Guidance and Control for Launch and Vertical Descend of Reusable Launchers using Model Predictive Control and Convex Optimisation

Guillermo Zaragoza Prous

Space Engineering, master's level (120 credits)
2020

Luleå University of Technology
Department of Computer Science, Electrical and Space Engineering
Abstract

The increasing market of small and affordable space systems requires fast and reliable launch capabilities to cover the current and future demand. This project aims to study and implement guidance and control schemes for vertical ascent and descent phases of a reusable launcher. Specifically, the thesis focuses on developing and applying Model Predictive Control (MPC) and optimisation techniques to several kino-dynamic models of rockets. Moreover, the classical MPC method has been modified to include a decreasing factor for the horizon used, enhancing the performance of the guidance and control. Multiple scenarios of vertical launch, landing and full flight guidance on Earth and Mars, along with Monte Carlo analysis, were carried out to demonstrate the robustness of the algorithm against different initial conditions. The results were promising and invite to further research.

Keywords

Model Predictive Control; Rocket; Reusable Launch Vehicle; Vertical Takeoff; Vertical Landing; Guidance and Control; Convex Optimisation; Microcontroller; Processor-in-the-Loop; Space Engineering.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>v</td>
</tr>
<tr>
<td>Contents</td>
<td>vii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xi</td>
</tr>
<tr>
<td>List of Tables</td>
<td>xiii</td>
</tr>
<tr>
<td>List of Abbreviations</td>
<td>xv</td>
</tr>
<tr>
<td>Nomenclature</td>
<td>xvii</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>xxi</td>
</tr>
<tr>
<td>1 Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Background</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Fundamental Questions</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Outline</td>
<td>3</td>
</tr>
<tr>
<td>2 Literature Review</td>
<td>5</td>
</tr>
<tr>
<td>2.1 Guidance and Control</td>
<td>5</td>
</tr>
<tr>
<td>2.1.1 Guidance Subsystem</td>
<td>6</td>
</tr>
<tr>
<td>2.1.1.1 Launch</td>
<td>7</td>
</tr>
<tr>
<td>2.1.1.2 Landing</td>
<td>8</td>
</tr>
<tr>
<td>2.1.1.3 Coastal Phase</td>
<td>9</td>
</tr>
<tr>
<td>2.1.2 Model Predictive Control</td>
<td>10</td>
</tr>
<tr>
<td>2.1.2.1 Definition</td>
<td>10</td>
</tr>
<tr>
<td>2.1.3 Minimum Principle</td>
<td>12</td>
</tr>
<tr>
<td>2.1.4 Convex Optimization</td>
<td>13</td>
</tr>
<tr>
<td>2.1.4.1 Definition</td>
<td>13</td>
</tr>
<tr>
<td>2.1.4.2 Convex Sets and Convex Functions</td>
<td>15</td>
</tr>
<tr>
<td>2.1.5 State of the Art Research</td>
<td>16</td>
</tr>
<tr>
<td>2.1.5.1 Powered Descend Guidance</td>
<td>16</td>
</tr>
<tr>
<td>2.1.5.2 Vertical Launch</td>
<td>18</td>
</tr>
<tr>
<td>2.2 Launcher Characteristics</td>
<td>19</td>
</tr>
<tr>
<td>2.3 Software</td>
<td>20</td>
</tr>
<tr>
<td>2.3.1 YALMIP and SeDuMi</td>
<td>20</td>
</tr>
</tbody>
</table>
## 2.3.2 Arduino IDE ............................................. 21
## 2.3.3 ChibiOS ................................................. 22
## 2.3.4 Gitlab .................................................... 22
## 2.4 Hardware ................................................... 24

### 3 Theory .................................................. 27

#### 3.1 Equations of Motion and Constraints .............................. 27

##### 3.1.1 Reference Frame ...................................... 28
##### 3.1.2 Weight Force .......................................... 29
##### 3.1.3 Aerodynamic Forces .................................... 30
##### 3.1.4 General Equation of Motion ............................. 33
##### 3.1.5 Point Mass 1-dimensional Model ......................... 34
##### 3.1.6 Point Mass 2-dimensional Model with Thrust Magnitude and Angle 35
##### 3.1.7 Point Mass 2-dimensional Model with Thrust Components ... 36
##### 3.1.8 Rigid Body 2-dimensional Model ......................... 37

#### 3.2 Guidance .................................................. 38

##### 3.2.1 Definition of Launch Problem ........................... 39
##### 3.2.2 Definition of Landing Problem .......................... 41
##### 3.2.3 Common Constraints .................................... 43

#### 3.3 Definition of Trajectory Optimisation ............................. 45

##### 3.3.1 Minimum Principle ...................................... 45

#### 3.3.1.1 Minimum Principle in 1 Dimension ....................... 45
##### 3.3.1.2 Minimum Principle in 2 Dimensions ................... 48
##### 3.3.2 Convex Optimisation ...................................... 50
##### 3.3.3 Discretisation of Equations of Motion .................... 54

#### 3.4 Model Predictive Control Algorithm .............................. 55

##### 3.4.1 Receding Horizon MPC .................................. 55
##### 3.4.2 Decreasing Horizon MPC ................................ 55

### 4 Implementation .............................................. 59

#### 4.1 MATLAB ................................................... 59

##### 4.1.1 Model Predictive Control with Minimum Principle .......... 60
##### 4.1.2 Model Predictive Control with Convex Optimisation ..... 61

#### 4.1.2.1 Main Scripts ........................................... 63
##### 4.1.2.2 State Propagator ...................................... 64
##### 4.1.2.3 Implementation Issues ................................. 67

##### 4.1.3 Code Versions ........................................... 68

#### 4.2 Real Time Simulation Setup .................................. 69

##### 4.2.1 Hardware ................................................ 69
##### 4.2.2 Software ................................................ 71

#### 4.3 Repository ................................................. 73

### 5 Simulations and Results ........................................ 75

#### 5.1 MPC with Minimum Principle .................................. 76

##### 5.1.1 1 Dimension Analytical Approach ........................ 76
##### 5.1.2 2 Dimension Numerical Approach ........................ 77

#### 5.2 MPC with Receding Horizon and Convex Optimiser ............... 79
### CONTENTS

5.3  MPC with Decreasing Horizon and Convex Optimiser ............................................. 80

5.3.1  Case A: Launch of Falcon 9 1st stage on Earth .................................................. 80

5.3.2  Case B: Landing of Falcon 9 1st stage on Earth .................................................... 82

5.3.3  Case C: Landing of Falcon 9 1st stage on Mars ..................................................... 84

5.3.4  Case D: Launch and Recovery of Falcon 9 1st stage on Earth .............................. 87

5.3.5  Case E: Launch and Recovery of Dummy Rocket on Mars .................................... 89

5.3.6  Case MC1: Monte Carlo Test for Landing the Falcon 9 1st stage on Earth .............. 91

5.3.7  Case MC2: Monte Carlo Test for Landing the Dummy Rocket on Mars .................. 94

5.3.8  Effect of the Time Percentage Update Factor ........................................................... 96

6  Conclusions and Further Work ................................................................................. 109

6.1  Further Work .............................................................................................................. 110

A  MATLAB README.md File ......................................................................................... 113

A.1  MPC Controller for Guidance and Control of a Reusable Vehicle ......................... 113

A.1.1  Version History ........................................................................................................ 113

A.1.1.1  Version 0.1 .......................................................................................................... 113

A.1.1.2  Version 0.1.1 ...................................................................................................... 114

A.1.1.3  Version 0.1.2 ...................................................................................................... 114

A.1.1.4  Version 0.1.3 ...................................................................................................... 114

A.1.2  SW Dependencies ................................................................................................... 115

A.1.3  Authors ..................................................................................................................... 115

B  Teensy Pinout ............................................................................................................... 117

References ....................................................................................................................... 119
List of Figures

1.1 Outline of the project ................................. 4
2.1 Phases of the ascent trajectory optimal control problem applying a gravity
turn manoeuvre. Image from [2] ........................... 8
2.2 Types of recovery and descent guidance problems. Image from [25] ..... 9
2.3 Falcon 9 1st stage separation. Initial part of the coastal phase. Screenshot
taken from live stream of a recent Falcon 9 launch ....................... 10
2.4 Block diagram of a model predictive controller .......................... 11
2.5 Functions that ........................................... 14
2.6 Landing ellipses for successful Mars landings, shown on elevation map of
Gale Crater. Image taken from [4]. Image credit: Ryan Anderson, USGS 16
2.7 User interface of the Arduino IDE ................................ 21
2.8 Git-flow branching model. Image taken from [10] ......................... 23
3.1 Representation of the launcher movement during the flight with respect to
the global reference frame of a planar Earth ............................... 29
3.2 Gravity force on the launcher .................................. 30
3.3 Representation of the angle-of-attack of the vehicle ........................ 31
3.4 Earth’s atmosphere density for the first 100 km of altitude according to
the exponential model .............................................. 32
3.5 Aerodynamic coefficients for the first stage of the VEGA launcher. Values
considered for the landing phase. Data collected from [25] ............... 33
3.6 Position of the thrusters in the Rigid Body model ......................... 38
3.7 Receding MPC algorithm ....................................... 56
4.1 UML diagram of the MATLAB Implementation of the 2 dimensional
landing case. The parameters in the config file with an asterisk refer to
the extra parameters for the comparative case .............................. 62
4.2 Hardware setup for the embedded simulation ............................ 70
4.3 Software design for the real time simulation setup ........................ 72
4.4 CranfieldRocketGNC group with the repositories ........................ 73
5.1 Position, velocities and thrust over time for the Minimum Principle prob-
lem in 1D .................................................................. 78
5.2 Launch path followed by the Falcon 9 1st stage in case A with the thrust
and velocity vectors ................................................. 82
5.3 Landing path followed by the Falcon 9 1st stage in case B with the thrust and velocity vectors. One execution of the convex optimiser. .................... 84
5.4 Landing path followed by the Falcon 9 1st stage in case B with the thrust and velocity vectors. MPC simulation. ................................. 85
5.5 Landing path followed by the Falcon 9 1st stage in case C with the thrust and velocity vectors. One execution of the convex optimiser. ............. 86
5.6 Landing path followed by the Falcon 9 1st stage in case C with the thrust and velocity vectors. MPC simulation. ................................. 87
5.7 Flight path followed by the Falcon 9 1st stage in case D with the thrust and velocity vectors. .................................................... 89
5.8 Flight path followed by the Dummy Rocket in case E with the thrust and velocity vectors. .................................................... 91
5.9 Initial positions and velocities for the Falcon 9 1st stage in the Monte Carlo Test MC1. Blue indicates that the problem is feasible, red that is unfeasible. .................................................... 93
5.10 Initial positions and velocities for the Dummy Rocket in the Monte Carlo Test MC2. Blue indicates that the problem is feasible, red that is unfeasible. .................................................... 95
5.11 Flight position for different time percentage factors. ....................... 96
5.12 Position, velocities and thrust over time for the Minimum Principle problem in 2D. .................................................... 98
5.13 Position, velocities and thrust plots for the MPC with receding horizon. Green plots represent the state of the rocket over time, while the red graphs are the last output of the optimal convex controller. .................... 99
5.14 Position, velocities and thrust plots for the Case A: launch of the Falcon 9 1st stage at Earth. .................................................... 100
5.15 Position, velocities and thrust plots for the Case B: landing of the Falcon 9 1st stage at Earth. One execution of the convex optimiser. ............. 101
5.16 Position, velocities and thrust plots for the Case B: landing of the Falcon 9 1st stage at Earth. MPC simulation. ..................................... 102
5.17 Position, velocities and thrust plots for the Case C: landing of the Falcon 9 1st stage at Mars. One execution of the convex optimiser. ............. 103
5.18 Position, velocities and thrust plots for the Case C: landing of the Falcon 9 1st stage at Mars. MPC simulation. ..................................... 104
5.19 Position, velocities and thrust plots for the Case D: launch and recovery of the Falcon 9 1st stage at Earth. ..................................... 105
5.20 Position, velocities and thrust plots for the Case E: launch and recovery of the Dummy Rocket at Mars. ..................................... 106
5.21 Paths in the Monte Carlo Test MC1 followed by the Falcon 9 1st stage at Earth. Blue paths are feasible problems and red mark the unfeasible ones. ..................................... 107
5.22 Paths in the Monte Carlo Test MC2 followed by the Dummy Rocket at Mars. Blue paths are feasible problems and red mark the unfeasible ones. ..................................... 108

B.1 Teensy 4.1 pinout [26] .......................................................... 117
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Comparative of present launchers.</td>
<td>19</td>
</tr>
<tr>
<td>3.1</td>
<td>Constraints for the Launch and Landing problems for the models using the MPC with Convex Optimisation.</td>
<td>57</td>
</tr>
<tr>
<td>4.1</td>
<td>Drag coefficients used in the simulation. The row index defines the relative velocity of the vehicle with respect to the atmosphere. It is represented in Mach. The column index defines the angle of attack (AOA) of the vehicle. It is represented in radians. This table is based on figure 3.5.</td>
<td>67</td>
</tr>
<tr>
<td>4.2</td>
<td>Lift coefficients used in the simulation. The row index defines the relative velocity of the vehicle with respect to the atmosphere. It is represented in Mach. The column index defines the angle of attack (AOA) of the vehicle. It is represented in radians. This table is based on figure 3.5.</td>
<td>67</td>
</tr>
<tr>
<td>4.3</td>
<td>MATLAB code versions and new features.</td>
<td>68</td>
</tr>
<tr>
<td>4.4</td>
<td>Some of the most important Teensy 4.1 technical specifications [26].</td>
<td>70</td>
</tr>
<tr>
<td>5.1</td>
<td>Rocket parameters used in the simulations.</td>
<td>75</td>
</tr>
<tr>
<td>5.2</td>
<td>Environment parameters used in the simulations. Data from [34] and [35].</td>
<td>76</td>
</tr>
<tr>
<td>5.3</td>
<td>Constant values for the analytical and numerical Minimum Principle problems.</td>
<td>77</td>
</tr>
<tr>
<td>5.4</td>
<td>Initial state values for the Minimum Principle problem in 1 Dimension.</td>
<td>77</td>
</tr>
<tr>
<td>5.5</td>
<td>Initial state values for the Minimum Principle problem in 2 Dimensions.</td>
<td>78</td>
</tr>
<tr>
<td>5.6</td>
<td>Configuration values for Cases A to E.</td>
<td>80</td>
</tr>
<tr>
<td>5.7</td>
<td>Initial, final and goal state values for the Case A simulation: Launch of a Falcon 9 1st stage on Earth.</td>
<td>81</td>
</tr>
<tr>
<td>5.8</td>
<td>Initial, final and goal state values for the Case B simulation: Landing of a Falcon 9 1st stage on Earth.</td>
<td>83</td>
</tr>
<tr>
<td>5.9</td>
<td>Initial, final and goal state values for the Case C simulation: Landing of the Falcon 9 on Mars.</td>
<td>84</td>
</tr>
<tr>
<td>5.10</td>
<td>Initial, final and goal state values for different phases of the Case D simulation: Launch and Recovery of Falcon 9 1st stage on Earth.</td>
<td>88</td>
</tr>
<tr>
<td>5.11</td>
<td>Initial, final and goal state values for different phases of the Case E simulation: Launch and Recovery of Dummy Rocket on Mars.</td>
<td>90</td>
</tr>
<tr>
<td>5.12</td>
<td>Configuration values for the Monte Carlo MC1 simulation: Landing of the Falcon 9 1st on Earth.</td>
<td>92</td>
</tr>
<tr>
<td>5.13</td>
<td>Statistical data for the feasible solutions in the Monte Carlo MC1 simulation: Landing of the Falcon 9 1st on Earth.</td>
<td>93</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>5.14</td>
<td>Configuration values for the Monte Carlo MC2 simulation: Landing of the Dummy Rocket on Mars</td>
<td>94</td>
</tr>
<tr>
<td>5.15</td>
<td>Statistical data for the feasible solutions in the Monte Carlo MC2 simulation: Landing of the Dummy Rocket on Mars</td>
<td>95</td>
</tr>
<tr>
<td>5.16</td>
<td>Final values for the state vector, the mass and the execution time for the different MPC TPUF values</td>
<td>97</td>
</tr>
</tbody>
</table>
# List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>DRL</td>
<td>Downrange Landing</td>
</tr>
<tr>
<td>EOM</td>
<td>Equations Of Motion</td>
</tr>
<tr>
<td>GNC</td>
<td>Guidance, Navigation and Control</td>
</tr>
<tr>
<td>GND</td>
<td>Ground</td>
</tr>
<tr>
<td>GPIO</td>
<td>General Purpose Input Output</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>IPM</td>
<td>Interior-Point Method</td>
</tr>
<tr>
<td>MCU</td>
<td>Microcontroller Unit</td>
</tr>
<tr>
<td>MIPS</td>
<td>Millions of Instructions per Second</td>
</tr>
<tr>
<td>MPC</td>
<td>Model Predictive Control</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>RLV</td>
<td>Reusable Launch Vehicle</td>
</tr>
<tr>
<td>RT</td>
<td>Real Time</td>
</tr>
<tr>
<td>RTLS</td>
<td>Return To Launch Site</td>
</tr>
<tr>
<td>RTOS</td>
<td>Real-Time Operating System</td>
</tr>
<tr>
<td>SATM</td>
<td>School of Aerospace, Technology and Manufacturing</td>
</tr>
<tr>
<td>SCP</td>
<td>Sequential Convex Programming</td>
</tr>
<tr>
<td>SDP</td>
<td>Semidefinite Programming</td>
</tr>
<tr>
<td>SOCP</td>
<td>Second Order Cone Programming</td>
</tr>
<tr>
<td>TBVP</td>
<td>Two Boundary Value Problem</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>TPUF</td>
<td>Time Percentage Update Factor</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modelling Language</td>
</tr>
<tr>
<td>VCS</td>
<td>Version Control System</td>
</tr>
<tr>
<td>VTVL</td>
<td>Vertical Takeoff/Vertical Landing</td>
</tr>
<tr>
<td>ZLGT</td>
<td>Zero-Lift Gravity Turn</td>
</tr>
</tbody>
</table>
Nomenclature

**Model Variables**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\vec{s}$</td>
<td>State vector of a vehicle</td>
<td></td>
</tr>
<tr>
<td>$x$</td>
<td>Horizontal position of a vehicle</td>
<td>m</td>
</tr>
<tr>
<td>$z$</td>
<td>Vertical position of a vehicle</td>
<td>m</td>
</tr>
<tr>
<td>$\theta$</td>
<td>Orientation of a vehicle</td>
<td>rad</td>
</tr>
<tr>
<td>$V_x$</td>
<td>Horizontal velocity of a vehicle</td>
<td>m/s</td>
</tr>
<tr>
<td>$V_z$</td>
<td>Vertical velocity of a vehicle</td>
<td>m/s</td>
</tr>
<tr>
<td>$\omega$</td>
<td>Angular velocity of a vehicle</td>
<td>rad/s</td>
</tr>
<tr>
<td>$m$</td>
<td>Mass of a vehicle</td>
<td>kg</td>
</tr>
<tr>
<td>$I_{sp}$</td>
<td>Specific Impulse of an engine</td>
<td>s</td>
</tr>
<tr>
<td>$R_L$</td>
<td>Radius of the vehicle</td>
<td>m</td>
</tr>
<tr>
<td>$L_L$</td>
<td>Length of the vehicle</td>
<td>m</td>
</tr>
<tr>
<td>$F$</td>
<td>Generic force</td>
<td>N</td>
</tr>
<tr>
<td>$W$</td>
<td>Weight of a vehicle</td>
<td>N</td>
</tr>
<tr>
<td>$M$</td>
<td>Torque applied to a vehicle</td>
<td>Nm</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
<td>Unit</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>I</td>
<td>Moment of Inertia of a vehicle</td>
<td>N</td>
</tr>
<tr>
<td>$c_g$</td>
<td>Centre of gravity to a vehicle</td>
<td></td>
</tr>
<tr>
<td>$\gamma$</td>
<td>Climb angle of a vehicle</td>
<td>rad</td>
</tr>
</tbody>
</table>

**Aerodynamic Variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D$</td>
<td>Drag force applied to a vehicle</td>
<td>N</td>
</tr>
<tr>
<td>$L$</td>
<td>Lift force applied to a vehicle</td>
<td>N</td>
</tr>
<tr>
<td>$c_p$</td>
<td>Centre of pressure of a vehicle</td>
<td></td>
</tr>
<tr>
<td>$C_D$</td>
<td>Drag coefficient of a vehicle</td>
<td></td>
</tr>
<tr>
<td>$C_L$</td>
<td>Lift coefficient of a vehicle</td>
<td></td>
</tr>
<tr>
<td>$S_T$</td>
<td>Total surface exposed to the advance of a vehicle</td>
<td>$m^2$</td>
</tr>
<tr>
<td>$S_C$</td>
<td>Circular surface exposed to the advance of a vehicle</td>
<td>$m^2$</td>
</tr>
<tr>
<td>$S_L$</td>
<td>Lateral surface exposed to the advance of a vehicle</td>
<td>$m^2$</td>
</tr>
<tr>
<td>$\varepsilon$</td>
<td>Angle of attack of a vehicle</td>
<td>rad</td>
</tr>
</tbody>
</table>

**Control Variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\vec{c}$</td>
<td>Control vector applied to a vehicle</td>
<td></td>
</tr>
<tr>
<td>$T$</td>
<td>Thrust magnitude of an engine</td>
<td>N</td>
</tr>
<tr>
<td>$T_x$</td>
<td>Horizontal component of the thrust</td>
<td>N</td>
</tr>
<tr>
<td>$T_z$</td>
<td>Vertical component of the thrust</td>
<td>N</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>Angle of the thrust</td>
<td>rad</td>
</tr>
<tr>
<td>$T_L$</td>
<td>Thrust magnitude of a lateral engine placed on the left of a vehicle</td>
<td>N</td>
</tr>
</tbody>
</table>
**NOMENCLATURE**

\( T_R \)  Thrust magnitude of a lateral engine placed on the right of a vehicle  \( N \)

**Optimisation Variables**

\( J \)  Objective index of an optimisation problem

\( H \)  Hamiltonian of an optimisation problem

\( \lambda \)  Lagrangian multiplier or costate

\( \Gamma \)  Relaxation slack variable for the thrust magnitude  \( N \)

\( \sigma \)  Convexification variable  \( m/s^2 \)

\( u \)  Convexification variable  \( m/s^2 \)

\( \Omega \)  Constraint limit on the rate of change of the thrust direction

\( \Upsilon \)  Constraint limit on the angle of attack

\( \eta \)  Convexification variable for the mass. Logarithm of the mass

**Physics Variables and Constants**

\( \rho_{\text{air}} \)  Density of the air  \( kg/m^3 \)

\( g \)  Planetary gravity acceleration  \( m/s^2 \)

\( g_0 \)  Planetary gravity acceleration at sea-level (Earth)  \( 9.81 m/s^2 \)

\( H \)  Scale height  \( m \)
Acknowledgements

After two years of polar adventures, unimaginable cold temperatures and rocket science, every trip has its own end. It would have not been possible to write this today without the support of my family and friends. For all those moments in Kiruna, for the Grönstensvägen ultras and the polar India, and for the reindeers!

Likewise, I would like to thank Dr. Leonard Felicetti for all his support and advice, which has been really helpful throughout the thesis, even more in the current circumstances and the remote teaching.

Finally, and most importantly, to the person that supports me in every crazy decision I make, and that pushes me to give the most out of me. I do not know what I would do without you, Vicky!

To the stars and back!
Chapter 1

Motivation

1.1 Background

Since the early days, humans have dreamed of reaching the Moon, the outer space and abandoning the atmosphere. From ancient societies all over the globe to the modern age, the vicinity of our planet has been thoughtfully studied and analysed. However, it was not until the mid 20th century that the necessary technology was mature enough to make this dream come true. In 1957, the Soviet Union put into orbit the first man-made satellite, which reached space aboard a modified R-7 Semyorka, the world’s first intercontinental ballistic missile. During the following decades, the development of the space technology was held on national space agencies and governmental organisations. The U.R.S.S. sent the first man into space assisted by a Vostok and U.S. reached the Moon aboard the Saturn V, the biggest rocket built.

This space race is now followed by the investment and efforts that private companies, along with public agencies, are putting on the development of launchers, microsatellites and sophisticated technology that ultimately should bring humans back to the Moon and, afterwards, to planet Mars. To democratise and reduce the cost of accessing space, the reuse of launch vehicles is of high importance. One way to do so is to vertically land the rocket after launch, and that is the main goal of this study. Moreover, this has been
already proven to be doable by several companies, such as SpaceX and Blue Origin.

Aligned with these developments, the present Master Thesis falls into the Guidance and Control studies for reusable launchers, with the help of optimisation tools. Different approaches have been studied and tested, in order to acquire the necessary knowledge and understanding to implement a feasible solution. This thesis sits on top of the work of a previous MSc ASE student that studied advance control and optimisation techniques focused on vertical descent of reusable launchers [38].

1.2 Fundamental Questions

The fundamental question to address in this thesis is to prove that a Model Predictive Control (MPC), along with a convex optimiser, can be used in a real launcher to guide and control it over the different phases of the flight, including its safe recovery. With this in mind, there are several objectives that must be studied, analysed and tested to demonstrate that the main goal is feasible:

- **Study of different optimal control approaches**: what are the multiple tools that are available to optimise the control over the rocket? which are the most suitable for the problem at hand?

- **Analysis of Model Predictive Control and its adequacy for guiding a space launcher**: since the early 2000s, several authors have recommended and researched the application of MPC to the descent of planetary landers and space launchers. Can it be applied to the whole flight of the vehicle?

- **Creation of several dynamic models of rockets and control them through Model Predictive Control**: in order to analyse the guidance and control problem, a separate number of dynamic models have to be matured to represent the behaviour of the launcher. How do they relate with the MPC algorithm? Are they affected by the optimised control law?
1.3. OUTLINE

- **Comparative analysis of the developed MPC through several cases and Monte Carlo tests**: the developed guidance and control algorithm has to be tested in a reasonable amount of situations to ensure its robustness and delimit the boundaries of the state space for the feasible solutions.

- **Implementation of the Model Predictive Control in a microcontroller unit (MCU)**: this microcontroller should be similar to the ones used in rockets to verify its potential in a real scenario. Therefore, it should be validated on top of a real-time simulation. Can a MPC controller really perform adequately to ensure the space launcher delivers its payload into orbit and safely lands afterwards?

1.3 Outline

At the beginning of the thesis, the main goals were to understand how a rocket behaves when it is in the air, and how its trajectory can be optimised fuel-wise. For that, several optimisation tools where studied, such as the Minimum Principle and Convex Optimisation. They were lately used in a Model Predictive Control implementation in MATLAB, where the Launch and Landing problems were being tested for several rocket models. The software implementation was enhanced at the same time as the literature review was progressing in different aspects of the project, e.g. regarding rocket guidance, space hardware and real-time programming. Ultimately, these algorithms were going to be tested on a hardware setup, where a microcontroller unit (MCU) would execute the MPC algorithm and a rocket simulation would be running on another MCU. Nonetheless, due to time constraints and various obstacles during the project, there has been not enough time to prove them on a microcontroller. Even so, the design of the real time setup is introduced for future developments. A sketch of how this work changed over time can be seen in figure 1.1.

The present document is organised in 6 chapters and 2 appendices. In Chapter 1, the motivation and outline of the thesis are introduced, with special focus on the fundamental
questions to be answered. Then, Chapter 2 reviews the current state of the art of the several topics that the project addresses, from guidance and control to rocket models and space hardware. Chapter 3 covers the theory that is used in the project, and how it was developed and thought. Alongside, Chapter 4 reflects how that theory was implemented in software and hardware, which makes the outcome of the thesis valuable. Finally, in Chapter 5 the results are studied and commented, while in Chapter 6 the thesis ends up with the final conclusions and remarks. Appendix A contains the README file for the MATLAB implementation and Appendix B shows the pinout of the microcontroller selected.
Chapter 2

Literature Review

The Guidance and Control of launchers has been a rising topic of study since the early space era. From the initial designs at the beginning of the 20th Century until nowadays with autonomous landing, passing through the space race during the cold war, engineers have always study the behaviour of rockets to control their performance.

Space launchers are highly varying systems that evolve quickly. Therefore there is a strict time constraint to generate the control signals that make the system stable and directs it to the mission goal. In this chapter, a thorough analysis of the current literature will be presented to demonstrate how this severe constraint can be overcome. For that, it starts introducing the Guidance and Control problem, including how several mathematical tools and algorithms can benefit the performance of the guidance. Then, there is a review of the rockets currently available at the market, taking into account their characteristics and performance. Finally, the required software for the implementation of the thesis, along with the necessary hardware for the real tests, are commented and explained.

2.1 Guidance and Control

The objective of the problem of Guidance and Control of space launchers is to maximise the payload into orbit subject to the equations of motion for a rocket and other side constraints, such as inequality constraints on the angle-of-attack and the dynamic pressure.
These are affected by the gravity and aerodynamic forces as a result of the effect of the atmosphere and the Earth [11].

In the case of vertical launch and descent vehicles, it is very important to consider the consumption of propellant and the quantity available, so the rocket does not run out of fuel before the landing goal has been reached. Moreover, as it is described in the book *Space Mission Engineering: The New SMAD* [33], the mass ratio between payload and rocket components is about $1/80 \text{ kg}$ for a chemical rocket, lifting 1 kg of payload for every 80 kg of rocket structure, engine, fuel, and propellant. A good performance of the Guidance system is of utterly importance to increase this ratio. For this purpose, several mathematical tools have been considered in the way to obtain a suitable solution for the problem of this thesis.

This section does a tour of the main theoretical tools that are going to be used. Starting with the explanation of what a guidance subsystem is and what is its purpose; going through the MPC theory and several optimisation mechanisms; and finishing with the current state of the art where researches around the world present novel techniques in Guidance and Control; this section covers all the basic topics to understand the rest of the project.

### 2.1.1 Guidance Subsystem

The guidance subsystem is one of the 4 main subsystems in a rocket, along with the structural, payload and propulsion subsystems. It has two main roles during the flight: ensure the stability of the vehicle and control it during manoeuvres. This subsystem typically consists of several navigation sensors to calculate the linear and angular position and velocity of the vehicle, along with one (or several) on board computers to perform such calculations. Then, it uses the available actuators, e.g. the main thrusters and fins, to correct any deviation in the trajectory and achieve its main goal. The guidance system is critical for the performance of the launcher and the success of delivering the payload safe into the desired orbit [3].
2.1. GUIDANCE AND CONTROL

Depending on the type of propulsion system, the guidance subsystem can control different aspects of the flight. For example, if the launcher is propelled by a solid thruster, as it could be the case for the Ariane Vega, the guidance subsystem can manage the orientation of the rocket, but the thrust magnitude will not depend on it. Solid boosters are meant to be ignited only once and their thrust can not be regulated. Nevertheless, liquid and hybrid thrusters allow for more control, as they can be turned on and off, and their thrust can be regulated. The latter case is the one that is going to be study in this thesis.

2.1.1.1 Launch

The launch phase is the first section of the flight. It covers from the launch pad until the cargo has been delivered safely and at a specific time. During the ascension the rocket body suffers critical stresses due to its bending and the aerodynamic forces. Accordingly, the performance of the control over the angle of attack is decisive to cross the denser layers of the atmosphere [11]. The angle of attack is the difference in orientation between the direction of the velocity of the launcher and its body orientation. Historically, the ascent guidance has been divided into the endo-atmospheric and the exo-atmospheric parts. The former is optimised off-line due to the complications in achieving a reliable real time guidance algorithm in the existence of aerodynamics forces. Conversely, the latter is calculated on board in a close loop system as the environmental disturbances are almost negligible [19].

The typical trajectory follows a curve that reorientates the vehicle from the vertical to the horizontal plane. This is usually meet performing a manoeuvre called gravity turn, which utilises the acceleration due to the gravitational force to rotate the launcher. It is similar to the spacecraft gravity gradient stabilisation in the sense that takes advantage of this force passively to perform a control action.

The gravity turn method consists of a series of steps from the launchpad until orbit, represented in Figure 2.1. Primarily, the rocket flights upwards in a direct vertical line.
After the vehicle passes the most dense part of the atmosphere, it carries out a pitch-over manoeuvre that slightly steers the launcher to reduce the transverse aerodynamic loads. This implies a negligible aerodynamic lift, in which is called a zero-lift gravity turn (ZLGT). The new orientation creates a misalignment between the rocket transverse axis and the gravity direction, which generates the necessary torque to adjust the attitude towards the horizon \[2\]. A thorough mathematical analysis of this type of trajectories can be found in \[7\] and \[30\].

![Figure 2.1: Phases of the ascent trajectory optimal control problem applying a gravity turn manoeuvre. Image from \[2\]](image)

### 2.1.1.2 Landing

The landing of space vehicles have been historically related to the return of astronauts to Earth and scientific missions to other planets. Notwithstanding, there is a recent trend to pursue the autonomous recovery of reusable launch vehicles (RLV), with SpaceX already achieving the reuse of its rockets. In this phase, the guidance and control subsystem aims to softly place the rocket on ground in a predefined spot. This is known as pinpoint landing.

In contrast with the launch phase, in this part of the flight the velocities experience are not as high, so the constraint in the angle of attack is not that strict, yet there are other requirements to fulfil. The addition of a landing phase implies that the payload capabilities of the RLV are decreased, or somehow the fuel must be even more optimised.
2.1. GUIDANCE AND CONTROL

Besides, the manoeuvre must be conducted completely autonomously and, thus, does not allow off-line optimisation.

Regarding the recovery of the RLV, there are 2 types of landing: following the inertia of the rocket, also know as downrange landing (DRL), or going back to the launch pad (RTLS), see Figure 2.2. Despite, both carry on the same launch trajectory in general. The DRL strategy positions the landing site close to its unpropelled impact location, optimising the necessary fuel. However, as the ascend phase is performed in an open-loop, there is an uncertainty on where that impact location is that grows throughout the flight and has to be compensated when the rocket is in the outer parts of the atmosphere. In the RTLS scenario, what changes in the RLS operations is that there is a second burn when the rocket is close to the apogee, to reduce the horizontal velocity and flip-over the vehicle. Then, the guidance and control subsystem performs similar actions to recovery the launcher at the launch site [25].

Figure 2.2: Types of recovery and descent guidance problems. Image from [25]

2.1.1.3 Coastal Phase

There is an intermediate phase in between the ascent and descent of a reusable launcher, which happens immediately after stage separation or in-orbit insertion of the rocket payload. This phase corresponds to the preparation of the launcher for the descent phase, performing a re-orientation of the vehicle towards the landing target, decreasing its ve-
locity direction and magnitude and deploying any necessary actuator, such as drag fins. These manoeuvres occur in high altitudes, such as the 80 km seen in Figure 2.3, where the density of the atmosphere is low.

Figure 2.3: Falcon 9 1st stage separation. Initial part of the coastal phase. Screenshot taken from live stream of a recent Falcon 9 launch.

### 2.1.2 Model Predictive Control

#### 2.1.2.1 Definition

Model Predictive Control (MPC) is an advanced technique of multivariable control that takes into account a predicted behaviour to operate a system in the near future. To anticipate this behaviour an internal model of the system is needed, see Figure 2.4. This simplified model is used by an optimisation solver that generates the necessary actions to reach an objective. Once those control actions are obtained, they are applied for a certain $\Delta t$, after which the solver produces a new set of control outputs and used again. This repetition contributes to minimise disturbances in our problem and the gap between the simplified model and the real plant, as it closes the control loop and updates the control action every step time $\Delta t$. [14]
2.1. GUIDANCE AND CONTROL

This method has been widely used in different industries, such as chemical and power plants, and proposed in the last decade as a possible solution for the vertical landing problem. The main ideas that appear in a Model Predictive Controller are [6]:

- **Explicit simplified model**: the controller takes advantage of a simplified model of the system to control, in this case a space launcher with no aerodynamic forces, to generate the control signals.

- **Minimisation of a cost function**: the controller must minimise the outcome of a predefined cost function that sets how well the controller performs. For the case that attains this thesis, the cost function will be related to the consumption of rocket propellant.

- **Comply with a set of constraints**: the control of the system must take into account that it cannot abandon a feasible working zone. An example could be that the rocket cannot be underground, or that the thrust applied must be between the limitations of the vehicle’s engine. The constraints can also consider the initial and desired final position and velocity of the launcher, which have to be withing the feasible area.

- **Receding horizon strategy**: the horizon time is one of variables to set before the execution of the MPC. It defines the time from present in which the control action is calculated. It is called receding horizon because the present time never reaches it,
but it is always in the future. This is something that will provoke some difficulties while landing a launcher, and it is covered in Chapter 3.

2.1.3 Minimum Principle

The Minimum Principle was defined by Pontryagin in 1956. It is utilised in optimal control theory to calculate optimal control signals based on a performance index $J$, and subject to initial $\vec{s}(t_0)$ and final conditions $\vec{s}(t_f)$. The Minimum Principle determines the necessary conditions for optimality, although they are not sufficient. In Equation 2.1, the objective index $J$ is composed of a function $\phi$ that depends on the final state vector $\vec{s}(t_f)$ and an integral of a Lagrangian function $L$ which depends on both the state vector and the control actions $\vec{c}(t_f)$ \[37\]. In the cases presented in Section 3.3, the component $\phi(\vec{s}(t_f), t_f)$ is considered to be 0, as the imposed final state vector for the launcher is $\vec{s}(t_f) = (0, 0, 0, 0)$, and $L(\vec{s}(t), \vec{c}(t), t)$ only depends on the control actions applied to the rocket.

$$J = \phi(\vec{s}(t_f), t_f) + \int_{t_0}^{t_f} L(\vec{s}(t), \vec{c}(t), t) dt \quad (2.1)$$

Introducing a new function, the Hamiltonian of the system, it is possible to reduce the optimisation problem to 1 dimension and, therefore, simplify it. The Hamiltonian, $H$, is defined as the sum of the Lagrangian function and the costate vector $\lambda$, that contains the costates\(^2\) of the system, Equation 2.2

$$H = L(\vec{s}(t), \vec{c}(t), t) + \lambda^T(t) f(\vec{s}(t), \vec{c}(t), t) \quad (2.2)$$

The necessary condition to find the optimal solution for a boundary value problem where the final value is constraint with an equality defines that the partial derivative of the

\(^{1}\)Also called Pontryagin’s Maximum Principle

\(^{2}\)Also know as Lagrange multipliers
2.1. GUIDANCE AND CONTROL

Hamiltonian with respect to the control actions has to be equal to 0 \[30\].

\[
\frac{\partial H}{\partial \vec{c}} = 0 \quad (2.3)
\]

The value over time of the Lagrange multipliers is obtained by using the costate equations, which are given by the equality in Equation \[2.4\]. According to the Minimum Principle, it is possible to find the optimal control using the necessary condition and the costate equations to compute the values of the system over time. Nonetheless, this problem can become analytically unsolvable in certain cases, and a numerical approach is needed, increasing the computational cost and not ensuring an optimal solution.

\[
\vec{\lambda}^T = -\frac{\partial H}{\partial \vec{s} t} \quad (2.4)
\]

In Section [3.3.1] this steps are put into practice with two simple rocket models, in order to land them at a certain ground spot, and with the aim to properly understand the principle through the derivation process.

2.1.4 Convex Optimization

Convex optimisation is a mathematical tool that uses polynomial time algorithms to solve optimisation problems. The main requisite to be able to use this method is that all sets and functions in the problem must be convex, including the objective. It ensures that every local optimal solution is a global optimal solution, as there is only one minimum in the functions given. Besides, if a convex optimisation algorithm does not find a feasible solution, it is proved that the problem is infeasible.

2.1.4.1 Definition

Expressed in a formal way, a convex optimisation problem is defined as the minimisation of an objective function \( f_0 \) subject to a set of constraints of the form \( f_i(x) \leq 0 \) where \( f_0, \ldots, f_m \) are convex functions and the equality constraint must be affine \[5\].
minimise \( f_0(x) \)

subject to \( f_i(x) \leq 0, \ i = 1, \ldots, m \)

subject to \( a_i^T x = b_i, \ i = 1, \ldots, p \)

where \( x = (x_1, x_2, \ldots, x_n) \) : optimisation variables

\( f_0 : \mathbb{R}^n \rightarrow \mathbb{R} \) : objective function

\( f_i : \mathbb{R}^n \rightarrow \mathbb{R}, i = 1, \ldots, m \) : constraint functions

Applying this concepts to the problem at hand, it could be said before moving on to the formal definition of guidance and control explained in the following chapters, that the objective function \( f_0 \) will be the fuel consumption of the rocket and the constraint functions \( f_i \) will be related to the state and control of the vehicle at the beginning, during the flight and at the end, with the limits of the thrust over time.
2.1.4.2 Convex Sets and Convex Functions

The most simple and clarifying way to know if a function is convex is by plotting it over its convex domain. In figure 2.5 there are several examples of functions that are and are not convex. Figures 2.5c and 2.5d are clearly convex for the convex set $-5 < x < 5$. The $\sin(x)$ seen in 2.5b is not convex at first sight, but it could be convexified if the domain is restricted to $-\pi < x < 0$. For figure 2.5a the case is a little bit more complex, although the outcome is the same. At first look, the function is obviously not convex. There are connections between points that cross the curve defined by the function, which is a sign that it is not convex. To convexify the function several steps should be considered. First, restrict the application domain to $x > 0$. This will prevent that any connection between points of the logarithm cross the function. Then, what we have is a concave function. To obtain the convex version of it, we can recall one of the properties of convex and concave functions: if $f(x)$ is concave for all the elements of a set, $-f(x)$ will be convex for the same set. In this case, the convexified version of the logarithm of $x$ is $-\log(x)$ for $x > 0$.

The function in figure 2.5d is a typical example of a convex function. Its U-shape is what defines a convex function. Optimisation problems around convex functions are fast to solve because there is no local minimum and through algorithms such as Interior-Point Method (IPM), it can be solved in about 10 to 100 steps or iterations [5]. This characteristic is the key of why convex optimisation is used in this thesis. It is no secret that convex optimisers are fast and efficient when the problem has the right configuration, i.e. functions and domains are convex, and that is why the dynamic equations of the model and the problem constraints are convexified before the solver is run, as it is seen in Section 2.1.5 and Chapter 3. This permits the MPC algorithm to obtain an optimum trajectory every time it executes the optimisation from the current position. More information about Convex Optimisation can be found in the book [5] by S. Boyd and L. Vandenberghe.
2.1.5 State of the Art Research

The literature on optimal fuel trajectories has been divided in two main topics: powered descent guidance and vertical launch. In the literature, the convex optimisation approach for guidance and control of launchers has been focused on the descending phase of the flight, as it can be seen in this subsection, but an increasing interest for the launch phase has been found lately.

2.1.5.1 Powered Descend Guidance

The study of powered descent guidance initiated with the Apollo program in the early 1960s [15] [22], using optimal control theory. These works were not taken into account for the actual guidance of the Apollo capsule, but started to show the characteristics of the optimal fuel guidance problem with a bang-bang profile. The characteristic of this profile is that the thrust only has values equal to the minimum and maximum allowed thrust.

Figure 2.6: Landing ellipses for successful Mars landings, shown on elevation map of Gale Crater. Image taken from [4]. Image credit: Ryan Anderson, USGS Astrogeology Science Center.
The interest of exploration missions to Mars in the late 1990s and 2000s prolonged the significance of constrained methods applied to the powered descent guidance problem and the pinpoint landing (i.e., landing a spacecraft on a previously selected position in the surface of a planet). Pinpoint landing in space exploration is of high importance as it potentially increases the outcome of a mission [4]. In figure 2.6 different landing ellipses for martian explorations are shown. It can be seen how the landing sites decrease in size in successive missions, which is an indicator of how the algorithms have gotten more accurate over time. These ellipses are placed over deserts and craters as the order of magnitude of the precision is measured in kilometres.

B. Açıkmese, being one of the main contributors in this area, published with S. Ploen a series of papers during the late 2000s focused on the convex programming approach to the planetary descent problem [1] [24]. They demonstrated that it can be solved as a second-order cone programming (SOCP) set of equations making use of state of the art convex optimisation solvers like interior point methods. Furthermore, their most important contribution was to introduce a convexified version of the constraint in the lower limit of the thrust magnitude. This was performed by an additional slack variable $\Gamma$ and proving that the necessary conditions of optimality were satisfied. The nonconvex constraint in the magnitude of the thrust in equation 2.5 was then transformed into a set of constraints with $\Gamma$, affecting also the mass rate of change (set of equations 2.6).

\[ 0 < T_{\min} \leq T(t) \leq T_{\max} \]  
\[ \begin{align*}
|T| & \leq \Gamma \\
T_{\min} & \leq \Gamma \leq T_{\max} \\
\dot{m} & = -\frac{\Gamma}{T_{IP} g}
\end{align*} \tag{2.6} \]

In the last decade, the enthusiasm for the guidance and control of launchers using this methodology has raised exponentially. Convex models have been developed not only for the translational dynamics, being the trend initially, but also covering the rotational dy-
namics for 6 degrees-of-freedom models. In [29], the powered descent guidance problem is tackled with the aim of successive convexification, which is a sequential convex programming (SCP) method that uses virtual control and trust region modifications to aid the convergence of the problem.

In [32], a different approach regarding the horizon of the MPC algorithm is introduced. It solves the problem several times decreasing the horizon so they can calculate where is the division between a free-ignition phase and a ignited phases. Furthermore, a parallel implementation of 2 algorithms on a multicore processor is used to solve the convex optimisation problem. This problem is relaxed in the final state constraint, as they consider that the last seconds are performed by a close loop control.

In [12], a convex optimisation approach is proposed for the guidance of the CAL- LISTO and THEMIS European demonstrators. Nevertheless, instead of taking advantage of the Model Predictive Control algorithm, a SOCP implicit guidance scheme is proposed to minimise the vertical error of the simulation of a real scenario.

The powered descend guidance problem has been researched in many different ways and it seems to be a very challenging topic that will bring new approaches in the near future.

### 2.1.5.2 Vertical Launch

The convex optimisation approach of the ascent trajectory has not been considered until the last couple of years. For that reason there is not much information available since it seems to be a very new topic. Recent studies, such as [16] [39], introduce convex optimisation with the aim to maximise the payload mass injected into orbit, which is equivalent to maximise the total mass of the rocket at the end of the flight. In [16], a convex formulation of the problem is presented, although it is not used since the initial time of the launch, but it optimise the problem when the gravity turn has been performed.
2.2 Launcher Characteristics

As the thesis is focused on the Guidance and Control of rockets, several models have
been developed and tested throughout the process, which are discussed in Section 3.1.
Nevertheless, the characteristics that those models have should be in consonance with the
vehicles which are launched almost every day. With this in mind, a survey is performed to
compare and extract the most important features of them.

Tables 2.1 contains the parameters of the engines and the physical properties of some
of the rockets that are launched nowadays almost every week or that are being developed. The
engine with the higher maximum thrust is the Merlink 1D+ from SpaceX with 845 kN,
followed by the BE-3 from Blue Origin with 490 kN. Nonetheless, the Electron rocket has
9 Rutherford engines, which at the end give a higher total thrust than the New Shepard.
The specific impulse, $I_{sp}$, of the engines of the Falcon 9 and the Electron rockets are
practically similar. In terms of shape, the Falcon 9 is by far the longest although the New
Shepard has a similar diameter. Accordingly, the vehicle with less mass is the Electron,
with an exiguous tonne of dry mass.

<table>
<thead>
<tr>
<th>Rocket</th>
<th>Falcon 9</th>
<th>Electron</th>
<th>New Shepard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company</td>
<td>SpaceX</td>
<td>Rocket Lab</td>
<td>Blue Origin</td>
</tr>
<tr>
<td>Engine</td>
<td>Merlin 1D+</td>
<td>Rutherford</td>
<td>BE-3</td>
</tr>
<tr>
<td>Thrust [kN]</td>
<td>845</td>
<td>192</td>
<td>490</td>
</tr>
<tr>
<td>Number of engines</td>
<td>9</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>Total Thrust [kN]</td>
<td>7607</td>
<td>1728</td>
<td>490</td>
</tr>
<tr>
<td>$I_{sp}$ [s]</td>
<td>311</td>
<td>303</td>
<td>-</td>
</tr>
<tr>
<td>Number of stages</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Length [m]</td>
<td>70</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>Diameter [m]</td>
<td>3.66</td>
<td>1.20</td>
<td>3.70</td>
</tr>
<tr>
<td>Wet Mass [tonnes]</td>
<td>549</td>
<td>12.5</td>
<td>-</td>
</tr>
<tr>
<td>Dry Mass [tonnes]</td>
<td>26</td>
<td>0.95</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 2.1: Comparative of present launchers.

With all this information, a customised launcher called "Dummy Rocket" was used
to run part of the simulations. It is similar to the shape and mass of the Electron and is
pushed by a single Merlin 1D+ engine. Its characteristics are presented in Table 5.1.
2.3 Software

The software development is divided in 2 parts. The first comprises an implementing and testing environment running on MATLAB, which is going to speed up the development of the guidance and control algorithm. The second part is composed of an embedded software that executes in real time and is going to probe that the final method is feasible and fast enough to be used in a real system.

2.3.1 YALMIP and SeDuMi

YALMIP is a freeware MATLAB toolbox to simplify the management of convex problems. It simplifies the interaction between the researcher and some of the most commonly used convex optimisation solvers that are available nowadays [17]. These includes SeDuMi, which is the solver used during the project.

In this thesis, YALMIP is used to defined the equations and constraints that are going to be used as a simplified model of the launcher. These are passed afterwards to the optimiser to solve the problem at a certain instant t. The equations used have to be previously convexified so the toolbox is able to determine which type of problem is an how it should be specified to the optimisation solver.

YALMIP has been selected over other possibilities, such as CVX (a package for specifying and solving convex programs [13]), because of its simplicity and the opportunities that it brings as it is connected with different solvers to compare performance. Besides, the work already done by Zapardiel in a previous Individual Research Project [38] is also implemented with this library, which establishes a precedent in the research group, and several cases have been found in the literature review where the results are based on YALMIP [12]. This, along with the pro-activity of the author to solve issues and doubts in the official forum, strengthens the opinion that is the correct software library to continue with.

SeDuMi stands for Self-Dual-Minimisation. It is a MATLAB package for solving
2.3. SOFTWARE

convex optimisation problems that takes into consideration linear equations and inequalities, second-order cone constraints, and semidefinite constraints (linear matrix inequalities) [27]. It has been selected as the main optimisation solver of the project as it is open source and the results from previous developments seem to be promising [38].

Another solver has been tested, called MOSEK, but the results obtained by this optimiser have been proven to be less accurate than the SeduMi solver, at least for the equations formulated.

2.3.2 Arduino IDE

In order to load the simulation and control programs into the hardware, an integrated development environment (IDE) is needed. In this case, the Arduino IDE has been selected due to its simplicity, well know structure and the gigantic support that can be found online. The Arduino IDE is an IDE thought for boards compatible with Arduino. It is used to write and load firmware into microcontrollers and allows to communicate with them via a serial port. This is extremely helpful when the software is being develop.

The user interface of Arduino IDE can be seen in figure 2.7. The software is separated in two main functions, setup and loop. The first one is purposed to include all the setup code, which only runs once at the beginning of the execution. The second function runs indefinitely and is where the main code belongs. In order to be able to program the desired board, it has to be added to the board list in the Arduino IDE. As it will be seen later in Section 4.2 the selected board that has to be
added to the list is the Teensy 4.1.

2.3.3 ChibiOS

ChibiOS is an open source environment for development of embedded applications which incorporates real time capabilities. Through the management of different threads, mutexes and global variables it is going to help to test the Guidance and Control algorithm in a real simulation, where the solutions have to be calculated fast.

The part of ChibiOS that will be used in this project is ChibiOS/RT, which is a high performance RTOS within ChibiOS embedded collection. It is designed to have a fully static architecture, a complete set of features with strong debug support and a clean code base [9].

The implementation of the simulation will make use of different threads, mutexes and shared variables to be able to execute the integration of the dynamic model of the vehicle while sharing information with the other board. The second board will also make use of real time capabilities to implement the convex optimiser.

2.3.4 Gitlab

The development over time of the software in the master thesis is monitored and controlled with a Version Control System (VCS). In this case the selected VCS is Git (https://git-scm.com/), which is a free and open source distributed VCS designed to handle projects efficiently. Git is composed of branches that are rooted ones in others and can be merged if necessary. This permits to develop different codes in parallel and safe the stable code separately. Furthermore, several versions of the same code can be stored in the same repository, which are identified by tags.

A Git-flow approach has been followed during the execution of the project. Git-flow was presented in 2010 by Vincent Driessen [10]. As it can be seen in figure 2.8 it defines a master branch where the stable code is stored (tagged with different versions), and the develop branch focuses the majority of the activity. From there, feature branches are
diverged, where a new feature is implemented and then merged to develop again after an intensive testing.

Figure 2.8: Git-flow branching model. Image taken from [10].

Between the different development and operation tools that include this type of VCS, Gitlab (https://gitlab.com/) is chosen because of the private repositories available for free, which allows to maintain the copyright and privacy policies of the university. It is of high importance to remark that the use of Git simplifies the control of the code developed during the thesis and allows to track the improvement over time.
2.4 Hardware

The purpose of this section is to extract the fundamental features that a space microprocessor has and then select a board similar for the simulation setup. Space electronics is a very particular topic due to the circumstances that these devices experience in the space environment. Particularly, all kinds of radiation affect negatively the performance of a central processing unit (CPU). Radiation can corrupt memory, affect calculations and even crash the execution of software due to high voltage difference. Accordingly, space grade processor must be tested on ground to prevent failures during a mission.

Prepare and test a space-hardened processor is a very expensive cost, that requires a high amount of time, cost and resources. Furthermore, this type of processor is slower than daily live processors, as they have other requirements and characteristics that affect their performance. There is a big gap between what the space-graded processor can provide in terms of computation and the necessities of the intense-computing and real-time tasks that are increasingly appearing within space systems. This tasks include autonomous guidance, storage of telemetry until download or payload on-board processing among other activities [18]. Besides, these processors can work alone, although redundancy is not discarded.

Two classical models of this type of processors are the RAD750 and the GR740 [41][42]. BAE Systems’ RAD750 is a single core CPU that is able to run up to 200 MHz and more than 2 MIPS, while the LEON GR740 is a quad-core with frequencies up to 250 MHz. The selected board for the simulation setup is a Teensy 4.1, which is able to compute instructions with a frequency up to 600 MHz. It doubles the frequency of the RAD650, but it only has one core compared to the LEON GR740. More features about the Teensy 4.1 can be seen in Table 4.4.

Another approach to tackle environmental effects is through redundancy. SpaceX vehicles have embedded a hardware architecture composed of 3 processors to prevent the system to collapse due to radiation, what is called radiation-tolerant architecture. The CPUs process the same calculation and then the results are compared. If one comput-
2.4. HARDWARE

When a processor fails, it is rebooted and synchronised with the other two. In addition, there is a fourth processor that decides which is the correct output to use [8]. An architecture of this style would be interesting to pursue in the future if the project is followed up by the research group.
Chapter 3

Theory

After studying the different topics with regard to the Guidance and Control problem, this chapter presents the theory behind the implementation conducted in this thesis. It starts with the multiple rocket models used, to present later on the general guidance problem that is going to be tackled and the constrains associated with each of the cases. These are given a context in the Launch and Landing problem sections, with the definition of the trajectory optimisation problems.

3.1 Equations of Motion and Constraints

This section introduces the different models to be used throughout the thesis. A model of a system is a mathematical representation of its behaviour in certain conditions. The multiple forces considered will be introduced and it will be explained how they relate to the surrounding environment.

According to Newton’s Second Law of Motion, the alteration of motion is ever proportional to the motive force impressed; and is made in the direction of the right line in which that force is impressed \(^{23}\). Mathematically, this can be expressed as Equation 3.1, where \(\sum F\) is the summatory of all external forces applied to a body and, for the rocket
case, the mass $m$ is not considered constant.

$$\sum \vec{F} = m \cdot \vec{v} + \dot{m} \cdot \vec{v}$$  \hspace{1cm} (3.1)

Equation 3.1 is decomposed in the various forces that build up the movement of the rocket in Section 3.1.4. Previously, these forces are explained including the reference frames in which they are represented. To describe the movement of the RLV during the different phases of the flight, several models have been considered. These start from a simple point-mass 1-dimensional model and build up until a rigid body 2-dimensional model. These are described in Sections 3.1.5 to 3.1.8.

3.1.1 Reference Frame

Prior to the introduction of the equation of motion, the frames utilised have to be described. A general global reference frame has been used in all of the cases, see figure 3.1, where the origin of the frame is set in the landing pad. Therefore, in the problems outlined further on, the rocket will be generally launched far away from the origin of the global reference frame and it will be supposed to land in the position (0,0). This reference frame simulates a flat Earth, in contrast to some of the models that can be seen in the literature, where a rounded Earth is taken into consideration [25] [32].

Apart from the reference frame, there is another frame to be considered in the modelling of the system, the body frame. This non-inertial frame is fixed to the centre of the launcher and moves with it. The X-body axis is defined along the body of the launcher passing through its centre of gravity. The Z-body axis is defined as a vector that is $\pi/2$ radians rightwise from the X-body axis and its origin is in the centre of gravity. In some of the models presented below, the angle between the body X-axis and the reference frame X-axis is taken into consideration and named as $\theta$. 
3.1. EQUATIONS OF MOTION AND CONSTRAINTS

3.1.2 Weight Force

The weight $W$ of a body depends on the gravity acceleration it is exposed to and its mass. In this particular case, the gravity field of the space body on which the launcher is flying is considered to be constant. According to this, the gravity value $g$ during the simulation will be the one at surface $g_0$. The weight force is applied to the centre of gravity $c_g$, which is supposed to be in the centre of the body, and can be described as

$$\vec{W} = \vec{g} m = g_0 m \quad (3.2)$$

Seeing that the gravity acceleration tends towards the centre of the space body and how the reference frame is characterised, the direction of $g$ will always be within the $-z$ axis and, thus, $W_z$ is equal to 0 by definition. Consequently,

$$W_z = g_0 m \quad (3.3)$$

Following the split of the weight force in the body frame, in Figure 3.2, it can be seen how the gravity turn comes into action. Determined by the orientation $\theta$ of the vehicle,
the gravity acceleration produces a normal acceleration in the $z_B$ axis that turns the vehicle path as it flies upwards.

### 3.1.3 Aerodynamic Forces

The aerodynamic forces considered are generated by the shape of the vehicle as well as from the dynamic pressure. These forces are separated in the drag $D$ and lift $L$ forces and their magnitudes are defined as in Equations 3.4 and 3.5 respectively, where $\rho_{air}$ is the density of the air at that altitude, $v$ is the velocity of the RLV at that moment, $S$ is the surface confronting the movement and $C_D$ and $C_L$ are the drag and lift coefficients. The centre of pressure $c_p$ is accepted to be in the $c_g$ and, hence, no aerodynamic moment is contemplated.

\[
D = \frac{1}{2} \rho_{air} \cdot v^2 \cdot S \cdot C_D \quad (3.4)
\]

\[
L = \frac{1}{2} \rho_{air} \cdot v^2 \cdot S \cdot C_L \quad (3.5)
\]

The values of $D$ and $L$ are affected by the velocity of the body and the angle of attack. The direction of the drag is opposite to the velocity of the vehicle, whilst the lift direction
is perpendicular to both and always with the Z component positive.

Figure 3.3: Representation of the angle-of-attack of the vehicle.

The surface considered for the drag and lift is calculated from the diagram shown in Figure 3.3. Recalling that the launcher is considered to be a cylinder for simplification, the total surface is equal to the sum of the circular extreme $S_C$ and lateral surfaces $S_L$ (Equation 3.6). The angle $\epsilon$ is the angle-of-attack of the vehicle as defined in Figure 3.3.

$$S_T = S_C \cdot \cos(\epsilon) + S_L \cdot \sin(\epsilon) = 2\pi R_L \cdot \cos(\epsilon) + R_L L_L \cdot \sin(\epsilon)$$  \hspace{1cm} (3.6)

The density of the air is treated as if the atmosphere of the planetary body were isothermal, which means that the temperature does not change with respect to the altitude. This is a simple simplification, but it will allow to better estimate the performance of the MPC controller. The isothermal model of the density is defined by Equation 3.7 \[40\]

$$\rho_{air} = \rho_0 \cdot e^{-z/H}$$  \hspace{1cm} (3.7)

where $\rho_0$ is the density of the air at ground level and $H$ is the scale height of the atmosphere. In Figure 3.4 a representation of this model can be seen, where $\rho_0$ would be
1.217 kg/m$^3$ and $H$ would be 8500 m, following the data from NASA’s Earth Fact Sheet [34]. For Mars, the surface air density would be 0.020 kg/m$^3$ and the scale height would be 11100 m [35]. In case of a flight on the Moon, the density of the atmosphere would be considered negligible, with $\rho_{\text{air}} = 0$ and, therefore, there would be no aerodynamic forces applied to the vehicle.

![Density Exponential Model](image)

Figure 3.4: Earth’s atmosphere density for the first 100 km of altitude according to the exponential model.

After the definition of the air’s density model and how the drag and lift surface of the body can be obtained, the last variables to introduce are the drag and lift coefficients. In the paper of Simplicio and Marcos regarding modelling and control of the VEGA launcher [25], several values were provided by AVIO. These are valid for the 1st stage of the launcher and are shown in Figure 3.5.

The colour of the graphs of these figures are dependent on the velocity of the vehicle, from Mach 0 to 4. As the launcher increases its speed, the drag and lift coefficients generally increase and, consequently, the aerodynamic forces. In addition to those values, the $C_D$ and $C_L$ coefficients for Mach 4 are maintained for higher velocities. Although the position of the centre of pressure $x_{CP}$ is always placed in the centre of gravity in this thesis, the diagram is left in the figure for completeness, as it also appears in [25]. The $\alpha_{\text{eff}}$ in [25] and in Figure 3.5 is the angle-of-attack represented in this project as $\varepsilon$. 
3.1. EQUATIONS OF MOTION AND CONSTRAINTS

Figure 3.5: Aerodynamic coefficients for the first stage of the VEGA launcher. Values considered for the landing phase. Data collected from [25].

3.1.4 General Equation of Motion

Wrapping the concepts seen in the previous subsections, a general dynamic equation can be drawn to be followed in the different models. Starting from Newton’s second law, Equation [3.1] the change of velocity of the vehicle can be expressed as

\[ \dot{v} = \frac{\sum\vec{F} - \dot{m} \cdot \vec{v}}{m} \]  

(3.8)

which is called the rocket equation of motion [31]. The second part of the right hand side of Equation [3.8] can be derived from the principle of the conservation of linear momentum as the propellant (thrust) force \( T \), which is the force product of the exhaust of propellant from the rocket, Equation [3.9]. The necessary steps to come to this conclusion
can be found in [31].

\[ \vec{T} = -\dot{m} \cdot \vec{v}_p \] \hspace{1cm} (3.9)

In the case of study, the motion of the rocket is subject to different forces due to its engines’ thrust and other environmental disturbances, such as the gravitational field and the atmosphere of the space body. Taking them into the equation of motion, Equation 3.8 can be expanded to

\[ \vec{v} = \frac{\vec{W} + \vec{D} + \vec{L} + \vec{T}}{m} \] \hspace{1cm} (3.10)

Equation 3.10 can be decomposed as a matrix separating the x and z components of each of the forces and velocities.

\[ \begin{bmatrix} \dot{v}_x \\ \dot{v}_z \end{bmatrix} = \frac{1}{m} \left( \begin{bmatrix} T_x \\ T_z \end{bmatrix} + \begin{bmatrix} W_x \\ W_z \end{bmatrix} + \begin{bmatrix} D_x \\ D_z \end{bmatrix} + \begin{bmatrix} L_x \\ L_z \end{bmatrix} \right) \] \hspace{1cm} (3.11)

The different models used in the thesis can be derived from Equation 3.11

### 3.1.5 Point Mass 1-dimensional Model

This simplified model in 1 dimension was developed as a first step towards the optimisation of the Rigid Body model, as it is easier to implement and analyse. The vehicle moves only in the vertical direction, \( z \), and its kinematics and dynamics are defined by the set of Equations 3.12 which are based on 3.11. The control variable is the thrust \( T \), which will be used to solve the landing problem. The rate of change of the mass \( \dot{m} \) is directly proportional to the thrust applied and it is affected by the specific impulse \( I_{sp} \) of the rocket engine and the gravity \( g \) of the planet. The aerodynamic forces are not considered for this
3.1. EQUATIONS OF MOTION AND CONSTRAINTS

model.

\[
\begin{align*}
\dot{z} &= V_z \\
\dot{V}_z &= \frac{T}{m} - g \\
\dot{m} &= -\frac{|T|}{I_{sp} g_0}
\end{align*}
\] (3.12)

This model will be used in the analysis of the Minimum Principle optimisation method and was only applied to the landing phase. The procedure to optimise this problem is described in Section 3.3.1.1.

3.1.6 Point Mass 2-dimensional Model with Thrust Magnitude and Angle

As a second step, a more advanced model covering vertical and horizontal movement was developed. The new set of Equations 3.13 represents the kinematics and dynamics equations. In this case, the control can be managed not only with the magnitude of the thrust \( T \), but also with the angle \( \alpha \) that defines its direction within the plane of movement. Therefore, the accelerations in the \( x \) and \( z \) axes are related to \( T \) via \( \alpha \) by trigonometry. The rate of change of \( V_x \) is proportional to the cosine of \( \alpha \) while the rate of change of \( V_z \) is proportional to its sine. The change of mass due to propellant exhaust is the same as in the previous model. The lift and drag forces are not considered.

\[
\begin{align*}
\dot{x} &= V_x \\
\dot{z} &= V_z \\
\dot{V}_x &= \cos(\alpha) \frac{T}{m} \\
\dot{V}_z &= \sin(\alpha) \frac{T}{m} - g \\
\dot{m} &= -\frac{|T|}{I_{sp} g_0}
\end{align*}
\] (3.13)

As well as the Point Mass 1-dimensional model, this model was only applied to the landing phase following the Minimum Principle method. The procedure to optimise this problem is described in Section 3.3.1.2.
3.1.7 Point Mass 2-dimensional Model with Thrust Components

The Point Mass 2-dimensional model with thrust components is the first model to be used with the convex optimisation method. The kinematics and dynamics equations are similar to the previous section, yet they will be applied to the whole flight from launch to touchdown. Nevertheless, the aerodynamic forces are now considered and the thrust $T$ is now proposed as components $T_x$ and $T_z$ rather than a magnitude and a direction angle. This is due to how convex optimisation works and it is described in more detail in Section 3.3.2.

\[
\begin{align*}
\dot{x} &= V_x \\
\dot{z} &= V_z \\
\dot{V}_x &= \frac{T_x - D\cos(\gamma)}{m} \\
\dot{V}_z &= \frac{T_z - D\sin(\gamma)}{m} - g \\
\dot{m} &= -\frac{|T|}{I_{sp80}}
\end{align*}
\] (3.14)

The orientation of the vehicle is not explicitly represented in the set of Equations 3.14, although one could consider the tangent of the thrust vector as that orientation, $\theta = \tan\left(\frac{T_z}{T_x}\right)$, assuming that the engines are fixed at the bottom part of the vehicle. It can become handy when the angle of attack $\epsilon$ is calculated for the aerodynamic forces. Nonetheless, the thrust vector becomes 0 in both components when the engine is off which implies that sometimes that implicit orientation cannot be obtained. Accordingly, the angle of attack is assumed to be always 0 and, therefore, only the drag force is included in the dynamic equations\[1\]. The angle $\gamma$ is defined as the climb angle of the vehicle, i.e. the angle between the horizon line and the velocity vector.

\[1\] Keep in mind that the lift force is perpendicular to the velocity, which in this case would be multiplied by the $\sin(\epsilon = 0)$, which is 0.
3.1.8 Rigid Body 2-dimensional Model

The Rigid body 2-dimensional model is the final representation of the launcher dynamics in this thesis. In this case, the vehicle jumps from a point mass to a rigid cylinder. The orientation $\theta$ and its angular velocity $\omega$ are now explicitly considered. Furthermore, as $\theta$ is now always available, the angle of attack can be properly calculated and, accordingly, the lift aerodynamic force can be introduced in the set of Equations 3.15.

\[
\begin{align*}
\dot{x} &= V_x \\
\dot{z} &= V_z \\
\dot{\theta} &= \omega \\
\dot{V}_x &= \frac{T_x - D_x - L_x}{m} \\
\dot{V}_z &= \frac{T_z - D_z + L_z}{m} - g \\
\dot{\omega} &= \frac{6(T_R - T_L)}{mL_L} \\
\dot{m} &= -\frac{|T|}{I_{sp0}}
\end{align*}
\] (3.15)

The equation related to the rate of change of the angular velocity $\omega$ is defined as the rate between the torque $M$ applied and the moment of inertia of the body $I$, Equation 3.16.

In this case, as the rocket is assumed to be a cylinder, the moment of inertia is defined as Equation 3.17.

\[
\omega = \frac{M}{I} \quad \text{(3.16)}
\]

\[
I = \frac{1}{12mL_L^2} \quad \text{(3.17)}
\]

Besides, the rocket is supposed to incorporate a pair of thrusters at the top of the body to control the orientation (see Figure 3.6). Considering that the centre of gravity $c_g$ is placed in the centre of the cylinder, the distance between those thrusters and $c_g$ will be the half of the vehicles length. This leads to the definition of Equation 3.18 as the torque applied to a body is obtained multiplying the force given by $T_R - T_L$ to the distance to its
centre.

\[ M = \frac{L_L}{2} (T_R - T_L) \]  \hspace{1cm} (3.18)

Figure 3.6: Position of the thrusters in the Rigid Body model.

### 3.2 Guidance

As it is stated in 2.1.1, there are 3 stages in the flight of a reusable launch vehicle: launch, coast and landing. The method suggested to guide a rocket is based on those phases and uses an intermediate point to reach the desired altitude in the ascent phase with a predefined velocity.

Depending on where the landing target is placed, the final phase in the proposed algorithm can be divided into 2 cases: Downrange Landing (DRL) and Return To Launch Site (RTLS). In the first scenario, the goal state vector is placed somewhere outside the launchpad, while in the second, the goal state vector is in the same position as when the
vehicle was launched.

The DRL scenario proposed will define an initial state vector on ground $s_t(0)$ for the launch phase with altitude equal to 0 and a horizontal position $x(0) \neq 0$ far from the landing site. During the ascension, the target will be at a high altitude and, when it is reached, the vehicle will shutdown the thrusters and perform an exo-atmospheric flight until it arrives at the apogee. At that moment, the descend starts and the guidance sets the target on ground, with a final state vector $s_t(t_f)$ at the origin of the reference frame [position in $s_t(t_f)$ is equal to $x(t_f) = 0$ and $z(t_f) = 0$]. The RTLS scenario is practically the same, but $s_t(0)$ is placed at the same spot as the landing target [$x(0) = z(0) = x(t_f) = z(t_f) = 0$].

In order to be able to safely land on ground, enough propellant mass must be available during the whole flight. It means that its consumption has to be as smallest as possible and, thus, there is where the optimisation problem lies. This objective index can be described as the smallest amount of thrust used, and it is shown in Equation \[3.19\]

$$J = \int_0^{t_f} |T| \, dt$$  \hspace{1cm} (3.19)

This section is focused on the Point Mass 2-dimensional model with thrust components and the Rigid Body 2-dimensional model, as they are the ones utilised for the guidance implementation.

### 3.2.1 Definition of Launch Problem

During the ascension of the vehicle, several considerations must be made. These are defined as constraints of the problem and imply that a certain state or control variable meets a given requirement. The Launch problem is a TBVP where the positions at the beginning and the end of the problem are predefined. The initial state vector $s_t(0)$ reflects that the vehicle is on ground and does not move. It tries to reach a final position $(x_f, z_f)$ in the air with a minimum velocity components. These boundary conditions are described
in Equations from 3.20 to 3.23. Equations 3.20 and 3.21 refer to the boundary constraints in the Point Mass model described in 3.1.7 and Equations 3.22 and 3.23 do the same for the Rigid Body model from 3.1.8.

\[
\begin{align*}
    x(t_0) &= 0 \\
    z(t_0) &= 0 \\
    V_x(t_0) &= 0 \\
    V_z(t_0) &= 0
\end{align*}
\] (3.20)

\[
\begin{align*}
    x(t_f) &= x_f \\
    z(t_f) &= z_f \\
    V_x(t_f) &\geq V_{x_f} \\
    V_z(t_f) &\geq V_{z_f}
\end{align*}
\] (3.21)

The initial orientation \( \theta_0 \) could be considered directly equal to \( \pi/2 \) radians. Nonetheless, the launchpad could not be necessarily aligned with the vertical and, thus, \( \theta(0) \) is kept as a variable and not as a constant.

\[
\begin{align*}
    x(0) &= 0 \\
    z(0) &= 0 \\
    \theta(0) &= \theta_0 \\
    V_x(0) &= 0 \\
    V_z(0) &= 0 \\
    \omega(0) &= 0
\end{align*}
\] (3.22)
Within the boundary equations, it can be seen that the final velocity condition is not a strict equality but it is relaxed, and the reason for this to happen is that the goal is to reach a certain altitude with the highest velocity. In addition, it is worth mentioning that during the ascension, the vehicle could reach a higher speed than the one specified with components $V_x$ and $V_z$ and decelerate the rocket during the ascend phase is not desired. This would happen if it were a strict constraint. It can be put in context if we bear in mind the possibility that the rocket carries a cargo to be placed into orbit, which requires very high velocities. This is reinforced with Constraint (3.24) that pushes the controller to increase the velocity over time. Moreover, the launcher would never go downwards before it arrives to the apogee, and Constraint (3.25) reflects that idea.

$$V_z(t) \leq V_z(t + 1) \quad (3.24)$$

$$z(t) \leq z(t + 1) \quad (3.25)$$

### 3.2.2 Definition of Landing Problem

The Landing Problem is also a TBVP but differs in several aspects from the Launch problem. First of all, the final time value restriction for the velocity components is a strict equality. When the rocket touches ground the velocity must be zero as it is pretended to be a soft landing. Conversely, the two boundary values are now turned around. The initial values are up in the air (Constraints (3.26) and (3.28)) and have to come to 0, as the landing
pad is assumed to be fixed in the origin of the reference frame (Constraints 3.27 and 3.29). In the same way as it is for the launch phase, Equations 3.26 and 3.27 identify the Point Mass model; and 3.28 and 3.29 the Rigid Body model.

\[
\begin{align*}
  &\begin{cases}
    x(t_0) = x_0 \\
    z(t_0) = z_0 \\
    V_x(t_0) = V_{x0} \\
    V_z(t_0) = V_{z0}
  \end{cases} \\
  &\begin{cases}
    x(t_f) = 0 \\
    z(t_f) = 0 \\
    V_x(t_f) = 0 \\
    V_z(t_f) = 0
  \end{cases}
\]

In contrast to the previous problem, the orientation at final time $\theta(t_f)$ is considered directly equal to $\pi/2$ radians to ensure that the rocket arrives at ground vertically. In this occasion there is no restriction on how the vertical position and velocities change over time, as the important detail is to maintain enough propellant to safely touch down.

\[
\begin{align*}
  &\begin{cases}
    x(t_0) = x_0 \\
    z(t_0) = z_0 \\
    \theta(0) = \theta_0 \\
    V_x(t_0) = V_{x0} \\
    V_z(t_0) = V_{z0} \\
    \omega(t_0) = \omega_0
  \end{cases}
\]
3.2. GUIDANCE

\[
\begin{align*}
    x(t_f) &= 0 \\
    z(t_f) &= 0 \\
    \theta(t_f) &= \frac{\pi}{2} \text{ rad} \\
    V_x(t_f) &= 0 \\
    V_z(t_f) &= 0 \\
    \omega(t_f) &= 0
\end{align*}
\]  
(3.29)

3.2.3 Common Constraints

Apart from the specific constraints for each of the phases, there are others that are common to both cases, implying restrictions in the state and control variables. Firstly, the essential constraint to start with is related to the thrust magnitude (Equation 3.30). It must be maintained within the limits of operation of the rocket engine. In Section 2.1.5, it is introduced how B. Açıkmeşe defines in [1] the minimum thrust as definite positive, \( T_{\text{min}} > 0 \). He assumes that the engine cannot be shut down during flight. Nevertheless, in Chapter 6 of *Rocket Propulsion Elements* [28], G. Sutton grants that capability to the liquid and hybrid rocket engines. Consequently, Equation 2.5 is now transformed to Equation 3.30, which will simplify the convexification of the constraint afterwards.

\[
0 = T_{\text{min}} \leq T(t) \leq T_{\text{max}}
\]  
(3.30)

Following on the state constraints to be taken into account, the position of the reusable launcher can never be below surface, hence Constraint 3.31. Additionally, the main engines should point always downwards. With this in mind, the orientation of the vehicle will always be between 0 and \( \pi \). This is represented as Equation 3.32 for both models and reinforced with Equation 3.33 for the Rigid Body model if \( \theta_{\text{max}} \) is contemplated as equal to \( \pi \).

\[
0 \leq z(t)
\]  
(3.31)
\[ 0 \leq T_z(t) \quad (3.32) \]

\[ 0 \leq \theta(t) \leq \theta_{\text{max}} \quad (3.33) \]

This orientation changes over time, so in order to prevent the rocket from rotating from one orientation to another dangerously fast, a constraint on the angular velocity must be added. For the Point Mass model, the orientation of the launcher is not explicitly considered. However, the thrust direction characterises implicitly the attitude of the rocket as the engine is supposed to be fixed to the body simulated as a point-mass. Sadly, the value of the thrust magnitude can be sometimes equal to 0 according to Constraint 3.30, which removes the possibility of defining an angle of attack during the whole flight. Despite of this drawback, Constraint 3.34 reflects the angular velocity restriction for the Point Mass model. Likewise, Constraint 3.35 defines the same concept for the Rigid Body model.

\[ |\dot{\alpha}(t)| \leq |\dot{\alpha}_{\text{max}}| \quad (3.34) \]

\[ |\omega(t)| \leq |\omega_{\text{max}}| \quad (3.35) \]

Regarding the aerodynamic forces during the flight, the angle of attack of the vehicle must be held close to zero. It is critical when it is an endo-atmospheric flight and the velocities are reasonably high, to prevent the body to fall apart. With that aim, a maximum angle of attack \( \varepsilon_{\text{max}} \) is predefined, limiting the value of \( \varepsilon \) through Constraint 3.36.

\[ |\varepsilon(t)| \leq |\varepsilon_{\text{max}}| \quad (3.36) \]

Finally, there is only one concern to be stated. In the case of the Rigid Body model, the orientation \( \theta \) must be equal to the direction of the thrust \( \alpha \), i.e. they are coupled, as the engine is supposed to be in a fixed position at the bottom of the rocket. This can be
3.3. Definition of Trajectory Optimisation

The development of the equations of motion and the definition of the problem constrains leads to the explanation of how the problem can be solved and optimised.

3.3.1 Minimum Principle

Following the studies carried out by E. Zapardiel in [38], two introductory cases of the Minimum Principle method are built up to get familiarised with the optimal control theory. This 2 problems are unconstrained and only the final state condition is enforced. It has been decided to pursue this 2 problems as they are basic enough to understand the fundamental concepts of the Minimum Principle.

3.3.1.1 Minimum Principle in 1 Dimension

In this case, the problem is defined as an unconstrained two-boundary value problem (TBVP), where the the initial and final position and velocity are described in Equations 3.38 and 3.39. This model will be used in the analysis of the Minimum Principle for a 1-dimensional model in Section 5.1.1 and was only applied to the landing phase.

\[
\begin{align*}
\theta(t) &= \alpha(t) \\
\end{align*}
\]  

(3.37)

Note: Recalling from Section 2.1.2, the equations of motion of the optimiser are based on a simplified version of the 4 models explained. The main difference is that there are no aerodynamic forces acting on the vehicle, as they would greatly increase the complexity of the problem to solve. These forces are left for the simulation phase of MPC algorithm.
\[ \begin{align*}
  z(t_f) &= 0 \\
  V_z(t_f) &= 0
\end{align*} \tag{3.39} \]

The initial values \( z(t_0) \) and \( V_z(t_0) \) are predefined as \( z_0 \) and \( V_{z_0} \) while the final values are assumed to be equal to 0, as the landing problem defines the point to land at the origin of the reference frame.

\[ J = \int_{t_0}^{t_f} T^2 \, dt \tag{3.40} \]

From the equations of motion in Equation 3.12 and considering the objective index as the square of the consumed thrust \( T^2 \), Equation 3.40, the Hamiltonian can be defined as Equation 3.41.

\[ H = T^2 + \lambda z V + \lambda V_z \left( \frac{T}{m} - g \right) + \lambda_m \left( \frac{T}{I_{sp} g} \right) \tag{3.41} \]

Nevertheless, the part of the \( \lambda_m \) related to the change of mass is neglected and the change of mass is consider to be 0 to simplify the problem. This prevents the coupling of the thrust and mass equations that would require a numerical solution.

\[ H = T^2 + \lambda z V + \lambda V_z \left( \frac{T}{m} - g \right) \tag{3.42} \]

Afterwards the necessary condition must be met, obtaining the equation that relates the control variable, i.e. the thrust, with the Lagrange multiplier \( \lambda V_z \).

\[ \frac{\partial H}{\partial T} = 0 \rightarrow \frac{\partial H}{\partial T} = 2T + \lambda \frac{V_z}{m} = 0 \rightarrow T = -\frac{\lambda V_z}{2m} \tag{3.43} \]

In order to know the values over time of the elements of the costate vector \( \hat{\lambda} \), the
costate Equation 2.4 must be solved for this particular case.

\[
\begin{align*}
\dot{\lambda}_z &= -\frac{\partial H}{\partial z} = 0 \\
\dot{\lambda}_V &= -\frac{\partial H}{\partial V_z} = -\lambda_z
\end{align*}
\] (3.44)

Integrating these derivatives, the equations of the costates are obtained, which can then be substituted in Equation 3.43 to express the thrust \( T \) dependent on the \( \lambda_{V_z} \).

\[
\begin{align*}
\lambda_z(t) &= \lambda_{z_0} \\
\lambda_{V_z}(t) &= -\lambda_{z_0} t + \lambda_{V_{z_0}}
\end{align*}
\] (3.45)

\[
T(t) = \frac{\lambda_{z_0}}{2m} t - \frac{\lambda_{V_{z_0}}}{2m}
\] (3.46)

From the equations of motion of the altitude and velocity in Equation 3.12, the formulas of \( z \) and \( V_z \) determined by the costates can be given via integration and substitution of the thrust, Equation 3.47.

\[
\begin{align*}
V_z(t) &= \frac{\lambda_{z_0}}{4m^2} t^2 + \left( -\frac{\lambda_{V_{z_0}}}{2m^2} - g \right) t + V_{z_0} \\
z(t) &= \frac{\lambda_{z_0}}{12m^2} t^3 + \left( -\frac{\lambda_{V_{z_0}}}{4m^2} - \frac{g}{2} \right) t^2 + V_{z_0} t + z_0
\end{align*}
\] (3.47)

After obtaining the different dynamic equations conditioned by the Lagrange multipliers, the only question left is the initial values of these. They can be obtained by replacement of the final conditions on \( z(t_f) \) and \( V_z(t_f) \), which are equal to 0, into Equation 3.47.

\[
\begin{align*}
0 &= \frac{\lambda_{z_0}}{4m^2} t_f^2 + \left( -\frac{\lambda_{V_{z_0}}}{2m^2} - g \right) t_f + V_{z_0} \\
0 &= \frac{\lambda_{z_0}}{12m^2} t_f^3 + \left( -\frac{\lambda_{V_{z_0}}}{4m^2} - \frac{g}{2} \right) t_f^2 + V_{z_0} t_f + z_0
\end{align*}
\] (3.48)

Clearing \( \lambda_{z_0} \) and \( \lambda_{V_{z_0}} \) in Equation 3.48 the values of the initial costates are left dependent on the initial values of the state variables and the final time \( t_f \), which is also know at
the beginning of an execution.

\[
\begin{align*}
\lambda_{z0} &= \frac{12m^2}{l_f^2} (V_{z0} + \frac{z_{00}}{l_f}) \\
\lambda_{V_{z0}} &= 2m^2 (-g + \frac{4V_{z0}}{l_f} + \frac{6z_{00}}{l_f^2})
\end{align*}
\] (3.49)

This solves analytically the unconstrained 1-dimensional problem to find an optimal control action. In the next section the model is complicated with another dimension. There, it cannot be solved analytically. Even though, these optimal control problems are still simple compared to the constrained models.

### 3.3.1.2 Minimum Principle in 2 Dimensions

As well as the previous model, it is an unconstrained TBVP, where the only constraints to be taken into account are the initial and final position and velocity (Equations 3.50 and 3.51). This model adds two new components for the position and velocity in the x axis and a thrust angle. It will be used in the analysis of the Minimum Principle for a 2-dimensional model in Section 5.1.2 and was only applied to the landing phase.

\[
\begin{align*}
x(t_0) &= x_0 \\
z(t_0) &= z_0 \\
V_x(t_0) &= V_{x0} \\
V_z(t_0) &= V_{z0}
\end{align*}
\] (3.50)

\[
\begin{align*}
x(t_f) &= 0 \\
z(t_f) &= 0 \\
V_x(t_f) &= 0 \\
V_z(t_f) &= 0
\end{align*}
\] (3.51)

Following the same strategy as in the previous section, the first step is to define the objective function \( J \) and the Hamiltonian \( H \) of the problem. The former is the same as the one used in the previous section. The latter is defined in Equation 3.52 and obtained from
the model proposed in Equation 3.13. In this case, the mass is again considered constant.

\[ H = T^2 + \lambda_x (x - V_x) + \lambda_{V_x} (V_x - \cos(\alpha) \frac{T}{m}) + \lambda_z (z - V_z) + \lambda_{V_z} (V_z - \sin(\alpha) \frac{T}{m} + g) \] (3.52)

The second step is to meet the necessary condition of optimality, which implies to partially derivate the Hamiltonian with respect to the two control variables, \( T \) and \( \alpha \). Equations 3.54 and 3.55 serve to obtain the thrust equation regarding the costate variables and \( \alpha \) is similarly defined through mathematical derivation in Equation 3.59.

\[ \frac{\partial H}{\partial \bar{c}} = 0 \] (3.53)

\[ \frac{\partial H}{\partial T} = 2T - \lambda_{V_x} \frac{\cos(\alpha)}{m} - \lambda_{V_z} \frac{\sin(\alpha)}{m} = 0 \] (3.54)

\[ T = \frac{\lambda_{V_x} \cos(\alpha) + \lambda_{V_z} \sin(\alpha)}{2m} \] (3.55)

\[ \frac{\partial H}{\partial \alpha} = \lambda_{V_x} \sin(\alpha) \frac{T}{m} - \lambda_{V_z} \cos(\alpha) \frac{T}{m} = 0 \] (3.56)

\[ \frac{T}{m} [\lambda_{V_x} \sin(\alpha) - \lambda_{V_z} \cos(\alpha)] = 0 \rightarrow \lambda_{V_x} \sin(\alpha) - \lambda_{V_z} \cos(\alpha) = 0 \] (3.57)

\[ \lambda_{V_x} \sin(\alpha) = \lambda_{V_z} \cos(\alpha) \rightarrow \frac{\sin(\alpha)}{\cos(\alpha)} = \frac{\lambda_{V_x}}{\lambda_{V_z}} = \tan(\alpha) \] (3.58)

\[ \alpha = \tan^{-1} \left( \frac{\lambda_{V_x}}{\lambda_{V_z}} \right) \] (3.59)

Subsequently, the following task is to obtain the derivatives of the costate variables through the costate equation, Equation 2.4. Once the formulas in Equation 3.44 are inte-
The issue with the analytical approach comes when the control and costate variables are substituted in the result of the integration of the equations of motion, Equation 3.13. The values of the velocity components are dependent on the sine and cosine of the thrust angle $\alpha$, which is the arctangent of the division of the costate variables $\lambda_{V_x}$ and $\lambda_{V_z}$. Certainly, a numerical approach is required to solve the problem. The strategy to do so passes through obtaining the values of the initial costate variables and then apply them to Equation 3.61. In this way, the thrust and the thrust angle can be calculated from the costate variables and the be used in the equations of motion.

The initial values of the costate variables are computed through numerical methods that require an initial guess for each $\lambda$. In this case, the values used were the ones calculated from the analytical problem in 1 dimension. Although it was only in the $z$ axis, the costate variables related to the position and velocity in $z$ were applied for the costate variables of the $x$ axis. The result of this simulations are explained in Section 5.1.

### 3.3.2 Convex Optimisation

After the equations of motion and the constraints of the Guidance problem were described in previous sections, several modifications are introduced in those equations to convexify...
3.3. **DEFINITION OF TRAJECTORY OPTIMISATION**

them. This will generate a system of equations that ultimately will allow to solve the optimisation problem. The convexification is performed by adding and changing variables that will support the construction of the problem as a Semidefinite Programming (SDP) problem \[1\]. The sources of non-convexity for the two models used are: (1) the change of mass equation \( \dot{m} = -\frac{\|\vec{T}\|}{I_{sp}g_0} \), (2) the thrust limit constraint, (3) the limitation in the rate of change of the thrust direction for the Point Mass model, (4) the angle of attack limit and (5) the coupling between the orientation and the thrust angle in the Rigid Body model.

In order to revert the non-convexity of the the first three equations, the procedure described in \[1\] and \[38\] has been followed. Initially, a change of variables is needed to overcome the non-convexity of the thrust constraint and a new slack variable \( \Gamma \) is defined in Equation 3.62. According to Lemma 1 in \[1\], this equations converts to \( \|\vec{T}\| = \Gamma \) for the optimal solution.

\[
\|\vec{T}\| \leq \Gamma 
\]  
(3.62)

\[
\sigma \triangleq \frac{\Gamma}{m}
\]  
(3.63)

\[
\vec{u} \triangleq \frac{\vec{T}}{m}
\]  
(3.64)

\[
\eta = ln(m)
\]  
(3.65)

Later, Equation 3.62 is modified using the new variables \( \sigma \) and \( |\vec{u}| \) into Equation 3.66 and the change of mass equation can be transformed to Equation 3.68 with the aid of the definition of \( \beta \) as the fuel consumption rate. Moreover, for the Rigid body model the variables \( u_R \) and \( u_L \) are introduced to represent the lateral thrusters in the convexified
version of the model.

\[ \| \vec{u} \| \leq \sigma \]  
\( \beta = -\frac{1}{I_{sp} g_0} \)  
\[ \dot{\eta} = \frac{\dot{m}}{m} = -\beta \sigma \]  
\[ u_R = \frac{T_R}{m}, \quad u_L = \frac{T_L}{m} \]  

In addition, the objective index defined in 3.19 can now be updated with the new \( \sigma \) variable as Equation 3.70.

\[ J = \int_0^{t_f} \sigma \, dt \]  

The particular constraint of the Point Mass model is the limitation in the rate of change of the thrust direction. Expressing the Constraint 3.34 as the dot product of two contiguous thrust vectors over time \( \vec{T}(t) \) and \( \vec{T}(t + \Delta t) \), Acikmese carries out a series of operations to convexify this restriction from Equation 3.72 to 3.73, which is convex \( \Omega \). In these equations, \( \Omega \) is a variable that represents the cosine of the maximum angular velocity multiplied by \( \Delta t \), and helps to hide the cosine to the convex model.

\[ \Omega = \cos(\omega \Delta t) \]  
\[ \frac{\vec{T}(t) \cdot \vec{T}(t + \Delta t)}{\| \vec{T}(t) \| \| \vec{T}(t + \Delta t) \|} \geq \Omega \]  
\[ \| Q^T z_k \| \leq \frac{\sqrt{1 - \Omega}}{\sqrt{2}} [\sigma(t) + \sigma(t + \Delta t)] \]
where $Q$ is a 4x4 matrix defined in (3.75) and $z_k$ is a column vector of the variable $\vec{u}$ at times $t$ and $t + \Delta t$.

$$z_k = (\vec{u}(t + \Delta t), \vec{u}(t))^T \quad (3.74)$$

$$Q = \begin{pmatrix}
1 & 0 & -\frac{1}{2} & 0 \\
0 & 1 & 0 & -\frac{1}{2} \\
-\frac{1}{2} & 0 & 1 & 0 \\
0 & -\frac{1}{2} & 0 & 1
\end{pmatrix} \quad (3.75)$$

Using the same convexification steps as for the constraint on the rate of change of the thrust direction, the angle of attack can be convexified too. Nevertheless, there are two considerations to bear in mind when this method is applied compared to the previous constraint. Instead of two successive thrust vectors over time, the ones utilised are the thrust vector and the vector composed of the velocity components of the model at the same time $t$. Moreover, instead of $\Omega$, $\Upsilon$ is now defined as the cosine of the maximum angle of attack permitted.

$$\Upsilon = \cos(\varepsilon) \quad (3.76)$$

$$\|Q^{\frac{1}{2}}y_k\| \leq \frac{\sqrt{1-\Upsilon}}{\sqrt{2}}[\sigma(t) + v(t)] \quad (3.77)$$

where $y_k$ is a column vector of the thrust and velocity components at time $t$.

$$y_k = (\vec{v}(t), \vec{u}(t))^T \quad (3.78)$$

Lastly, the particular constraint that must be applied to the Rigid Body model is focused on the connection between the orientation and the thrust direction. These are not explicitly related on the equations of motion and, therefore, the thrust components must
be restricted to the attitude angle of the launcher. In [29], a constraint on the direction of the thrust is imposed for a predefined gimbal angle $\delta_{\text{max}}$, Equation 3.79, where $e$ is a perpendicular vector to the direction of movement.

$$
\cos \delta_{\text{max}} \|u(k)\| \leq e \cdot u(k) \tag{3.79}
$$

Imposing that the maximum gimbal angle for the case at hand is 0, the Constraint 3.79 transforms to Constraint 3.80.

$$
\|u(k)\| \leq e \cdot u(k) \tag{3.80}
$$

### 3.3.3 Discretisation of Equations of Motion

The final step to be able to use the equations of motion of the models in the convex optimisation tool, YALMIP, is to discretise them. It can be performed by a finite-difference approximation of the derivatives over time, where the constraints are applied to the temporal nodes. The thrust is maintained between a time $k$ and $k + 1$ [38]. The systems of Equations 3.81 and 3.82 represent the result of this discretisation for the Point Mass and Rigid Body models respectively.

\[
\begin{align*}
x(k + 1) &= u_1(k) \ dt^2 + V_x(k) \ dt + x(k) \\
z(k + 1) &= (u_2(k) - g) \ dt^2 + V_z(k) \ dt + z(k) \\
V_x(k + 1) &= u_1(k) \ dt + V_x(k) \\
V_z(k + 1) &= (u_2(k) - g) \ dt + V_z(k) \\
\eta(k + 1) &= -\beta \sigma(k) \ dt + \eta(k)
\end{align*}
\]
3.4 Model Predictive Control Algorithm

Concluding with the introduction of the theory developed throughout the thesis, a modification to the Model Predictive Control algorithm is presented. It changes how the algorithm behaves over time, trying to reach the goal state in a predefined time and overcoming the involved issues in the receding horizon MPC.

3.4.1 Receding Horizon MPC

The receding horizon MPC strategy grows from the base that its embedded controller obtains a solution for the problem for a static horizon $t_f$. Nonetheless, as it gets closer to the goal, the MPC internal solver still generates a control vector that will reach the desired state at time $t + t_f$. This implies that the controller will maintain the rocket forever flying above the landing spot. This issue was also mentioned by Zapardiel [38] and is simulated and analysed in Section 5.2.

3.4.2 Decreasing Horizon MPC

The proposed modification to the MPC method focuses in the adjustment of its horizon to improve the performance of the algorithm. For this purpose, a new variable is introduced, called Time Percentage Update Factor (TPUF). This factor determines when and by which rate the horizon is decreased. The value of this new variable must be set between 0 to 1.
In Figure 3.7, a diagram demonstrates how this update works. Assuming that the algorithm is running with a TPUF of 0.5, the simulation would run for half of the maximum simulation time $t_{\text{sim max}}$ (because TPUF is exactly 0.5) with the horizon $t_h$ specified. Then, these parameters are decreased by a factor of $(1 - \text{TPUF})$ and the MPC performs the control with the new $t_{\text{sim max}}$ and $t_h$. When the horizon is shorter in time than the $\Delta t$ of the simulation, the rocket uses the last control generated by the MPC and finishes the execution. In this new manner, the new MPC algorithm prevents the RLV to never reach the landing target.
### Table 3.1: Constraints for the Launch and Landing problems for the models using the MPC with Convex Optimisation.

<table>
<thead>
<tr>
<th>Id</th>
<th>Constraints</th>
<th>2D Point Mass model</th>
<th>2D Rigid Body model</th>
<th>Launch</th>
<th>Landing</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Initial state vector</td>
<td>$x(0) = x_0, \ z(0) = z_0, \ V_x(0) = x_0, \ V_z(0) = z_0$</td>
<td></td>
<td>(3.20) (3.22)</td>
<td>(3.26) (3.28)</td>
</tr>
<tr>
<td>2A</td>
<td>Final state vector</td>
<td>$x(t_f) = x_f, \ z(t_f) = z_f, \ V_x(t_f) = x_f, \ V_z(t_f) = z_f$</td>
<td></td>
<td></td>
<td>(3.27) (3.29)</td>
</tr>
<tr>
<td>2B</td>
<td>Final state vector</td>
<td>$x(t_f) = x_f, \ z(t_f) = z_f, \ V_x(t_f) \geq x_f, \ V_z(t_f) \geq z_f$</td>
<td>(3.21) (3.23)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Maximum and minimum Thrust</td>
<td>$0 = T_{\text{min}} \leq T(t) \leq T_{\text{max}}$</td>
<td></td>
<td>(3.30)</td>
<td>(3.30)</td>
</tr>
<tr>
<td>4</td>
<td>Positive altitude</td>
<td>$0 \leq z(t)$</td>
<td></td>
<td>(3.31)</td>
<td>(3.31)</td>
</tr>
<tr>
<td>5</td>
<td>Thrust orientation</td>
<td>$0 \leq T_z(t)$</td>
<td></td>
<td>(3.32)</td>
<td>(3.32)</td>
</tr>
<tr>
<td>6</td>
<td>Rocket orientation</td>
<td>$- \quad 0 \leq \theta(t) \leq \theta_{\text{max}}$</td>
<td></td>
<td>(3.33)</td>
<td>(3.33)</td>
</tr>
<tr>
<td>7</td>
<td>Thrust direction rate of change</td>
<td>$</td>
<td>\alpha(t)</td>
<td>\leq</td>
<td>\alpha_{\text{max}}</td>
</tr>
<tr>
<td>8</td>
<td>Rocket orientation rate of change</td>
<td>$- \quad</td>
<td>\omega(t)</td>
<td>\leq</td>
<td>\omega_{\text{max}}</td>
</tr>
<tr>
<td>9</td>
<td>Angle of attack</td>
<td>$</td>
<td>\epsilon(t)</td>
<td>\leq</td>
<td>\epsilon_{\text{max}}</td>
</tr>
<tr>
<td>10</td>
<td>Increasing altitude</td>
<td>$z(t) \leq z(t+1)$</td>
<td>(3.25)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Increasing velocity in z axis</td>
<td>$V_z(t) \leq V_z(t+1)$</td>
<td>(3.24)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Coupling between orientation and thrust direction</td>
<td>$- \quad \theta(t) = \alpha(t)$</td>
<td>(3.37)</td>
<td>(3.37)</td>
<td>(3.37)</td>
</tr>
</tbody>
</table>
Chapter 4

Implementation

In this chapter the implementation of the theory and models explained in Chapter 3 is introduced. First, the MATLAB implementation of the Minimum Principle problems in 1D and 2D is discussed. Later on, the Model Predictive Control Algorithm with the Convex Optimiser is dissected in several parts. Various versions have been developed during the implementation phase, and they are described at the end of Section 4.1. In Section 4.2 a design for the real time simulation test bench is proposed, although the complete implementation of it has not been finished. Finally, Section 4.3 talks about how the development of the project benefited from the version control system (VCS).

4.1 MATLAB

Following the description of the 4 models in Section 3.1 this section presents the implementation of those models and the associated control laws in MATLAB. This is a necessary step before programming the MPC controller on an embedded microcontroller as it allows a faster validation of the optimisation solver. This code does not aim to prove that the optimisation tools are fast enough to control a real launcher, but to demonstrate that they are able to solve robustly the guidance and control problem.
CHAPTER 4. IMPLEMENTATION

4.1.1 Model Predictive Control with Minimum Principle

The MPC using the Minimum Principle optimisation tool has been implemented following the theory and equations in Section [3.3]. This development is divided into two parts, one regarding the 1D analytical problem and another for the 2D numerical problem.

The most important part of the main script for both problems is represented in the MATLAB pseudocode below. The algorithm runs calculating, first, the Model Predictive Control thrust to arrive at the destination for a predefined horizon in time with the help of the `horizonMPC()` function; then, a simulation is executed for a small delta t using the `ode45()` MATLAB built-in function and the custom function `rocketPlant()`, which contains the ordinary differential equations of the vehicle’s plant. Afterwards, using the new state vector, the MPC recalculates a new control thrust with the same objective and a numerical integration is run afterwards again. The algorithm runs for a maximum number of iterations while checking if the difference between the current state vector and the objective $(0,0)$ is less than the error margin.

```matlab
% Initialise constants and initial state vector
% Create vectors for final values
while distance > error_margin && it < n_it
    % Call Controller
    [..] = horizonMPC(state_vector, g, mpc_horizon_time, ...
        mpc_integ_step);

    % Simulation using controller output
    [t, st] = ode45(@(t, st) rocketPlant(..), ..);

    % Safe iteration values
    % Calculate distance
end
```

`horizonMPC()` obtains the thrust with respect to time for controlling the descent of
the vehicle. It also returns the value of the lambdas and the state vector used to obtain the thrust. The goal of this algorithm is to land at the position \((0, 0)\) with 0 velocity. In the 1D analytical problem the initial lambdas are calculated with Equations 3.49, while in the 2D numerical problem the initial values for the lambdas, or Lagrange coefficients, are obtained using the MATLAB \texttt{fmincon()} function. \texttt{fmincon()} is a general solver for non-convex problems that numerically minimises a passed function considering a predefined set of constraints. It needs an initial guess of the values of the lambdas to calculate the optimised initial lambdas. This guessed value is the one obtained in the analytical problem. If the guess is not proper, the function can wander indefinitely.

\texttt{rocketPlant()} simulates the dynamics of a launch vehicle in the air. These dynamics are represented by the equations of motion defined in Sections 3.1.5 and 3.1.6.

### 4.1.2 Model Predictive Control with Convex Optimisation

For the MPC implementation, a personalised MPC Controller has been developed. This is preferred rather than using MATLAB’s MPC toolbox, because it gives more flexibility on how to code it and modify its horizon and the Convex Optimisation solver. Furthermore, it is of high importance for the reasons stated in Section 3.4.

This software is designed following an object-oriented approach. Rockets, environments and problems are coded as interfaces which adds a modular capability to the software. A Unified Modelling Language (UML) diagram of the implementation of the 2D Landing problem is presented in figure 4.1. It is used here as an example to represent a specific case tackled in this thesis, although a 2D Launch or Guidance problem could also have been used. In this diagram, the main script is shown in green, while the abstract and implemented classes appear in blue. The software is designed with the idea of being reusable for other problems. Different rockets and environments can be tested through the \texttt{AbstractRocket} and \texttt{Environment} classes. Likewise, multiple optimisers could be tested following the \texttt{AbstractMPCCController} interface.
Figure 4.1: UML diagram of the MATLAB Implementation of the 2 dimensional landing case. The parameters in the config file with an asterisk refer to the extra parameters for the comparative case.
4.1. MATLAB

4.1.2.1 Main Scripts

For the simulation and testing, several main scripts have been developed. These can be
categorised in one-case execution and multiple-case execution. The former are focused
on running the simulation once, while the latter perform a comparative analysis between
multiple cases. The one-case scripts are:

- **Main2DLaunch**: simulates the descent phase in which it has to perform a soft land-
ing at a given time.

- **Main2DLanding**: simulates the ascent phase to reach a desired altitude with a cer-
tain velocity.

- **Main2DGuidance**: this script is divided in three parts. First, it performs a launch
  simulation until it reaches the desired altitude. Second, it simulates a coastal phase,
  which could be used for a stage separation, for instance. In this phase the engines
  are turned off and waits until the vehicle reaches the apogee. Third, the guidance
  algorithm executes the descent control algorithm to land on the desired position.

A multiple-case script has been implemented and it is called **Main2DComparative**. It
performs a Monte Carlo analysis of the descent and landing phase, generating aleatory
initial state vectors for the rocket. Afterwards, the MPC controller tries to land it at the
specified final time.

These scripts are configured with the **Config** class. It defines the initial and final
desired state vectors (which implies position, velocity and mass), the horizon and delta t
values of the convex optimiser, and the simulation and time percentage update used in the
proposed MPC algorithm. A simulation is considered successful when the final state is
within the error position and velocity margins defined in this class. Lastly, **Config** specifies
the **Rocket** and **Environment** instances to be used. At the beginning of the execution, the
**configProcessor()** function extracts all this information and prepares all the necessary
variables before the simulation starts.
The \textit{MPCAndSimulation()} function is in charge of running the simulation, including the convex optimisation and the numerical integration. It first creates the \textit{MPCController} instance that will be used to find a feasible control strategy to place the launcher in the desired state. Then, it iterates over time until one of several conditions has been reached. These conditions can be:

- The rocket is within acceptable margin around the desired final state.
- The rocket has reached ground. In case of a launch simulation, it also considers reaching the desired altitude as a final condition.
- The simulation time has been completed.

At the end of the simulation, the state vector components and the thrust profile are plotted to analyse the result. These figures are the ones shown in Chapter 5. Finally, the values are stored in an instance of the \textit{Result} class. This class is thought to be a container of all the necessary information to replicate and process a test.

\textbf{4.1.2.2 State Propagator}

The numerical integration of the state vector of the vehicle is handled with a custom function called \textit{StatePropagator()}, which is passed to the \texttt{ode45()} MATLAB built-in function. This MATLAB function integrates the differential equations defined in \textit{StatePropagator()} through a Runge-Kutta execution of 4\textsuperscript{th} order. The implemented differential equations are previously described in Sections 3.1.7 and 3.1.8.

\textit{StatePropagator()} receives as arguments the current time \( t \) and the state vector \( st \), the thrust profiles —\texttt{thrust, thrust.x and thrust.z}— as arrays and their assigned time \texttt{thrust.time}, and the \textit{Rocket} and \textit{Environment} instances configured for the test. The classes of these 2 instances are defined in the UML diagram of Figure 4.1. The thrust to apply for a certain time \( t \) is obtained comparing \( t \) with the values within the \texttt{thrust.time} array and finding the immediately smaller value in it. This value is the index of the correct thrust magnitude and components in the thrust arrays. This is performed between lines 7
to 17 of the MATLAB pseudocode below. This pseudocode does not represent the whole implementation of the function, but the most representative lines. The state vector used there is \([x,z,V_x,V_z,m]\), which corresponds to the model defined in Section 3.1.7.

```matlab
function st_dot = StatePropagator(t, st, thrust, thrust_x, ...  
    thrust_z, thrust_time, rocket, environment)

% Change in position
st_dot(1) = st(3);
st_dot(2) = st(4);

% Obtains the thrust for the current time if there is still mass ...
% in the rocket
if st(5) > 0
    thrust_index = find(t ≥ thrust_time, 1, 'last');
    thrust_at_time = thrust(thrust_index);
    thrust_x_at_time = thrust_x(thrust_index);
    thrust_z_at_time = thrust_z(thrust_index);
else
    thrust_at_time = 0.0;
    thrust_x_at_time = 0.0;
    thrust_z_at_time = 0.0;
end

% Obtains the aerodynamic forces according to the equations in ...
% Chapter 3 of the thesis report
% - Velocity magnitude in Mach
% - Angle of attack
% - Surface exposed
% - Air density at the current altitude
% - Drag and Lift coefficients
% - Drag and Lift forces
```
% Change in velocity
if st(5) > 0
    st_dot(3) = (thrust_x_at_time - drag_force + cos_climb_angle ... 
                  - lift_force * sin_climb_angle) / st(5) ;
    st_dot(4) = (thrust_z_at_time - drag_force * sin_climb_angle ... 
                  + lift_force * cos_climb_angle) / st(5) - ... 
                  environment.gravity;
end

% Change in mass according to the equations in Chapter 3 of the ... 
  thesis report
..
end

The drag and lift forces are calculated from the state vector and the parameters defined in the *Rocket* and *Environment* instances, such as rocket diameter, air density at sea level and gravity. Following the definition of these forces in Section 3.1, the code determines the angle of attack and the surface exposed, as well as the air density, prior to the calculation of the aerodynamic forces. The values used for the drag and lift coefficients are based on Ariane’s VEGA launcher, given in [25] (see figure 3.5). These values are stored in two matrices, see tables 4.1 and 4.2, in the *AbstractRocket* class. During the numerical integration of the simulation, the current value is extracted through interpolation, using the *interp2* MATLAB function, for the calculated angle of attack and velocity magnitude in radians and Mach respectively. Although in figure 3.5 the \(C_D\) and \(C_L\) are shown up to a relative velocity of 4 Mach, these values are maintained for higher velocities.

Finally, there are two event functions, *groundReachedEvent()* and *apogeeReachedEvent()* that monitor the status of the vehicle and halt the numerical integration when the altitude or the velocity in z is equal or smaller than 0 respectively. The first function is considered in all the simulations while the second is only added in the coastal phase of the *Main2DGuidance* script.
4.1. MATLAB

<table>
<thead>
<tr>
<th>Velocity &amp; AOA</th>
<th>0</th>
<th>π/6</th>
<th>π/3</th>
<th>π/2</th>
<th>2π/3</th>
<th>5π/6</th>
<th>π</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.5</td>
<td>0.9</td>
<td>1.2</td>
<td>1.7</td>
<td>2.0</td>
<td>1.5</td>
<td>1.2</td>
<td>0.4</td>
</tr>
<tr>
<td>0.8</td>
<td>1.1</td>
<td>1.4</td>
<td>2.8</td>
<td>3.3</td>
<td>2.9</td>
<td>1.6</td>
<td>0.5</td>
</tr>
<tr>
<td>1.0</td>
<td>1.1</td>
<td>2.8</td>
<td>4.2</td>
<td>4.5</td>
<td>3.8</td>
<td>2.2</td>
<td>1.0</td>
</tr>
<tr>
<td>1.2</td>
<td>1.4</td>
<td>2.8</td>
<td>4.5</td>
<td>5.5</td>
<td>5.0</td>
<td>2.6</td>
<td>1.2</td>
</tr>
<tr>
<td>1.6</td>
<td>1.6</td>
<td>2.8</td>
<td>4.2</td>
<td>5.2</td>
<td>4.5</td>
<td>2.6</td>
<td>1.3</td>
</tr>
<tr>
<td>2.0</td>
<td>1.6</td>
<td>2.3</td>
<td>4.3</td>
<td>5.4</td>
<td>4.2</td>
<td>2.2</td>
<td>1.4</td>
</tr>
<tr>
<td>4.0</td>
<td>1.7</td>
<td>2.0</td>
<td>4.0</td>
<td>4.9</td>
<td>3.3</td>
<td>2.0</td>
<td>1.3</td>
</tr>
<tr>
<td>1000.0</td>
<td>1.7</td>
<td>2.0</td>
<td>4.0</td>
<td>4.9</td>
<td>3.3</td>
<td>2.0</td>
<td>1.3</td>
</tr>
</tbody>
</table>

Table 4.1: Drag coefficients used in the simulation. The row index defines the relative velocity of the vehicle with respect to the atmosphere. It is represented in Mach. The column index defines the angle of attack (AOA) of the vehicle. It is represented in radians. This table is based on figure 3.5.

<table>
<thead>
<tr>
<th>Velocity &amp; AOA</th>
<th>0</th>
<th>π/6</th>
<th>π/3</th>
<th>π/2</th>
<th>2π/3</th>
<th>5π/6</th>
<th>π</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.5</td>
<td>0.0</td>
<td>0.5</td>
<td>0.6</td>
<td>0.0</td>
<td>-0.3</td>
<td>-0.5</td>
<td>0.0</td>
</tr>
<tr>
<td>0.8</td>
<td>0.0</td>
<td>0.7</td>
<td>1.1</td>
<td>0.0</td>
<td>-1.1</td>
<td>-1.1</td>
<td>0.0</td>
</tr>
<tr>
<td>1.0</td>
<td>0.0</td>
<td>1.8</td>
<td>1.5</td>
<td>0.0</td>
<td>-1.8</td>
<td>-1.8</td>
<td>0.0</td>
</tr>
<tr>
<td>1.2</td>
<td>0.0</td>
<td>1.6</td>
<td>1.5</td>
<td>0.0</td>
<td>-1.4</td>
<td>-1.4</td>
<td>0.0</td>
</tr>
<tr>
<td>1.6</td>
<td>0.0</td>
<td>1.3</td>
<td>1.6</td>
<td>0.0</td>
<td>-1.6</td>
<td>-1.4</td>
<td>0.0</td>
</tr>
<tr>
<td>2.0</td>
<td>0.0</td>
<td>1.1</td>
<td>1.6</td>
<td>0.0</td>
<td>-1.4</td>
<td>-1.2</td>
<td>0.0</td>
</tr>
<tr>
<td>4.0</td>
<td>0.0</td>
<td>0.6</td>
<td>1.7</td>
<td>0.0</td>
<td>-1.1</td>
<td>-0.9</td>
<td>0.0</td>
</tr>
<tr>
<td>1000.0</td>
<td>0.0</td>
<td>0.6</td>
<td>1.7</td>
<td>0.0</td>
<td>-1.1</td>
<td>-0.9</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Table 4.2: Lift coefficients used in the simulation. The row index defines the relative velocity of the vehicle with respect to the atmosphere. It is represented in Mach. The column index defines the angle of attack (AOA) of the vehicle. It is represented in radians. This table is based on figure 3.5.

4.1.2.3 Implementation Issues

During the implementation of the convexified models and constraints defined in Chapter 3 several issues were encountered. The implementation of the angle of attack defined in Equation 3.77 seems to be erroneous for some reason that has not been found yet. Multiple approaches were tried but with no result. This is one of the main obstacles found during the entire duration of the thesis.

On the other hand, the implementation of the maximum thrust constraint generated an issue in the optimised solver that prevent the simulation of scenarios of more than 10
kilometres of distance in the first months of the programming part of the thesis. Neverthe-
less, this difficulty was overcome by simplifying the thrust constraint to a linear limit that
over-restricted the thrust by defining the maximum value for $\sigma$ as the maximum engine
thrust $T_{\text{max}}$ divided by the mass at the beginning of the problem. This was the constraint
used in the simulations of Chapter 5.

$$0 \leq \frac{T_{\text{max}}}{m_0} - \sigma(t)$$

(4.1)

4.1.3 Code Versions

The MATLAB software has been created incrementally from version 0.1 to version 0.2,
gaining different features to increase the performance and veracity of the simulations and
tests. Versions 0.1.1-3 are focused on the point mass model, while version 0.2 is centred
on the rigid body model. Version 0.2 was developed after the Point Mass implementation
was done and, therefore, it includes all the features previously thought for the Point Mass
adequate to the new model. However, version 0.2 is still in development and is not used in
the simulations in Chapter 5. Table 4.3 shows the different added features in each version.
This information is completed in the README.md file in the Appendix A.

<table>
<thead>
<tr>
<th>Version</th>
<th>New features</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>Implementation of the 2 MPC Minimum Principle problems for soft landing. MPC Convex optimisation problem with receding horizon. Initial implementation of the MPC with decreasing horizon for Point Mass model.</td>
</tr>
<tr>
<td>0.1.1</td>
<td>Change in the maximum thrust constraint to reach higher altitudes.</td>
</tr>
<tr>
<td>0.1.2</td>
<td>Implementation of the launch phase. New constraint on the rate of change of the thrust vector orientation. New constraint on the angle of attack. Constraints on $z$ and $V_z$ values for the launch phase. Full flight Guidance simulation added.</td>
</tr>
<tr>
<td>0.1.3</td>
<td>Addition of new Rockets and Environments. Addition of the aerodynamic forces into simulation model. Relaxation of velocity final value for the launch problem.</td>
</tr>
<tr>
<td>0.2</td>
<td>Rigid Body dynamics in the convex optimiser and the numerical integration.</td>
</tr>
</tbody>
</table>

Table 4.3: MATLAB code versions and new features.
4.2 Real Time Simulation Setup

After developing the simulation in MATLAB and setting all the different parameters and features of the predictive controller, the idea would be to perform real time simulations to prove that the algorithm is worth to be considered for a real mission. This can be achieved through an embedded implementation of the model predictive controller and testing it over a real simulation. There is also a computer in the system, however it does not interfere. It used to monitor the test and it is not considered for the real time simulation as it is harder to ensure real time performance on a PC than on a microcontroller, due to the overkilling layers of the Operating System. In this section a Real Time Simulation setup is presented, covering the different aspects of hardware and software that it involves.

4.2.1 Hardware

The design of the real time simulation setup is based on two microcontrollers: the Controller Board and the Simulation Board. The former is centred on the numerical simulation of the vehicle state vector, while the latter calculates the control actions to perform the guidance and control of the rocket. These communicate with each other through Serial to transmit the state vector and the control actions, while the Simulation Board also sends some data to the computer.

The selected microcontroller model to use in the simulation setup is a Teensy 4.1, which is a 32 Bit Arduino-Compatible microcontroller. This board contains an ARM Cortex-M7 running up to 600 MHz, which is powerful enough to perform the vehicle simulation in real time and compute the optimised control algorithm. It contains a microSD card slot, which is handful when debugging and logging data is needed and an Ethernet connection that could be interesting to use in a future project. Furthermore, its low price is an advantage that must be taken into account for this kind of projects. More characteristics can be found in Table 4.4 and [26].

Figure 4.2 describes how the setup would be built. The Serial port to be used on the
CHAPTER 4. IMPLEMENTATION

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>ARM Cortex-M7 at 600 MHz</td>
</tr>
<tr>
<td>Memory</td>
<td>1 MB of RAM and 8 MB of Flash Memory</td>
</tr>
<tr>
<td>Communication</td>
<td>2 USB ports, 3 CAN Bus, 8 Serial ports, 1 Ethernet port at 10 / 100 Mbit and 55 GPIO pins, among other types of communication ports</td>
</tr>
<tr>
<td>Storage</td>
<td>1 Micro SD card socket</td>
</tr>
</tbody>
</table>

Table 4.4: Some of the most important Teensy 4.1 technical specifications [26].

Teensys for communication would be the one in pins 0 and 1 (purple and lilac connections). The power would be supplied by 5V batteries connected to the \( V_{in} \) and GND pins, placed on the top part of the board (red and black connections), and the Simulation Board would be connected through USB to the computer (light blue connection). A microSD card would be inserted into the bottom microSD card slot of each of them. The complete pinout of the board can be found in Appendix B.

Figure 4.2: Hardware setup for the embedded simulation.
4.2. REAL TIME SIMULATION SETUP

4.2.2 Software

The embedded code would be written in C/C++ within the chibiOS RT framework. It is compatible with Arduino-alike microcontrollers and, therefore, can be coded in the Arduino Integrated Development Environment (IDE). Accordingly, the firmware of each boards would be loaded through this IDE.

The design of the real time software depends on the microcontroller to be applied to. Three threads run on the Simulation Board while the Controller Board has a single thread executing within a loop (see Figure 4.3). The threads running on the Simulation Board share 2 global arrays for the state vector and the thrust profile over time, which are access-controlled by 2 mutexes. These threads deal with the following tasks:

- **Thread 1**: it carries out the communication with the other devices in the system. Through Serial communication, it sends the current state vector to the Controller board and receives the thrust profile to be applied in the numerical integration. Likewise, it also forwards the current state vector to the computer with a smaller frequency, to assess if the simulation is running or not.

- **Thread 2**: it handles a numerical integration of the current state vector through a custom implementation of the Runge-Kutta method of 4th order. It has a high cadence to replicate the status of the vehicle in real time. This thread is the main reason why a RT framework is needed.

- **Thread 3**: it stores the state vector and the received thrust profile in a file within the microSD card. This provides a post-processing stage after the simulation, to draw figures based on the simulation values and interpret the results.

The thread running on the Controller Board is described in Algorithm 1. It awaits until the simulation board tells it to obtain the necessary thrust to reach a target position and then answers with the solution found, if the problem is feasible. At this point, CVXGEN or a custom implementation of the interior point method would be used in order to
Algorithm 1 Controller Board thread

1: \textbf{while} Always \textbf{do} \\
2: \hspace{1em} Wait until a state vector is passed from the Simulation Board \\
3: \hspace{1em} Calculates the thrust profile to reach the desired state (position and velocity) in \\
\hspace{1em} the given final time \\
4: \hspace{1em} Forwards the thrust profile to the Simulation Board \\
5: \hspace{1em} Stores the thrust profile into the microSD card \\
6: \textbf{end while} \\

program the convex optimiser in C/C++. Nonetheless, this second option would involve a 
higher amount of development time. CVXGEN is a code generator for small convex op-
timisation problems turning a mathematical description of the convexified problem into 
C/C++ code [21].

Figure 4.3: Software design for the real time simulation setup.
4.3 Repository

A GitLab group was established with the supervisor as owner with 2 private repositories to store all the developed code, see Figure 4.4. The fact that Dr. Felicetti has an owner role in the project ensures the accessibility of the code for future students keen on continuing this project.

![Image of CranfieldRocketGNC group with repositories](image_url)

Figure 4.4: CranfieldRocketGNC group with the repositories.

In the MATLABSimulation repository, multiple branches were created throughout the project following the Git-flow methodology. These can be divided into several categories: master, develop, release and feature, each of them having different purposes. The branches alive at the end of this project are:

- **master**: contains the stable code. All the versions implemented and used in Chapter 5 are stored here when they are finished.

- **develop**: contains the stable code being developed, before it is closed in the master branch.

- **release X**: is the step between develop and master. The X is the version code for each release, and their new features are presented in Table 4.3.

- **feature_comparative_mpc**: implementation of the Monte Carlo analysis code.

- **feature_rigid_body**: implementation of the rigid body dynamics and constraints for the MPC with Covex Optimiser problem.
Two other important branches were used during the development: \textit{feature\_orientation} and \textit{feature\_waypoint\_landing}. These were discarded after a considered amount of time was spent on them and, thus, it is important to remark what was the idea behind them. \textit{feature\_waypoint\_landing} was a first approach to solve the initial restriction on the state space of the problem due to SeDuMi. The concept was based on a division of the trajectory in several waypoints strategically selected, so that the problem to be solved was a succession of smaller problems. However, the performance of the optimisation solver about those waypoints was rather poor and consequently it was discarded. \textit{feature\_orientation} contained a different Rigid Body model with 3 thrusters at the bottom of the rocket. However, this significantly complicates the control of the angular velocity of the vehicle. For that reason, the current Rigid Body model was introduced.
Chapter 5

Simulations and Results

After having discussed the background and theory related to Model Predictive Control, optimisation and the models developed, this section will be focused on proving the feasibility and singularities of these methods. Multiple tests have been performed, replicating conditions found on Earth and Mars, and it has been checked how the algorithm behaves in those scenarios.

In the section concerning the MPC with the Minimum Principle method as the optimisation tool, the configuration parameters for the simulations are described in the tables at the beginning. In the simulations run with MPC and a convex optimiser, several scenarios were considered. The launchers utilised to obtain the results were a customised simple rocket, called "Dummy Rocket", and the first stage of SpaceX's Falcon 9 rocket. These flew over the Earth and Mars in the simulations performed and the parameters used are shown in tables 5.1 and 5.2.

<table>
<thead>
<tr>
<th>Rockets</th>
<th>Dummy Rocket</th>
<th>Falcon 9 1st Stage</th>
</tr>
</thead>
<tbody>
<tr>
<td>N. Engines</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>Max. Thrust [kN]</td>
<td>845</td>
<td>845</td>
</tr>
<tr>
<td>Specific Impulse [s]</td>
<td>311</td>
<td>311</td>
</tr>
<tr>
<td>Wet Mass [t]</td>
<td>20</td>
<td>395.7</td>
</tr>
<tr>
<td>Dry Mass [t]</td>
<td>8</td>
<td>25.6</td>
</tr>
<tr>
<td>Length [m]</td>
<td>25</td>
<td>41.2</td>
</tr>
<tr>
<td>Diameter [m]</td>
<td>2</td>
<td>3.7</td>
</tr>
</tbody>
</table>

Table 5.1: Rocket parameters used in the simulations.
Table 5.2: Environment parameters used in the simulations. Data from [34] and [35]

Two computers were used to execute the simulations, one with an Intel i7-4702MQ CPU with 4 cores at 2.2 GHZ and 16 GB of RAM, and the other with an Inter i3-6100U CPU at 2.30 GHz with 4 cores and 32 GB of RAM. Although this could affect the results regarding the execution time, it is purely statistical and if there is any comparison it is done between execution times in the same machine. For the numerical approach in the Minimum Principle method, the function fmincon() has been called, while for the Convex Optimisation the third-party software libraries YALMIP and SeDuMi were used. The version of the code developed that has been utilised in the simulations is 0.1.3, as it is the latest stable version.

5.1 MPC with Minimum Principle

The first tests to analyse are focused on the performance of the Model Predictive Control algorithm with an optimiser implementing the Minimum Principle method. These are based on the models developed in Sections 3.1.5 and 3.1.6. In addition, the selected parameters for both simulations replicate Earth conditions without aerodynamic forces.

5.1.1 1 Dimension Analytical Approach

The analytical approach for the 1-dimensional model is assessed through a simple scenario were the rocket initiates the simulation at 10 kilometres of altitude and has to land with 0 velocity. This rocket is supposed to weight 55 tonnes and is affected by the gravitational pull of the Earth, indicated in Tables 5.3 and 5.4. They also reflect the configuration regarding the MPC algorithm. For this scenario, the horizon is set to 30 seconds, which
5.1. MPC WITH MINIMUM PRINCIPLE

should be enough for the mission, and the $\Delta t$ is chosen to be 0.1 for a proper smoothness of the control output. The simulation model in the $\text{StatePropagator()}$ function is similar to the one used in the definition of the optimisation model, with the variation that the mass is not constant, but changes over time.

<table>
<thead>
<tr>
<th>Gravity $[m/s^2]$</th>
<th>Specific Impulse [s]</th>
<th>N Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.81</td>
<td>311</td>
<td>2000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MPC Horizon [s]</th>
<th>MPC $\Delta t$ [s]</th>
<th>Sim $\Delta t$ [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>0.1</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Table 5.3: Constant values for the analytical and numerical Minimum Principle problems.

<table>
<thead>
<tr>
<th>$z_0$ [m]</th>
<th>$v_0$ [m/s]</th>
<th>$m_0$ [Kg]</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>0</td>
<td>55000</td>
</tr>
</tbody>
</table>

Table 5.4: Initial state values for the Minimum Principle problem in 1 Dimension.

The outcome shown in Figure 5.1 manifests an issue with this model. Even though it is a simplification, the RLV goes underground for a short period of time. This is due to the fact that the constraint on the positive altitude is not considered in the definition of the Hamiltonian. Therefore, the optimiser understands that negative values in this variable are allowed and takes advantage of it.

The altitude oscillation can be also explained by how the problem is set. The chosen horizon $t_h$ seems to be sufficiently short to obligate the controller to push the rocket downwards, as the thrust plot reflects in the initial seconds. This generates the overshoot in the altitude and velocity variables, as the second one seems to be positive when the altitude is below zero.

5.1.2 2 Dimension Numerical Approach

The 2-dimensional model is analysed in similar initial conditions. The configuration of the previous problem, in Table 5.3, is also applied in this case. In addition, the initial conditions are analogous, with the addition of the $x_0$ position at a distance of 3000 metres, as it is seen in Table 5.5. Besides, the only difference between the model in the optimiser
and the model in the simulation is the change of mass, which only occurs in the latter, as in the previous scenario.

<table>
<thead>
<tr>
<th>$x_0$ [m]</th>
<th>$z_0$ [m]</th>
<th>$V_{x0}$ [m/s]</th>
<th>$V_{z0}$ [m/s]</th>
<th>$m_0$ [Kg]</th>
</tr>
</thead>
<tbody>
<tr>
<td>3000</td>
<td>10000</td>
<td>0</td>
<td>0</td>
<td>55000</td>
</tr>
</tbody>
</table>

Table 5.5: Initial state values for the Minimum Principle problem in 2 Dimensions.

For the numerical approach, it is fundamental to remark the importance of the initial guess for the lambdas, applied in this case in the \texttt{fmincon()} numerical function, for an acceptable performance. They define how fast it will converge to a pseudo-optimal solution, if it does. In the case at hands, the initial guesses are based on the 1-dimensional model. The speculated values for $\lambda_{x0}$ and $\lambda_{z0}$ are obtained from the equation of the $\lambda_{z0}$ of the analytical model, and the values for $\lambda_{Vx0}$ and $\lambda_{Vz0}$ are given by the analytical $\lambda_{V0}$ equation.

The behaviour of the rocket is presented in Figure 5.12. The issue regarding the negative altitude is detected again, but it is normal as the problem was conceived unconstrained. It is also reflected on the value of the thrust magnitude and direction, $\alpha$, which has a large peak at the beginning of the simulation, i.e. it is not limited.
5.2 MPC WITH RECEDING HORIZON AND CONVEX OPTIMISER

On the other side, a curious feature is spotted in the position and velocity graphs over time (the 4 plots on the left side of the figure). The reusable launcher seems to stay close to the surface for about 100 seconds before it actually lands. It is explained by the strict final condition imposed in the simulation, where the maximum distance to accept the RLV as landed is up to 0.25 metres, and a maximum of 0.25 metres per second of velocity. Furthermore, the MPC utilised in the simulation, which manages a receding horizon, implies that the controller will behave slowly in time.

5.2 MPC with Receding Horizon and Convex Optimiser

Before presenting the results with the proposed MPC algorithm with decreasing horizon, it is necessary to show an outcome for a simulation with the classical MPC algorithm and a convex optimiser. In this case, the horizon is left constant. In Figure 5.13 the values presented describe how the classical MPC algorithm performed for a landing scenario with the Dummy Rocket and an initial state vector \( \mathbf{s_t} = (1000, 1000, 0, 0) \). Green values show how the MPC algorithm controls the vehicle until the time \( t \) reaches the horizon value at the beginning of the simulation. That is the time when the rocket was supposed to land. Nonetheless, the last output of the optimiser, represented in red, is a control signal to touchdown on the desired state after \( t_h \) seconds. This implies, and has been tested during the project, that no matter how long the simulation runs, the rocket will never approach the desired landing position.

The problematic particularity of this method lays on the fact that the terminal time is always in the future. As the horizon does not change, the launcher will always pretend to land \( t_h \) seconds after the current position and, consequently, will be flying over the ground forever. The receding horizon algorithm behaves well when the system is slow, which is totally the opposite to what happens for a rocket.
CHAPTER 5. SIMULATIONS AND RESULTS

5.3 MPC with Decreasing Horizon and Convex Optimiser

In this section, multiple real scenarios are put into practice, trying to validate the implementation of the proposed algorithm. Making use of the rockets and planets described at the beginning of this chapter, various cases are studied. These are divided into launch, landing, guidance and Monte Carlo analysis, where random initial positions are generated within a maximum range. Afterwards, a detailed examination is done about the effect of the horizon $t_h$ and the Time Percentage Update Factor (TPUF) in the suggested method. The solutions are obtained with version 0.1.3 of the implementation. This implies that the model used was the 2-dimensional Point Mass model with thrust components.

<table>
<thead>
<tr>
<th>Case</th>
<th>MPC Horizon [s]</th>
<th>MPC $\Delta t$ [s]</th>
<th>Max Sim Time [s]</th>
<th>Sim $\Delta t$ [s]</th>
<th>TPUF</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>150</td>
<td>0.1</td>
<td>200</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>B</td>
<td>100</td>
<td>0.1</td>
<td>200</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>C</td>
<td>130</td>
<td>0.1</td>
<td>200</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>D-Launch</td>
<td>150</td>
<td>0.1</td>
<td>200</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>D-Landing</td>
<td>250</td>
<td>0.1</td>
<td>500</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>E-Launch</td>
<td>23</td>
<td>0.1</td>
<td>40</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>E-Landing</td>
<td>150</td>
<td>0.1</td>
<td>500</td>
<td>1</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Table 5.6: Configuration values for Cases A to E.

The initial configuration of the proposed simulations is detailed in Table 5.6. Apart from the horizon and the TPUF parameters, which will be discussed in more detail in the following cases, it is also important to state the effect of the MPC $\Delta t$ in the solution: it has to be small enough to allow the controller to generate a smooth control signal to overcome the possible deviations in the real simulation. If the $\Delta t$ is large, it will create instabilities in the control.

5.3.1 Case A: Launch of Falcon 9 1st stage on Earth

The first case refers to the launch of a Falcon 9 1st stage, where the conditions of a real launch are replicated to some extent. The goal state is inspired by the scenario seen in Figure 2.3. In that image, the 1st and 2nd stages of the vehicle are separated at about 73 km
above the surface, and that is why the objective is at 80km of altitude in the simulation, Table 5.7.

A MPC Horizon of 150 seconds was selected for this sketch. It is not aleatory, but a previous study must be done to find a suitable horizon. Consequently, if it is not a proper one, the algorithm may not be able to solve the problem. Apart from it, Table 5.6 shows the other configuration parameters. The most important of those is the TPUF, as it is explained in Section 5.3.8. The typical value that is used throughout the simulations is 0.2, which seems to be a reasonable trade between the MPC thrust optimisation performance and the execution time.

<table>
<thead>
<tr>
<th>Case A</th>
<th>$x$ [m]</th>
<th>$z$ [m]</th>
<th>$V_x$[m/s]</th>
<th>$V_z$[m/s]</th>
<th>$m$[kg]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial state</td>
<td>20000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>395700</td>
</tr>
<tr>
<td>Goal state</td>
<td>19000</td>
<td>80000</td>
<td>-100</td>
<td>1000</td>
<td>-</td>
</tr>
<tr>
<td>Final state MPC</td>
<td>18842</td>
<td>81633</td>
<td>-76.8</td>
<td>851</td>
<td>154659</td>
</tr>
</tbody>
</table>

Table 5.7: Initial, final and goal state values for the Case A simulation: Launch of a Falcon 9 1st stage on Earth.

Figure 5.2 displays the path followed by the rocket during flight. The thrust seems to be aligned reasonably with the velocity vector for this particular case, even though the implementation of the angle of attack constraint was not successful for the reasons stated in Section 4.1.2.3. The behaviour at the start appears to be similar to a bang-bang, although as the final constraint in the velocity components was relaxed in the problem, the final bang is not executed. The velocity at the end of the flight for the convex optimiser has a minimum value but not a maximum.

The state variables and the thrust plots shown in Figure 5.14 reflect how they change over time. It is interesting to see the final values of the position and velocity. The first one reaches its goal, yet the rocket does it with a decreasing velocity. The most probable reason for this is a conjunction of the problem becoming infeasible in the last moments of flight due to the horizon effect and the environmental forces, which some are not accounted in the convex optimisation, i.e. the aerodynamic forces. Moreover, the peaks visualised in the thrust graphics are due to the update in the horizon $t_h$. As $t_h$ decreases
as a result of the application of the TPUF, the rocket has to increase its speed to reach the desired state in a shorter time.

5.3.2 Case B: Landing of Falcon 9 1st stage on Earth

After launching a Falcon 9 1st stage, the next step is to measure the performance of the MPC algorithm in the descent phase. This time the horizon is 100 seconds and the initial state is at 20 kilometres of altitude and 10 kilometres away horizontally.

In this scenario, a comparison is being proposed between the initial control signal and state vectors from the convex optimiser and the actual values after the MPC execution. The values displayed in Figure 5.15 present a smooth curve and an ideal trajectory finished at the landing point with zero velocity. This corresponds to the ideal case in which the convex optimiser and the simulation models are the same and the horizon decreases at
the same pace as the simulation passes. Nonetheless, the real scenario behaves in a completely different way. In Figure 5.16, the altitude changes quicker than the position in x, implying that the rocket encounters more resistance in that direction. Obviously, in the z axis the acceleration due to the gravitational force pushes downwards, which contributes to a fast movement towards the ground. Additionally, the drag force is now taken into account acting on the dynamics of the RLV.

<table>
<thead>
<tr>
<th>Case B</th>
<th>x [m]</th>
<th>z [m]</th>
<th>Vx[m/s]</th>
<th>Vz[m/s]</th>
<th>m[kg]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial state</td>
<td>10000</td>
<td>20000</td>
<td>0</td>
<td>0</td>
<td>395700</td>
</tr>
<tr>
<td>Goal state</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>Final state MPC</td>
<td>0</td>
<td>0.68</td>
<td>0</td>
<td>-1.60</td>
<td>218603</td>
</tr>
</tbody>
</table>

Table 5.8: Initial, final and goal state values for the Case B simulation: Landing of a Falcon 9 1st stage on Earth.

The effect of the TPUF on the performance of the MPC algorithm is likewise considered to be one of the origins of the “flotation” phase during the last 100 seconds of the MPC simulation. It is most probable that the stable thrust around 300 kN in the 3rd graphic of 5.16 could be in some extent saved if the horizon is managed in another manner during the last part of the simulation.

Moreover, if Figures 5.3 and 5.4 are analysed, it is easy to see how the trajectory diverges from the initial prediction. The first one is rapidly directed towards the goal while the MPC path takes a longer approach. Even though, the MPC with decreasing horizon succeeds in its mission consuming a similar amount of propellant and landing at less than a metre from the landing pad with a vertical velocity of 1.6 m/s, see Table 5.8.
Figure 5.3: Landing path followed by the Falcon 9 1st stage in case B with the thrust and velocity vectors. One execution of the convex optimiser.

5.3.3 Case C: Landing of Falcon 9 1st stage on Mars

The second landing test defines totally opposite conditions. The idea behind this simulation is to conduct a pinpoint landing on planet Mars when the rocket is arriving from an interplanetary journey. The initial position is at 50 km of altitude and 50 of horizontal distance and the vehicle is supposed to have been decelerated prior to the atmospheric entry. The velocities in this circumstances are -1.5 km/s in the horizontal axis and -0.3 km/s in the vertical. The $t_h$ configured is 130 seconds.

<table>
<thead>
<tr>
<th>Case C</th>
<th>$x$ [m]</th>
<th>$z$ [m]</th>
<th>$V_x$ [m/s]</th>
<th>$V_z$ [m/s]</th>
<th>$m$ [kg]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial state</td>
<td>50000</td>
<td>50000</td>
<td>-1500</td>
<td>-300</td>
<td>395700</td>
</tr>
<tr>
<td>Goal state</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>Final state MPC</td>
<td>0</td>
<td>0.3</td>
<td>0</td>
<td>0</td>
<td>39857</td>
</tr>
</tbody>
</table>

Table 5.9: Initial, final and goal state values for the Case C simulation: Landing of the Falcon 9 on Mars.
In this case, the output of the convex optimiser for the initial scenario is analysed to verify if some of the most problematic constraints are met. At first sight in Figure 5.5, the thrust direction $\alpha$ follows a nice progression along the flight. Furthermore, its magnitude is steadily changing over time, which can be contrasted in Figure 5.17. Certainly, the altitude of the RLV is always over the ground and the desired final conditions described in Table 5.9 seemed to be achieved. Additionally, the thrust vector is always kept upwards, which a $T_z$ strictly positive. Nevertheless, the constraint on the angle of attack is not upheld and requires further efforts to understand why YALMIP does not accept this restriction.

Focusing on the results for the MPC algorithm, the rocket does achieve its mission (Figure 5.6). However, the complication of the floating phase does appear. It is important
Figure 5.5: Landing path followed by the Falcon 9 1\textsuperscript{st} stage in case C with the thrust and velocity vectors. One execution of the convex optimiser.

to notice the peak of thrust in the z-axis in Figure 5.18 which occurs when the vehicle is closer to the ground. Associating it to the behaviour of the velocities over time, the controller gives the impression to be focused on decreasing the high horizontal speed in the first half of the simulation, and rapidly increasing the vertical thrust when the rocket gets closer to the surface with high $V_z$. 
5.3. MPC WITH DECREASING HORIZON AND CONVEX OPTIMISER

5.3.4 Case D: Launch and Recovery of Falcon 9 1st stage on Earth

Unifying both phases, launch and landing, the guidance algorithm can be completed with a coastal phase to reach the apogee of the flight. This scenario mixes both parts to represent a launch of a Falcon 9 1st stage and its recovery. The launcher is originally placed 20 kilometres away from the landing pad, and has to reach an altitude of 80 kilometres with a velocity of 1 kilometre per second. Once it passes the launch goal state, it turns off the engine and performs a free flight until the apogee. This is supposed to replicate the conditions of a stage separation. Later on, the guidance algorithm will call the MPC controller again and safely land on the desired spot. All these steps and the output of the simulation are represented in Table 5.10.

The horizon during the simulation is different for both phases. The launch has a $t_h$ of 150 seconds to carry out the ascension in the shorter possible time, and the landing counts
<table>
<thead>
<tr>
<th>Case D</th>
<th>$x\ [m]$</th>
<th>$z\ [m]$</th>
<th>$V_x\ [m/s]$</th>
<th>$V_z\ [m/s]$</th>
<th>$m\ [kg]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch Initial state</td>
<td>20000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>395700</td>
</tr>
<tr>
<td>Launch Goal state</td>
<td>19000</td>
<td>80000</td>
<td>-100</td>
<td>1000</td>
<td>-</td>
</tr>
<tr>
<td>Launch Final state MPC</td>
<td>18890</td>
<td>81442</td>
<td>-71.5</td>
<td>711.8</td>
<td>156290</td>
</tr>
<tr>
<td>Landing Initial state</td>
<td>13697</td>
<td>107263</td>
<td>-71.5</td>
<td>0</td>
<td>156290</td>
</tr>
<tr>
<td>Landing Goal state</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>Landing Final state MPC</td>
<td>0</td>
<td>0.4</td>
<td>0</td>
<td>-1.5</td>
<td>48373</td>
</tr>
</tbody>
</table>

Table 5.10: Initial, final and goal state values for different phases of the Case D simulation: Launch and Recovery of Falcon 9 1st stage on Earth.

on a $t_h$ of 250 seconds for a larger path (about 20 km more).

Figures 5.7 and 5.19 represent the path that the rocket covers to perform the complete manoeuvre and how the state and control variables change over time. The ascend phase shows a smooth trajectory, although it does not achieve the desired final velocity because of the $t_h$ complication, with only up to 711 m/s instead of 1000 m/s. During the descend the convex optimiser seems not to be able to solve the landing problem until the RLV is about 20 km above the ground. This can be seen relating the $V_z$ and $z$ graphs in Figure 5.19 as the lower peak in the velocity plot corresponds in time with that altitude. Additionally, it is indicated by the rapid increase of $T_z$. The final part of the flight is occupied by the ”floating” phase, as it is seen in both figures (specially in the altitude graph).

An important final remark from the information in Table 5.10 and Figure 5.19 is that at the end of the simulation, there are still almost 50 tonnes of rocket mass, which is almost the double of the dry mass. Consequently, the adequacy of the MPC algorithm for this type of control is reinforced and encourages more research to build up more complicated models in the future.
5.3. MPC WITH DECREASING HORIZON AND CONVEX OPTIMISER

5.3.5 Case E: Launch and Recovery of Dummy Rocket on Mars

The second simulation regarding a full flight replicates a Hopper manoeuvre on Mars, performed by the Dummy Rocket. The RLV moves from 10 kilometres of distance passing through an apogee at 11 kilometres of altitude. The path of the Dummy Rocket in Figure 5.8 is in appearance similar to the previous scenario. The launcher performs the ascent and descent phases in an analogous manner and demonstrates the landing behaviour previously discussed. Furthermore, plots in Figure 5.20 reflect an akin performance in the control of the velocities and the thrust peaks. However, the rocket mass availability was dramatically reduced compared to case D. In this sketch, there are only 12 tonnes of propellant, and finding an optimal solution was more challenging.

In Table 5.11 there are several particularities to mention. The effect of the aerodynamic drag can be seen in the difference between the velocity in the x axis at the final
Case E

<table>
<thead>
<tr>
<th></th>
<th>$x [m]$</th>
<th>$z [m]$</th>
<th>$V_x [m/s]$</th>
<th>$V_z [m/s]$</th>
<th>$m [kg]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch Initial state</td>
<td>10000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>20000</td>
</tr>
<tr>
<td>Launch Goal state</td>
<td>6000</td>
<td>6000</td>
<td>-50</td>
<td>150</td>
<td>-</td>
</tr>
<tr>
<td>Launch Final state MPC</td>
<td>4577</td>
<td>7484</td>
<td>-158</td>
<td>169</td>
<td>14506</td>
</tr>
<tr>
<td>Landing Initial state</td>
<td>-2442</td>
<td>11274</td>
<td>-153.8</td>
<td>0</td>
<td>14506</td>
</tr>
<tr>
<td>Landing Goal state</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>Landing Final state MPC</td>
<td>0</td>
<td>0</td>
<td>0.18</td>
<td>-10.5</td>
<td>8033</td>
</tr>
</tbody>
</table>

Table 5.11: Initial, final and goal state values for different phases of the Case E simulation: Launch and Recovery of Dummy Rocket on Mars.

state of the launch phase and the beginning of the landing phase. This reduction would not happened if there were no aerodynamic forces, as the engine is turned off during the coastal phase. In case D, this velocity was maintained, yet the reason why is found in the fact that the coastal phase is performed in the outer part of the atmosphere, even partially exo-atmospheric, which implies that these forces are almost neglected.

The final velocities in the launch problem are higher than the defined in the goal state. This is in accordance with the definition of the optimisation problem for the launch phase, where the constraint in the targeted velocities was relaxed to a minimum value. Nonetheless, if it is compared with the launch and recovery cases A and D, which occur on Earth, they do not achieve the predefined minimum value, especially for the velocity in the z axis. This, again, is due to the effect of the aerodynamic forces and points out the importance of these in the performance of the flight.
5.3. **MPC WITH DECREASING HORIZON AND CONVEX OPTIMISER**

5.3.6 Case MC1: Monte Carlo Test for Landing the Falcon 9 1\textsuperscript{st} stage on Earth

At last, a pair of Monte Carlo simulations were carried out to investigate the robustness of the MPC algorithm. In the first one, multiple landing scenarios of a Falcon 9 1\textsuperscript{st} stage were carried out, in a space of 10 km by 10 km from the landing site. The initial velocity in each component was up to 50 m/s. The MPC horizon was calculated depending on the initial position. For the maximum possible distance, i.e. at an initial position equal to (10000, 10000), the \( t_h \) was 120 seconds. But if the starting position is closer to the landing target, the proportional \( t_h \) is assigned. Likewise, the maximum simulation time was obtained. The MPC and simulation \( \Delta t \) were constant at 0.1 and 1 seconds respectively. This information is summarised in Table 5.12.
Table 5.12: Configuration values for the Monte Carlo MC1 simulation: Landing of the Falcon 9 1st on Earth.

<table>
<thead>
<tr>
<th>Configuration Value</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max Position Component [m]</td>
<td>10000</td>
</tr>
<tr>
<td>Max Velocity Component [m/s]</td>
<td>50</td>
</tr>
<tr>
<td>Max MPC Horizon [s]</td>
<td>120</td>
</tr>
<tr>
<td>MPC ∆t [s]</td>
<td>0.1</td>
</tr>
<tr>
<td>Max Sim Time [s]</td>
<td>300</td>
</tr>
<tr>
<td>Sim ∆t [s]</td>
<td>1</td>
</tr>
<tr>
<td>Num Iterations</td>
<td>250</td>
</tr>
</tbody>
</table>

The output of the MC1 test strengthens the data observed and the interpretation given in cases A to E. The test was run for 250 iterations, where 235 were feasible. Understanding why the other 15 are not achievable is key to enhance the performance of the predictive control. The results displayed in Figure 5.9 contrast the feasibility of each of the initial state vectors with their elements.

A first look into the initial position subfigure already reveals a pattern in the acquired data. All the infeasible state vectors are close to the touchdown position. Accordingly, there must be a relation between the initial state vector and the feasibility of the problem at hand. This connection is presumed to be the same as in the previous cases, which is that when the rocket approaches the final part of the flight, the horizon must be properly selected or otherwise it will become unfeasible (if $t_h$ is short) or will reflect a "flotation" phase (if $t_h$ is large).

At first glance, the viability with respect to the velocity part of the starting state vector does not seem to follow any correlation. The 3rd and 4th subplots of Figure 5.9 only show a trend in their unfeasible data in relation to $x$ and $z$ position axes. Moreover, that trend is seen when those values are close to 0. Nevertheless, the trajectories covered by the Falcon 9 seen in Figure 5.21 disclose a possible correspondence between unfeasibility and initial velocity: most of the red paths tend to go outwards in the $x$ direction, which added to the short distance makes a pretty hard problem to solve.

A simple statistical analysis of the feasible solutions has been carried out in Table 5.13. Comparing the median values relative to minimum and maximum values of different final
Figure 5.9: Initial positions and velocities for the Falcon 9 1\textsuperscript{st} stage in the Monte Carlo Test MC1. Blue indicates that the problem is feasible, red that is unfeasible.

<table>
<thead>
<tr>
<th>Type of Value</th>
<th>Minimum</th>
<th>Median</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance to Goal [m]</td>
<td>0</td>
<td>0.74</td>
<td>1.99</td>
</tr>
<tr>
<td>Touchdown Velocity [m/s]</td>
<td>0.27</td>
<td>1.56</td>
<td>1.99</td>
</tr>
<tr>
<td>Final Mass [kg]</td>
<td>164755</td>
<td>233256</td>
<td>337571</td>
</tr>
</tbody>
</table>

Table 5.13: Statistical data for the feasible solutions in the Monte Carlo MC1 simulation: Landing of the Falcon 9 1\textsuperscript{st} on Earth.

values, it is possible to obtain the typical landing manoeuvre for the configuration of the MC1 test. When a problem is feasible, the landing position is regularly closer than 1 metre away from the goal. The touchdown velocity seems to be closer to the higher end of the feasible solutions. The maximum errors to consider a problem feasible were 2 metres in the distance between the final and the targeted positions and 2 metres per second in the velocity. Even though the final mass is substantially dependent on the initial conditions, it is engaging to see that the minimum value is way higher than the dry mass of the rocket.
5.3.7 Case MC2: Monte Carlo Test for Landing the Dummy Rocket on Mars

The second Monte Carlo simulation checks the performance of the MPC algorithm in a harder scenario. For the same maximum distance from the landing site, the initial magnitude of the velocity can be higher, up to 500 $m/s$, and the launcher has a larger dry mass/wet mass ratio\(^1\) compared to the configuration of MC1. Consequently, the expected results should be worse.

For the same MPC $t_h$, see Table 5.14 and simulation time, the test ran this time 500 different initial positions and only 186 were feasible. The initial position subplot in Figure 5.10 shows the distribution of these in comparison to the unfeasible problems. This time, there is no correlation between these and the feasibility of the paths.

<table>
<thead>
<tr>
<th>Max Position Component [m]</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max Velocity Component [m/s]</td>
<td>500</td>
</tr>
<tr>
<td>Max MPC Horizon [s]</td>
<td>120</td>
</tr>
<tr>
<td>MPC $\Delta t$ [s]</td>
<td>0.1</td>
</tr>
<tr>
<td>Max Sim Time [s]</td>
<td>300</td>
</tr>
<tr>
<td>Sim $\Delta t$ [s]</td>
<td>1</td>
</tr>
<tr>
<td>Num Iterations</td>
<td>500</td>
</tr>
</tbody>
</table>

Table 5.14: Configuration values for the Monte Carlo MC2 simulation: Landing of the Dummy Rocket on Mars.

Nonetheless, the other 3 subplots reveal that the initial velocity vector clearly affects the results. In the top right subfigure there is a range between $-300$ and $200$ $m/s$ in the $z$ component of the velocity that delimits the success of the simulation. Outside those boundaries, the starting conditions appear to be complicated enough to convert the problem unfeasible. Furthermore, the valid range in the $x$ axis is about $-400$ and $400$ $m/s$, which is more relaxed. Likewise, the bottom subplots reinforce the velocity effect theory: the simulations are feasible when its initial velocity magnitude is less than $400$ $m/s$ and especially the velocity component $V_z$ is negative.

In Figure 5.22 it can be seen how the algorithm tries to reach the desired landing point.

---

\(^1\)This ratio is 0.065 for the Falcon 9 1\(^{st}\) stage and 0.4 for the Dummy Rocket.
5.3. MPC WITH DECREASING HORIZON AND CONVEX OPTIMISER

Figure 5.10: Initial positions and velocities for the Dummy Rocket in the Monte Carlo Test MC2. Blue indicates that the problem is feasible, red that is unfeasible.

for a high portion of initial values that led to an infeasible problem. What is probably happening there is that the rocket runs out of fuel before the touchdown, which rises the importance of a good mass ratio in the RLV utilised.

<table>
<thead>
<tr>
<th>Type of Value</th>
<th>Minimum</th>
<th>Median</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance to Goal [m]</td>
<td>0.25</td>
<td>1.23</td>
<td>1.99</td>
</tr>
<tr>
<td>Touchdown Velocity [m/s]</td>
<td>0.78</td>
<td>1.50</td>
<td>1.99</td>
</tr>
<tr>
<td>Final Mass [kg]</td>
<td>8028</td>
<td>9657</td>
<td>14952</td>
</tr>
</tbody>
</table>

Table 5.15: Statistical data for the feasible solutions in the Monte Carlo MC2 simulation: Landing of the Dummy Rocket on Mars.

The statistical analysis displays an alike outcome compared to the data of MC1 for the same range of acceptance, i.e. 2 metres of distance and 2 metres per second for the final velocity. The minimum and median distances are larger although the values are reasonable for the distances used in this type of problems. Even so, the remarkable change is centred in the final mass. The minimum and median final masses for the feasible problems are critically close to the dry mass of the vehicle. In addition, it has been checked that increasing the acceptance margin does not rises dramatically the number of feasible simulations but the opposite. For large margins, i.e. kilometres in the distance
and hundreds of metres per second in the velocity errors, the number of feasible problem
increases in a couple of dozens.

5.3.8 **Effect of the Time Percentage Update Factor**

The last variable to test and comprehend is the TPUF of the proposed MPC algorithm. It is discussed how it affects the performance of the controller. In Figure 5.11 5 different cases are shown, with the TPUF values from 0.1 to 1, and Table 5.16 contains the related variables performance. The environment and rocket used in this test were the Earth and the Dummy Rocket.

![Figure 5.11: Flight position for different time percentage factors.](image)

The trajectory followed by the rocket is clearly affected by this parameter. As the TPUF is larger, the launcher gets closer to the surface because it has "to wait" more time before landing. Recalling what was explained in previous sections, the initial horizon will be maintained for a longer time when this coefficient is larger and, thus, the horizon used when the vehicle is reaching its target will be larger too.

If the actual values for the 5 simulations are analysed, the algorithm has a better final
### 5.3. MPC WITH DECREASING HORIZON AND CONVEX OPTIMISER

<table>
<thead>
<tr>
<th>TPUF value</th>
<th>0.1</th>
<th>0.2</th>
<th>0.5</th>
<th>0.8</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Final Position $x$ [m]</td>
<td>-0.07</td>
<td>-0.12</td>
<td>-0.27</td>
<td>-1.08</td>
<td>4.72</td>
</tr>
<tr>
<td>Final Position $z$ [m]</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Final Velocity $V_x$ [m/s]</td>
<td>0.09</td>
<td>-0.60</td>
<td>-2.20</td>
<td>-0.50</td>
<td>-2.27</td>
</tr>
<tr>
<td>Final Velocity $V_z$ [m/s]</td>
<td>-1.19</td>
<td>-1.61</td>
<td>-2.70</td>
<td>-2.62</td>
<td>-2.82</td>
</tr>
<tr>
<td>Final Mass [kg]</td>
<td>17841</td>
<td>17933</td>
<td>17769</td>
<td>17470</td>
<td>17028</td>
</tr>
<tr>
<td>Execution Time [s]</td>
<td>101</td>
<td>58</td>
<td>39</td>
<td>29</td>
<td>29</td>
</tr>
</tbody>
</table>

Table 5.16: Final values for the state vector, the mass and the execution time for the different MPC TPUF values.

value performance in general terms when the TPUF is smaller. The $x$ is nearer to the desired landing position and the final velocities are higher when the coefficient is closer to 1.0. Another critical parameter that indicates if the performance of the algorithm is close to optimal is the final mass, which marks how much fuel has been consumed. As it can be seen, there is a remarkable difference between the two smallest coefficients and the last one, with almost a ton of propellant saved in the first cases. Nevertheless, the execution time was longer with the first conditions, but it is due to how YALMIP works and how the algorithm is implemented.

These results strengthen the reason of why the receding horizon is not suitable for the guidance problem of a space launcher. The outcome is worst when the time factor is higher, which means that the initial horizon is maintained for a longer time, i.e. closer to the receding horizon concept.
Figure 5.12: Position, velocities and thrust over time for the Minimum Principle problem in 2D.
Figure 5.13: Position, velocities and thrust plots for the MPC with receding horizon. Green plots represent the state of the rocket over time, while the red graphs are the last output of the optimal convex controller.
Figure 5.14: Position, velocities and thrust plots for the Case A: launch of the Falcon 9 1st stage at Earth.
Figure 5.15: Position, velocities and thrust plots for the Case B: landing of the Falcon 9's stage at Earth. One execution of the convex optimiser.
Figure 5.16: Position, velocities and thrust plots for the Case B: landing of the Falcon 9 1st stage at Earth. MPC simulation.
Figure 5.17: Position, velocities and thrust plots for the Case C: landing of the Falcon 9 1st stage at Mars. One execution of the convex optimiser.
Figure 5.18: Position, velocities and thrust plots for the Case C: landing of the Falcon 9 1st stage at Mars. MPC simulation.
Figure 5.19: Position, velocities and thrust plots for the Case D: launch and recovery of the Falcon 9 first stage at Earth.
Figure 5.20: Position, velocities and thrust plots for the Case E: launch and recovery of the Dummy Rocket at Mars.
Figure 5.21: Paths in the Monte Carlo Test MC1 followed by the Falcon 9 1st stage at Earth. Blue paths are feasible problems and red mark the unfeasible ones.
Figure 5.22: Paths in the Monte Carlo Test MC2 followed by the Dummy Rocket at Mars. Blue paths are feasible problems and red mark the unfeasible ones.
Chapter 6

Conclusions and Further Work

In this thesis, a comprehensive analysis of the Model Predictive Control algorithm and different optimise control methods was conducted. Since the beginning, when the first simple models were proved with the Minimum Principle technique, until the last weeks of the thesis with the convex optimisation of a rigid body model, the fundamental knowledge of optimal guidance and control of reusable launchers has been built.

Many obstacles were encountered during the development of the guidance scheme and the software implementation. YALMIP is a great software library to interact with different optimisers, but there are certain aspects that seem to be very opaque. The implementation of several constraints delayed the progress of the thesis and at the end the implementation of the real time setup had to be discarded. In addition, Chapter 3 presents the equations of motion of several RLVs, although the development of the Rigid Body model was not finish in time to carry out the simulations with it.

Practically all the fundamental questions stated in the introductory chapter were answered. Two different optimal control approaches were applied to control a rocket model. Clearly, the Convex Optimisation method is more suitable than the Minimum Principle for the problem at hand. The equations of motion based on the costates can become very complicated for a simple model, needing a numerical solver to optimise the problem, which makes it totally unrealistic for a real implementation. Furthermore, the Convex
Optimisation method benefits of state of the art algorithms, like the interior point method, that solve the problem in polynomial time. Nonetheless, Convex Optimisation implies that all the elements taken into account for the embedded model must be convex, which can be very hard to achieve for certain cases. Even though, it is fast enough to be implemented in a real scenario.

Certainly, the Model Predictive Control seems to be very adequate for guidance and control of space launchers. The results shown in Chapter 5 reinforced this idea and present a base for further development in the future. A detailed cluster of scenarios and simulations has proved that the proposed MPC algorithm with decreasing horizon has the capabilities to carry out not only a vertical descend phase and landing of a RLV, but also a vertical launch and full flight of the vehicle passing through the coastal phase were the vehicle reaches the apogee of its trajectory. There are still plenty of tasks to be carried out in order to enhance the control of the rockets, but the Monte Carlo simulations demonstrated that there is future in this technology and that it could be potentially adopted to perform a real landing in the near future. These tasks include the investigation and resolution of the issue regarding the control at the end of the flight due to the problematic MPC horizon.

6.1 Further Work

For future development, there are certain areas that could be enhanced or implemented to cover a more generic and realistic scenario. The most critical aspects to proceed with the project are:

- Considering a free ignition time at the beginning of the landing phase. This can prevent the launcher to waste propellant in certain cases, and enhances its optimisation.

- Substituting the percentage time factor with a calculated time to target from the current distance and velocity, or a similar strategy to solve the issue seen in the simulations.
6.1. FURTHER WORK

• Trying different MATLAB libraries for convex optimisation and fix the Angle of attack constraint

• Implementing a 3-dimensional model and a thrust vector control with new constraints

• Adding a new constraint in the maximum velocity magnitude depending on the characteristics of the material used in the rocket, to prevent excessive heating.

• Finishing the RT implementation and hardware setup to verify that the MPC and Convex Optimisation algorithms are suitable for the guidance and control of real RLVs.
Appendix A

MATLAB README.md File

A.1 MPC Controller for Guidance and Control of a Reusable Vehicle

This project is part of the Individual Research Project (IRP) performed by Guillermo Zaragoza Prous for the MSc in Astronautics and Space Engineering at Cranfield University (UK) within the SpaceMaster programme. This project is supervised by Dr. Leonard Felicetti.

A.1.1 Version History

A.1.1.1 Version 0.1

Initial version of the code.

- Minimum principle: 1 dimensional point-mass analytical solution.
- Minimum principle: 2 dimensional point-mass numerical solution.
- Convex optimization: 2-dimensional point-mass YALMIP solution. Simple MPC implementation with receding horizon.
- MPC implementation + Convex Optimization: Point-mass dynamics with the landing problem up to 10 Km. MPC implementation with decreasing horizon.
A.1.1.2 Version 0.1.1

- MPC implementation + Convex Optimization: Change in the constraint of the maximum thrust to solve the issue with the limits of the workspace in YALMIP.

A.1.1.3 Version 0.1.2

- MPC implementation + Convex Optimization: Implementation of the launch phase.
- MPC implementation + Convex Optimization: Constraint on the rate of change of the thrust vector orientation added for Launch and Descent phases.
- MPC implementation + Convex Optimization: Constraint on the angle of attack added for the Launch and Descent phases. This constraint is commented and is not used as YALMIP and SeDuMi does not support it as it is proposed. It is a Work In Progress (WIP).
- MPC implementation + Convex Optimization: Constraint on the values of z and vz for the Launch phase. They can only increase.
- MPC implementation + Convex Optimization: Guidance simulation added. The launcher flies up to the position and velocity specified by Config2DLaunch. Then the simulation is run until the vehicle reaches apogee. Finally the launcher tries to land in the position given by Config2D.

A.1.1.4 Version 0.1.3

- MPC implementation + Convex Optimization: New environments and rockets.
- MPC implementation + Convex Optimization: Simulation model now includes the aerodynamic forces.
- MPC implementation + Convex Optimization: Constraint on the final value for the velocity components in the Launch problem is now relaxed to greater or equal to the desired final value.
A.1. MPC CONTROLLER FOR GUIDANCE AND CONTROL OF A REUSABLE VEHICLE

A.1.2 SW Dependencies

The MATLAB third-party tools used in the last version of the project (0.1.2) are: - YALMIP\(^1\) - SeDuMi\(^2\)

A.1.3 Authors

MSc student: Guillermo Zaragoza Prous, Cranfield University. Supervisor: Dr. Leonard Felicetti, Cranfield University.

\(^1\)<https://github.com/yalmip/YALMIP/releases>
\(^2\)<http://sedumi.ie.lehigh.edu/?page_id=58>
Appendix B

Teensy Pinout

Figure B.1: Teensy 4.1 pinout [26]
References


REFERENCES


REFERENCES


