars Blackmore SpaceX. Frontiers of engineering: Reports on leading-edge engineering from the 2016 symposium. chapter 10: Autonomous precision landing of space

rockets, 2017. URL https://www.nationalacademies.org/read/23659/chapter/10.

[46] SpaceX. Starship - service to earth orbit, moon, mars and beyond. URL https://www.spacex.com/vehicles/starship.

[47] Feng Lin. Robust control design – an optimal control approach, citations from pages 8–9, 101–102, 112–113, Publicated in 2007.

[48] Earl Bryson and Yu-Chi Ho. Applied optimal control: Optimization, estimation and control, 1975.

[49] Vincenzo Pesce, Andrea Colagrossi, and Stefano Silvestrini. In *Modern Spacecraft Guidance, Navigation, and Control – From System Modeling to AI and Innovative Applications*, chapter 10, pages 609–612. 2023. Chapter 10 edited by Francesco Cavenago, Aureliano Rivolta, Emanuele Paolini, Francesco Sanfedino, Andrea Colagrossi, Stefano Silvestrini and Vincenzo Pesce.

# List of Figures

# List of Tables

# *Ringraziamenti*

*In primo luogo, ci tengo a ringraziare Paolo, il mio supervisore in azienda, che mi ha seguito con cadenza praticamente giornaliera in questi mesi e senza il quale questo lavoro non sarebbe stato possibile. Grazie per la tua disponibilità e la tua pazienza.*

*Un grazie anche al mio relatore, il professor Lunghi, per il supporto datomi in ogni fase del progetto.*

*Grazie a tutti coloro che mi hanno accompagnato in questi anni di studi, da Torino a Milano, passando per Madrid. E' tutto più leggero avendovi a fianco.*

*Sono grato a tutta la mia famiglia, e in particolare ai miei nonni, chi c'è ancora e chi non c'è più, da voi ho ricevuto e ricevo ancora l'affetto più puro e genuino che si possa ricevere.*

*Infine, il più grande grazie va ai miei genitori. Mamma, Papà, mi avete sempre sostenuto, mi avete sempre permesso tutto, dimostrandomi un amore incondizionato. Devo tutto al vostro supporto, al coraggio che mi date e alla stima che riponete in me ogni giorno. E' grazie a voi se posso ritenermi una persona fortunata, vi dedico ogni mio traguardo.*

*Continuerò a perseguire la mia estenuante ricerca del bello.*