DAS Departamento de Automação e Sistemas
CTC Centro Tecnológico
UFSC Universidade Federal de Santa Catarina

Analysis of a Model Predictive Impulsional Control for Time Variant Systems

Relatório submetido à Universidade Federal de Santa Catarina

como requisito para a aprovação na disciplina

DAS 5511: Projeto de Fim de Curso

Bruno Eduardo Benetti

Florianópolis, agosto de 2015

Resumo

Este trabalho apresenta uma estratégia de controle preditivo baseado em modelo para um sistema variante no tempo. O sistema em questão é formado por dois satélites que compõe o problema do Rendez-vous orbital. O problema do Rendez-vous orbital consiste em um satélite chamado alvo que é inerte e orbita ao redor da Terra e outro satélite chamado caçador que deve permanecer na vizinhança do alvo. O objetivo é então fornecer uma lei de controle que será aplicada nos atuadores do satélite caçador de modo que ele nunca saia de uma região determinada ao redor do alvo.

Esta lei de controle é impulsional por considerar que a dinâmica de mudança das velocidades é instantânea em relação à dinâmica do movimento orbital. O controle deve gerar uma trajetória que possui restrições dimensionais, além disso ele deve levar em conta a saturação dos atuadores e o nível de combustível no reservatório do satélite. É definido então um problema formal que fornece o conjunto de controles que satisfaz todas as restrições. O método de cálculo de controle consiste em achar um ponto na intersecção de dois conjuntos que definem uma trajetória admissível.

Devido a restrições computacionais (memória e processamento) e de tempo não é possível embarcar solvers de otimização nesses sistemas. Dadas estas restrições é necessário buscar um algorítimo rápido e leve que forneça os controles a serem aplicados. O algorítimo utilizado neste trabalho é o algorítimo de projeções alternadas que, apesar de não fornecer uma resposta ótima em termos de consumação, é capaz de fornecer um controle válido respeitando as restrições.

O trabalho então foca na análise do uso deste algorítimo como solução para o problema. Realiza-se uma análise da eficácia deste método de controle quanto à consumação de combustível, uma análise de factibilidade do problema e, por fim, uma análise numérica da parte iterativa do algorítimo.

Palavras Chave: Controle Preditivo Baseado em Modelo, Rendez-Vous Orbital, Algorítimo de Projeções Alternadas, Análise Numérica.

Abstract

Impulsive thrust corrections must be executed by spacecrafts to keep a desired trajectory during orbital rendezvous missions. Robust and simple algorithms are required to compute these corrections in order to overcome the limited performance of the spacecrafts computing devices. In this report it is analysed the use of the Alternating Projections Algorithm that was proposed as a solution to this problem. Initially, a representation of the spacecraft relative movement is given with the modelisation of the system. Afterwards, the control law and the modelisation of the problem's constraints using positive-definite symmetric matrices is presented in order to formulate the mathematical problem that is solved by the algorithm. The algorithm is then analysed to see its limitations and capacities. The results given are compared to the use of an open source optimization solver to carry out an optimality analysis. In the end a study of the feasibility of the problem and a numerical analysis of the algorithm is made.

Keywords: Model Predictive Control, Orbital RendezVous, Alternating Projections Algorithm, Numerical Analysis.

Acknowledgements

First of all I would like to thank my parents and sister for all the support they gave me even being a more than 9 thousand kilometres away. If it wasn't for their support and advice I would never cross the boundaries of my own country alone. Thanks Kelly for showing mom and dad that is possible to survive when we are far away from home!

In second place I would like to thank all my friends and, in special, my girlfriend for being there for me every night and every weekend to remember that there is something else in life besides my job.

In third place I would like to thank UFSC, ENSEEIHT and LAAS for being responsible of my professional success and everything I've learnt in engineering and mathematics.

I could never forget all the efforts of my Professors: Eduardo Camponogara for being my tutor since the begining of my journey in the university untill the last moments of the graduation course and Christophe Louembet for guiding me through the work in the aerospace field. You are certainly responsible for everything I conquered in this last years.

I would also like to thank CAPES and the CNRS for financing my travel and internship period and also to provide oportunities like this for young students to enter in the academic world.

Contents

1 Introduction **11**
- 1.1 Spacecraft Rendezvous . . . 11
- 1.2 Motivation and Objectives . . . 13
- 1.3 Organisation . . . 14

2 Spacecraft relative motion **15**
- 2.1 Introduction . . . 15
- 2.2 Dynamics of a spacecraft orbiting the Earth . . . 15
- 2.3 Spacecraft relative motion . . . 17
  - 2.3.1 Local Cartesian dynamics . . . 17
- 2.4 Linearized Cartesian relative motion and State Space Representation . . . 18
- 2.5 Parametric expressions for the spacecraft relative trajectory . . . 21
  - 2.5.1 Parametrizing relative trajectories . . . 21
- 2.6 Periodicity Properties . . . 23
- 2.7 Summary . . . 23

3 Model Predictive Control Law and its Computation **25**
- 3.1 Control System . . . 25
- 3.2 Actuator Saturation and Limit Budget . . . 26
- 3.3 Constrained spacecraft relative trajectories . . . 27
  - 3.3.1 Definition of admissible trajectories . . . 27
  - 3.3.2 Finite description of admissible trajectories . . . 29
  - 3.3.3 Finite description using non-negative polynomials . . . 29
  - 3.3.4 Rational expressions for the spacecraft relative motion . . . 30
  - 3.3.5 Constrained periodic trajectories . . . 32
- 3.4 Problem Formulation . . . 33
  - 3.4.1 Conclusion . . . 34

4 Alternating Projections Algorithm **35**
- 4.1 Algorithm Principle . . . 35
- 4.2 Example . . . 38
- 4.3 Implementation . . . 39
  - 4.3.1 Details of Implementation: Construction of the iteration variable . . . 40
  - 4.3.2 Details of Implementation: Translation of the problem . . . 42
  - 4.3.3 Break Conditions . . . 44
  - 4.3.4 Pseudocode . . . 45
- 4.4 Initial Guess . . . 45

4.5 Justification of the use of the Alternating Projections Algorithm 47
4.6 Conclusion 48

5 Initialisation of the Alternating Projections Algortihm 49
5.1 Simulator 49
5.1.1 Scenario of simulation 50
5.2 Classic Initialisations 51
5.2.1 Cold Start 51
5.2.2 Warm Start 51
5.3 Current Point Approach 52
5.3.1 Contruction of the initial guess 54
5.4 Optimization Process 57
5.5 Simulated Annealing 59
5.5.1 Pseudocode 61
5.6 Results Analysis 62

6 Feasibility of the Problem 65
6.1 Feasibility of a state 65
6.2 Control Action in D Space 70
6.3 Conclusion 74

7 Numerical Analysis 75
7.1 Example 75
7.1.1 Trajectory 75
7.1.2 Iterations 76
7.1.3 Control Output 79
7.1.4 Constraints 80
7.2 Break Conditions 81
7.2.1 Convergence 81
7.2.2 Prediction 86
7.2.3 Maximum Iterations 88
7.3 Conclusions 88

Conclusion 92

A Properties of non negative polynomials 93
A.1 Checking polynomials non negativity on a finite interval 93
A.2 Checking polynomials non negativity on an infinite interval 95

B Dynamics of the vector of parameters 97

C Nature of the roots of a quartic function 101

List of Figures

1.1 Stages of the rendezvous representation. 11
1.2 Three axis actuators representation. 12

2.1 The Earth Centered Inertial frame and the satellite trajectory. 16
2.2 The spacecraft relative position and the leader's LVLH frame 18

3.1 Block Diagram of the Control System. 26
3.2 Examples of periodic spacecraft relative trajectories that evolve inside a polytopic set. 29

4.1 Algorithm behaviour when there is an intersection 36
4.2 Algorithm behaviour when there is no intersection 37
4.3 Ilustration of a fast convergence 38
4.4 Ilustration of a slow convergence 39
4.5 Example of the topology the algorithm works in 46
4.6 Another example of topology the algorithm work in 47

5.1 Simulink simulation scheme used in this work 50
5.2 Trajectory for Cold start 52
5.3 Trajectory for Warm start 53
5.4 Trajectory Comparison Warm Start and Cold Start 53
5.5 Simulated trajectory for the initial point approach. 56
5.6 Simulated Trajectory using Optimal Controls 59
5.7 Consumption Comparison between Warm, Optimal and Simulated Annealing 62
5.8 Relative Comsumptions Comparison 63

6.1 Region of feasible solutions 67
6.2 Top view of the region of feasible solutions 67
6.3 Front/Lateral view of the region of feasible solutions 68
6.4 Cuts of the region for some d_3 values. 68
6.5 Superquadric that model the feasible region 70
6.6 Feasible region for $e = 0.3$ 71
6.7 Lateral view of the feasible region for $e = 0.3$ 71
6.8 Cuts for the feasible region for $e = 0.3$ 71
6.9 Attainable region from an arbitrary point using one control. 72
6.10 Region result of the Minkowski sum for anomaly 0 73
6.11 Region result of the Minkowski sum for two different anomalies (0 and $\pi/3$) 74

7.1 Iterations per Step Warm Start 77
7.2 Iterations per Step Warm Start 77
7.3 Iterations per Step Cold Start 78
7.4 Iterations per Step Cold Start 78
7.5 Iterations per Step when Varying Tolcons 82
7.6 Iterations per Step when Varying Tolvp 84
7.7 Iterations per Step when Varying both Tolerances 85
7.8 Iterations per Step for Infinity Norm 87
7.9 Iterations per step for Cold Start with Inf. Norm 88

List of Tables

5.1 Data for the scenario simulated . . . 51
5.2 Consumption for each initial guess strategy for each control frequence . . . 57
5.3 Optimal Controls given by the solver SeDuMi using Yalmip . . . 58
5.4 Controls given by the warm approach . . . 58
5.5 Consumption Comparison between Cold, Warm and Optimal controls. . . . 60

7.1 Data for the scenario simulated . . . 76
7.2 Control Values for Warm Start . . . 79
7.3 Control Values for Cold Start . . . 79
7.4 Constraints Evolution for the Transitory Regime . . . 80
7.5 Constraints Evolution for the Steady Regime . . . 80
7.6 Constraints Evolution Cold Start . . . 81
7.7 Constraints Evolution Cold Start . . . 81
7.8 Constraints Evolution Cold Start . . . 81
7.9 Difference in the Control Output when Varying Tolcons . . . 83
7.10 Difference in the Control Output when Varying Tolvp . . . 84
7.11 Difference in the Control Output when Varying both Tolerances . . . 84
7.12 Constraints Evolution in a Non Convergence Case . . . 86
7.13 Constraints Evolution in a Non Convergence Case . . . 89
7.14 Prediction of Iterations Evolution . . . 90

Chapter 1

Introduction

1.1 Spacecraft Rendezvous

The orbital rendezvous can be characterized as the set of operations performed by a satellite (the follower) in order to reach a specific position with respect to another satellite (the target). Normally, this set of operations is divided into several stages, between two consecutive stages the follower must remain in a certain neighbourhood of the last reference position. Figure 1.1 ilustrate the problem.

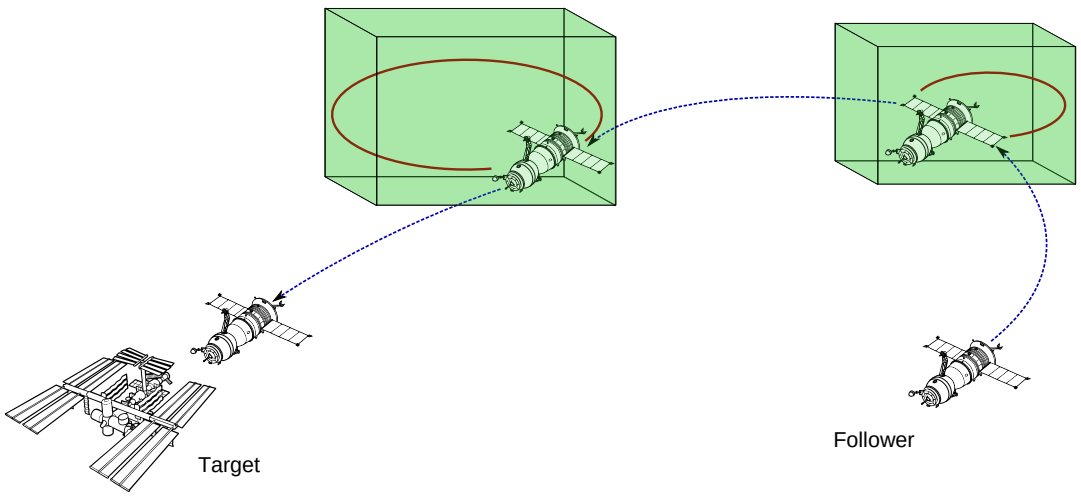

Figure 1.1: Stages of the rendezvous representation.

The presence of perturbations does not allow the satellite to remain at a point or on a stable trajectory and, therefore, corrections must be applied to the follower's movement to ensure a proper development of the relative trajectories between the satellites and thus the completion of the rendezvous mission.

The interest is to build a predictive control law that will keep the follower inside a given tolerance polytope. As the target is considered passive, the control is only applied on the target in the form of speed impulses. These impulses have the objective to bring the follower into a periodic trajectory that respect the dimensional constraint. It was chosen a periodic trajectory, because in the absence of pertubations (short period of time) the spacecraft will maintain itself inside the polytope with no need to apply corrections.

It is considered that all the corrections applied to the spacecraft trajectory are realized by the follower's thrusters (three couples of indentical propulsors, each being symmetrically mounted on each of the three axes, see Figure 1.2). This fact makes the follower spacecraft controllable in all the directions (three degrees of freedom). It is also considered that the velocity change is instantaneous, which is justified because the time constant of the orbit trajectory is much greater than the time constant to change the satellite's speed.

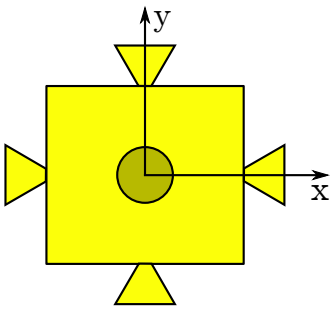

Figure 1.2: Three axis actuators representation.

The total consumption of these thrusters is modeled as the sum of all velocity variations on each axis and at each impulse:

$$\sum_{i=1}^{N} \|\Delta V_i\|_1 = \sum_{i=1}^{N} (|\Delta V_{i_x}| + |\Delta V_{i_y}| + |\Delta V_{i_z}|) \tag{1.1}$$

here ΔV_i represents the speed change that is applied to the aircraft. It is used a simplification when it comes to the consumption, it is considered that the consumption is equal to the sum of the velocity variations applied during the mission.

In order to calculate the control, a numerical algorithm is used. This process is based on finding a solution that respects all the constraints of the problem (geometrical, saturation, fuel budget) and guarantees that, apart from perturbations, the spacecraft will be in a good orbit for the problem.

1.2 Motivation and Objectives

This work is inspired by the work of Deaconu, Louembet and Arantes [8], [6], [9],[10] who proposed solutions to the problem of spacecraft rendezvous. The problem formulation proposed by Deaconu in [6] was solved by Arantes and Louembet in [9], [10] using the idea of a simple algorithm that could be used even with low computional resources.

To clarify where this work begins and what it wants to do, it can be explicited all the material received from past researches. This works receive already made:

- The model of the system presented in [6];
- The problem formulation done in [8] and [9];
- The solution using the Alternating Projections Algorithm proposed in [10];
- The nonlinear simulator written in *Simulink* from [11].

What this work does is to analyse this solution doing both a numerical analysis of the algorithm used (number of iterations, tolerances used, different behaviours face to different problems) and an optimality analysis of the solution produced by the algorithm in terms of fuel consumption.

This work aims to analyse ways to improve the solution proposed in the above cited work. This objective is divided between:

- An analysis of the input parameters of the algorithm that calculates the control;
- A study of feasibility of the problem proposed;
- A numerical analysis of the iterative part of the process.

These objectives are justified by the following reasons:

1. There is no proof of how good can the algorithm be in terms of optimization;
2. The good selection of input parameters can improve the control performance;
3. The knowledge of problem feasibility gives a way to know when it is possible to calculate the control;

4. The control law will be physically implemented in a parallel project and the numerical analysis is necessary for its success and validation.

1.3 Organisation

Chapter 2 presents the dynamical model and a closed form solution for it. Chapter 3 describes the predictive control law and presents the problem. Chapter 4 focuses on the algorithm used to calculate the control which is a solution to the problem presented in Chapter 3. Chapter 5 brings an analysis of the initialisation of the Alternating Projections Algorithm and compares its solution to the solution obtained with an optimization solver. Chapter 6 gives a deeper insight into the conditions of feasibility of the problem and how this can be used to improve the control law. Chapter 7 discusses the numerical analysis of the proposed algorithm.

Chapter 2

Spacecraft relative motion

2.1 Introduction

In the rendezvous problem treated in this work, the dynamic is composed of the relative motion between two spacecraft, one called *the follower*, with respect to the dynamics of another spacecraft, called *the leader* or *the target*. In order to achieve the dynamic model, first it is described the dynamic of the leader spacecraft orbiting around the earth. Than it is taken the target dynamic and the relative motion between them.

The solution of the dynamic is presented in the form of a state transition matrix. This way the dynamic can be predicted for a time horizon if needed. Also, it is proposed a change of variable based on the state transition matrix. With the new set of variables some characteristics of the trajectories can be put in evidence and a formal description of trajectories based on a set of parameters can be made.

This chapter is based on the work of Georgia Deaconu [6].

2.2 Dynamics of a spacecraft orbiting the Earth

It was chosen a Keplerian model in which the Earth is represented as a homogeneous sphere and the spacecraft motion is affected only by gravitational accelerations. The Keplerian framework leads to less accurate but simplified dynamical models for the spacecraft relative motion. These simplified models are well suited for control synthesis purposes, which is the case in this work.

The Keplerian dynamics of a spacecraft with respect to the Earth can be derived from

Newton's equations of motion between two mass particles. In this case, the motion of a spacecraft orbiting the Earth is described by the following differential equation [3]:

$$\left(\frac{d^2\vec{R}}{dt^2}\right)_{B_0} = -\frac{\mu}{\|\vec{R}\|^3}\vec{R} \tag{2.1}$$

where $\vec{R}$ represents the vector from the center of mass of the Earth to the spacecraft center of mass and μ is the Earth's gravitational constant. The dynamics are expressed with respect to an Earth centered inertial frame $R_0 = (0, \vec{X}, \vec{Y}, \vec{Z})$ illustrated in Figure 2.1. The fundamental plane for R_0 is the Earth's equatorial plane, the $\vec{Z}$ axis coincides with the rotation axis of the Earth and is oriented towards the North Pole, the $\vec{X}$ axis points to the vernal equinox and the $\vec{Y}$ axis is orthogonal to the $\vec{X}\vec{Z}$ plane.

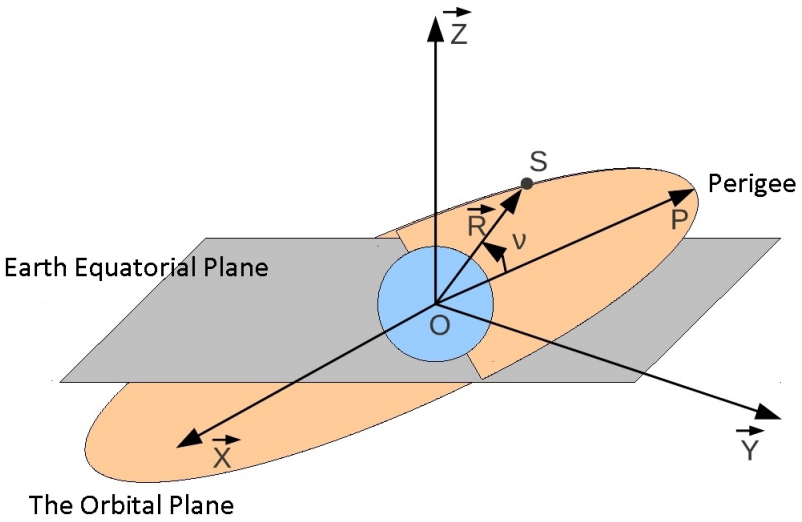

Figure 2.1: The Earth Centered Inertial frame and the satellite trajectory.

Let the orbital plane be the plane which contains the trajectory of the orbiting spacecraft (see Figure 2.1). The equation of the spacecraft trajectory expressed using polar coordinates with respect to this plane is given by [3]:

$$R = \|\vec{R}\| = \frac{a(1-e^2)}{1+e\cos\nu} \tag{2.2}$$

where a is called the semi-major axis of the spacecraft orbit, e is called the eccentricity and ν is called the true anomaly. The satellite's orbit is bounded if $e < 1$ and unbounded

if $e \geq 1$. For $e = 0$ the spacecraft trajectory is a circle of radius a and for $0 < e < 1$ the trajectory is an ellipse. The true anomaly ν represents the angle between the spacecraft's current position and the direction of the perigee (Figure 2.1).

The parameters a and e define the dimension and the shape of the satellite's orbit, while ν gives the instantaneous location of the satellite on its orbit. Under Keplerian assumptions, the true anomaly changes with time [3]:

$$\dot{\nu} = \sqrt{\frac{\mu}{a^3(1-e^2)^3}}\,(1 + e\cos\nu)^2. \tag{2.3}$$

To complete the description of the spacecraft relative motion, the state of the follower satellite must be expressed with respect to the state of the leader and the representation used in this work is introduced next.

2.3 Spacecraft relative motion

The spacecraft relative motion refers to the study of the dynamics of the follower spacecraft in relation to a moving frame in the leader spacecraft center of mass. Here it is made the choice to use the cartesian position and speed as the state of the spacecraft model. The mathematical development is presented in what follows.

2.3.1 Local Cartesian dynamics

The spacecraft relative motion represented using local Cartesian dynamics is defined with respect to a local rotating Cartesian frame centered on the leader satellite. A commonly used frame is the Local Vertical Local Horizontal (LVLH) frame $R_l = (S_l, \vec{x}, \vec{y}, \vec{z})$ illustrated in Figure 2.2. The $\vec{z}$ axis is radially oriented from the leader satellite towards the center of the Earth, the $\vec{y}$ axis is orthogonal to the orbital plane, in the opposite direction with respect to the angular momentum vector, and the $\vec{x}$ axis lays in the leader's orbital plane in the direction of the satellite's velocity.

The relative position between the leader spacecraft S_l and the follower spacecraft S_f is represented by $\vec{r} = \overrightarrow{S_l S_f}$ in Figure 2.2. Considering that the Keplerian dynamics of each satellite with respect to the Earth can be described using (2.1), the relative inertial

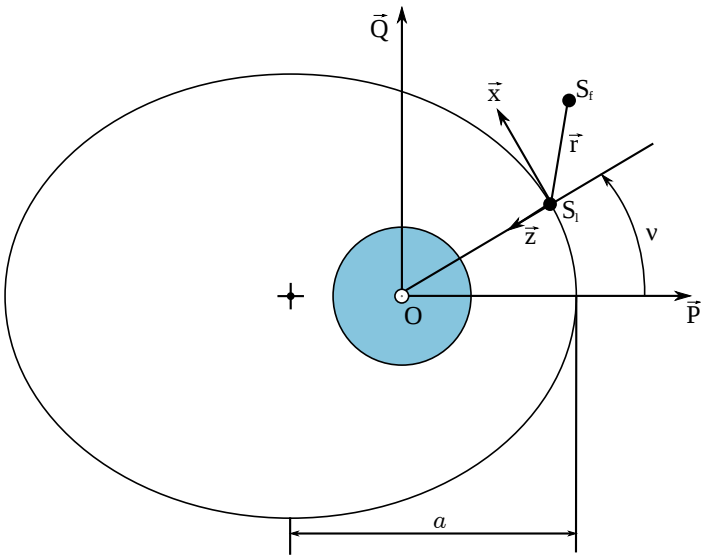

Figure 2.2: The spacecraft relative position and the leader's LVLH frame

acceleration can be written as:

$$\left(\frac{d^2\vec{r}}{dt^2}\right)_{B_0} = -\frac{\mu}{\|\vec{R}+\vec{r}\|^3}(\vec{R}+\vec{r}) + \frac{\mu}{\|\vec{R}\|^3}\vec{R} \tag{2.4}$$

where $\vec{R} = \overrightarrow{OS_f}$ represents the inertial position of the leader spacecraft.

Assuming that the dynamics of the leader spacecraft are expressed using orbital elements and that the spacecraft relative state is given by the local relative position and velocity $X = \begin{bmatrix} x & y & z & v_x & v_y & v_z \end{bmatrix}^T$, we have:

$$\begin{aligned} \ddot{x} - 2\,\dot{\nu}\,\dot{z} - \ddot{\nu}\,z - \dot{\nu}^2 x &= -\frac{\mu\,x}{\sqrt{(x^2+y^2+(R-z)^2)^3}} \\ \ddot{y} &= -\frac{\mu\,y}{\sqrt{(x^2+y^2+(R-z)^2)^3}} \\ \ddot{z} + 2\,\dot{\nu}\,\dot{x} + \ddot{\nu}\,x - \dot{\nu}^2 z &= -\frac{\mu(R-z)}{\sqrt{(x^2+y^2+(R-z)^2)^3}} + \frac{\mu}{R^2} \end{aligned} \tag{2.5}$$

2.4 Linearized Cartesian relative motion and State Space Representation

Let the spacecraft relative state vector be defined by the relative position and velocity projected on each axis of the leader's LVLH frame: $X = \begin{bmatrix} x & y & z & v_x & v_y & v_z \end{bmatrix}^T$. In the case where the distance between the two satellites is a lot smaller than the distance from

the leader satellite to the center of the Earth ($\|\vec{r}\| \ll \|\vec{R}\|$), the linearized Tschauner-Hempel equations can be used to describe the spacecraft relative motion [17]:

$$\begin{aligned} \ddot{x} &= 2\,\dot{\nu}\,\dot{z} + \ddot{\nu}\,z + \dot{\nu}^2 x - \frac{\mu}{R^3}x \\ \ddot{y} &= -\frac{\mu}{R^3}y \\ \ddot{z} &= -2\,\dot{\nu}\,\dot{x} - \ddot{\nu}\,x + \dot{\nu}^2 z + 2\frac{\mu}{R^3}z \end{aligned} \tag{2.6}$$

It can be noticed that for the linearized equations, the dynamics on the y axis are decoupled from the dynamics in the xz plane and define a harmonical oscillator.

If in (2.6) the independent variable time is replaced by the true anomaly of the leader spacecraft, a simplified form can be obtained for the equations describing the relative dynamics between the leader and the follower spacecraft. The derivatives with respect to time are replaced by:

$$\frac{d(\cdot)}{dt} = \frac{d(\cdot)}{d\nu}\frac{d\nu}{dt} = (\cdot)'\dot{\nu} \quad \frac{d^2()}{dt^2} = \frac{d^2()}{d\nu^2}\dot{\nu}^2 + \frac{d()}{d\nu}\ddot{\nu} \tag{2.7}$$

and the following variable change is used:

$$\tilde{X}(\nu) = \begin{bmatrix} (1+e\cos\nu)I_3 & 0_3 \\ -e\sin\nu I_3 & \dfrac{(1+e\cos\nu)}{\dot{\nu}}I_3 \end{bmatrix} X(t) \tag{2.8}$$

where $I_3 \in \mathbb{R}^{3\times3}$ is the identity matrix and $0_3 \in \mathbb{R}^{3\times3}$ is the zero matrix.

This operation leads to a periodic state-space model for the spacecraft relative dynamics:

$$\tilde{X}'(\nu) = \tilde{A}(\nu)\tilde{X}(\nu) + \tilde{B}\,\tilde{u} \tag{2.9}$$

where the dynamical matrix $\tilde{A}(\nu)$ is given by:

$$\tilde{A}(\nu) = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \dfrac{3}{1+e\cos\nu} & -2 & 0 & 0 \end{bmatrix} \tag{2.10}$$

the control matrix $\tilde{B}$ is defined by $\tilde{B} = [0_3 \ I_3]^T$ and $\tilde{u} = [\tilde{u}_x \ \tilde{u}_y \ \tilde{u}_z]^T$ represents the acceleration generated by the spacecraft thrusters.

The system (2.6) can be solved analytically. In this work is used the solution proposed by [18] that consists in using the fact that the motion in the y axis is decoupled from x and z. So the state transition matrix has the form of:

$$\tilde{X}(\nu) = \Phi(\nu, \nu_0)\tilde{X}(\nu_0) \tag{2.11}$$

And can be divided in:

$$\begin{aligned} \tilde{X}_y(\nu) &= \Phi_y(\nu, \nu_0)\tilde{X}_y(\nu_0) \\ \tilde{X}_{xz}(\nu) &= \Phi_{xz}(\nu, \nu_0)\tilde{X}_{xz}(\nu_0) \end{aligned} \tag{2.12}$$

with solutions:

$$\tilde{X}_y(\nu) = \begin{bmatrix} \tilde{y}(\nu) \\ \tilde{v}_y(\nu) \end{bmatrix}, \qquad \Phi_y(\nu, \nu_0) = \begin{bmatrix} \cos(\nu - \nu_0) & \sin(\nu - \nu_0) \\ -\sin(\nu - \nu_0) & \cos(\nu - \nu_0) \end{bmatrix} \tag{2.13}$$

$$\tilde{X}_x z(\nu) = \begin{bmatrix} \tilde{x}(\nu) \\ \tilde{z}(\nu) \\ \tilde{v}_x(\nu) \\ \tilde{v}_z(\nu) \end{bmatrix}, \qquad \Phi_{xz}(\nu, \nu_0) = \phi_{xz}(\nu)\phi_{xz}^{-1}(\nu_0) \tag{2.14}$$

where

$$\phi_{xz}(\nu) = \begin{bmatrix} 1 & -\cos\nu(2+e\cos\nu) & \sin\nu(2+e\cos\nu) & 3(1+e\cos\nu)^2 J \\ 0 & \sin\nu(1+e\cos\nu) & \cos\nu(1+e\cos\nu) & 2-3e\sin\nu(1+e\cos\nu)J \\ 0 & 2\sin\nu(1+e\cos\nu) & 2\cos\nu(1+e\cos\nu)-e & 3-6e\sin\nu(1+e\cos\nu)J \\ 0 & \cos\nu+e\cos 2\nu & -\sin\nu-e\sin 2\nu & -3e\left((\cos\nu+e\cos 2\nu)J+\dfrac{\sin\nu}{1+e\cos\nu}\right) \end{bmatrix} \tag{2.15}$$

and

$$\phi_{xz}^{-1}(\nu_0) = \frac{1}{e^2-1}\begin{bmatrix} e^2-1 & -\dfrac{3e\sin\nu_0(2+e\cos\nu_0)}{1+e\cos\nu_0} & e\sin\nu_0(2+e\cos\nu_0) & 2-e\cos\nu_0(1+e\cos\nu_0) \\ 0 & \dfrac{3\sin\nu_0(e\cos\nu_0+1+e^2)}{1+e\cos\nu_0} & -\sin\nu_0(2+e\cos\nu_0) & -(\cos\nu_0+e\cos^2\nu_0-2e) \\ 0 & 3(e+\cos\nu_0) & -(2\cos\nu_0+e\cos^2\nu_0+e) & \sin\nu_0(1+e\cos\nu_0) \\ 0 & -(3e\cos\nu_0+e^2+2) & (1+e\cos\nu_0)^2 & -e\sin\nu_0(1+e\cos\nu_0) \end{bmatrix} \tag{2.16}$$

2.5 Parametric expressions for the spacecraft relative trajectory

A set of parametric expressions can be used to describe the spacecraft relative motion. These parameters are used as a way to classify different trajectories that we can obtain. By working in the space of parameters, some interesting properties of the spacecraft relative motion can be evidenced. Additionally, it is desirable to work only with periodic trajectories, so a characterisation is made to separate them using the parameters.

2.5.1 Parametrizing relative trajectories

Consider the relative motion between two spacecraft on arbitrary elliptical Keplerian orbits. The relative state $\tilde{X}(\nu)$ is defined by the spacecraft relative position and velocity expressed in the LVLH frame attached to the leader.

$$\tilde{X}(\nu) = \begin{bmatrix} \tilde{x}(\nu) & \tilde{y}(\nu) & \tilde{z}(\nu) & \tilde{v}_x(\nu) & \tilde{v}_y(\nu) & \tilde{v}_z(\nu) \end{bmatrix}^T \tag{2.17}$$

Parametric expressions for the relative position between the spacecraft can be obtained by expanding the terms in (2.11) and then factoring out some of the terms related to the independent variable ν:

$$
\begin{aligned}
\tilde{x}(\nu) &= (2 + e\cos\nu)(d_1\ \sin\nu - d_2\ \cos\nu) + d_3 + 3\,d_0\,J(\nu)(1 + e\cos\nu)^2 \\
\tilde{y}(\nu) &= d_4\ \cos\nu + d_5\ \sin\nu \qquad\qquad , \nu \geq \nu_0 \\
\tilde{z}(\nu) &= (1 + e\cos\nu)(d_2\ \sin\nu + d_1\ \cos\nu) - 3\,e\,d_0\,J(\nu)\sin\nu(1 + e\cos\nu) + 2\,d_0
\end{aligned}
\tag{2.18}
$$

Let $D(\nu_0) \in \mathbb{R}^6$ be the *vector of parameters* for the spacecraft relative motion evaluated at ν_0, defined as:

$$
D(\nu_0) = \begin{bmatrix} d_0(\nu_0) & d_1(\nu_0) & d_2(\nu_0) & d_3(\nu_0) & d_4(\nu_0) & d_5(\nu_0) \end{bmatrix}^T \tag{2.19}
$$

The elements of the vector $D(\nu_0)$ depend *linearly* on the initial spacecraft relative state from which the relative trajectory is propagated:

$$
D(\nu_0) = C(\nu_0)\tilde{X}(\nu_0) \tag{2.20}
$$

The matrix $C(\nu) \in \mathbb{R}^{6\times 6}$ is defined as a function of the eccentricity of the orbit of the leader satellite and the true anomaly for which the vector of parameters needs to be evaluated:

$$
C(\nu) = \begin{bmatrix}
0 & 0 & \frac{-(3\,e\cos\nu + e^2 + 2)}{e^2 - 1} & \frac{(1 + e\cos\nu)^2}{e^2 - 1} & 0 & \frac{-e\sin\nu(1 + e\cos\nu)}{e^2 - 1} \\
0 & 0 & \frac{3(e + \cos\nu)}{e^2 - 1} & \frac{-(2\cos\nu + e\cos^2\nu + e)}{e^2 - 1} & 0 & \frac{\sin\nu(1 + e\cos\nu)}{e^2 - 1} \\
0 & 0 & \frac{3\sin\nu(1 + e\cos\nu + e^2)}{(e^2 - 1)(1 + e\cos\nu)} & \frac{-\sin\nu(2 + e\cos\nu)}{e^2 - 1} & 0 & \frac{-(\cos\nu + e\cos^2\nu - 2e)}{e^2 - 1} \\
1 & 0 & \frac{-3\,e\sin\nu(2 + e\cos\nu)}{(e^2 - 1)(1 + e\cos\nu)} & \frac{e\sin\nu(2 + e\cos\nu)}{e^2 - 1} & 0 & \frac{e^2\cos^2\nu + e\cos\nu - 2}{e^2 - 1} \\
0 & \cos\nu & 0 & 0 & -\sin\nu & 0 \\
0 & \sin\nu & 0 & 0 & \cos\nu & 0
\end{bmatrix}
\tag{2.21}
$$

The advantage of expressing the spacecraft relative position in the form (2.18) is that it enables the direct identification of some of the effects that the values of the parameters have on the spacecraft relative trajectory. Parameters d_1 and d_2 influence the amplitude of the motion in the xz plane while parameters d_4 and d_5 define the amplitude of the periodic motion on the y axis. The value of the parameter d_3 corresponds to an offset term on the position on the x axis and the parameter d_0 defines an offset on the z axis

and influences the contribution of the integral term $J(\nu)$.

2.6 Periodicity Properties

Expressions (2.18) show that the integral term $J(\nu)$ is the only non periodic term in the propagation of the spacecraft relative position. In the particular case where:

$$d_0(\nu_0) = 0 \tag{2.22}$$

the resulting relative trajectory is *periodic* because the drifting term $J(\nu)$ is cancelled. This leads to the following simplified parametric expressions for the propagation of spacecraft periodic relative trajectories:

$$\begin{aligned}
\tilde{x}(\nu) &= (2 + e\cos\nu)(d_1(\nu_0)\sin\nu - d_2(\nu_0)\cos\nu) + d_3(\nu_0) \\
\tilde{y}(\nu) &= d_4(\nu_0)\cos\nu + d_5(\nu_0)\sin\nu \\
\tilde{z}(\nu) &= (1 + e\cos\nu)(d_2(\nu_0)\sin\nu + d_1(\nu_0)\cos\nu)
\end{aligned} \tag{2.23}$$

Expressions (2.23) reveal the fact that the spacecraft relative periodic trajectories are always centered around zero on the y and z axes. An offset can be set on the x axis through the d_3 parameter.

The dynamics associated with the parameters D can be studied depper, this study is presented in the appendix B.

2.7 Summary

The development done in this chapter can be summed up in the following way. First of all it was used the Newton gravitational law to describe the dynamics of a satellite (the *follower*), under Keplerian assumptions, orbiting around the Earth. Then the framework of this dynamic was changed to the center of mass of another satellite orbitating around the earth (the *target*). This dynamic was then expressed in terms of 3 second order differential equations. These equations have a analytical solution, so a state transition matrix was calculated. The state transition matrix is used to predict any future state based on the initial conditions, in another words, given the initial conditions it can trace

the trajectory of the system.

Analysing the elements in the state transition matrix, it is possible to group all the terms that are independent of the anomaly into a set of 6 variables that were called the D variables. These 6 variables appear as coefficients in the trajectory prediction and give some specific information about the characteristics of the trajectory such as periodicity, boundaries and centralization.

The following chapters will heavily use the vector D as way to parametrize a trajectory, so it is an alternative state of the system as importante as the classic positions and velocities.

Chapter 3

Model Predictive Control Law and its Computation

In this Chapter a predictive control law that will be used in the system is defined. The specifications of the problem are translated into convex constraints in order to have a rigorous formal description of it. Once the problem is written, it is possible to start searching for solutions. With the change of variable D introduced in the last chapter, there is a notion of modeling a trajectory now. The control objective is to find a periodic orbit that respects all the constraints and, then calculate the speed impulse needed to reach this orbit.

The chapter is divided into the Control system and the three different constraints: Saturation, Budget and Polytopic along with the final problem formulation and a brief discussion of the solutions.

This chapter is based on the work of Georgia Deaconu [6] and Paulo Arantes [9].

3.1 Control System

The control system in this work uses a model predictive impulsive law. Each time that it is triggered, it acquires the data from the navigation system in order to calculate a speed impulse that will maintain the orbit of the satellite inside a pre-defined polytope. Figure 3.1 schematizes the control system in a block diagram.

The control law is said predictive because it takes into account the evolution of spacecraft in a time horizon using the state transiton matrix. It is also said impulsive because the control is applied in the form of velocity impulses, once the change of velocity has

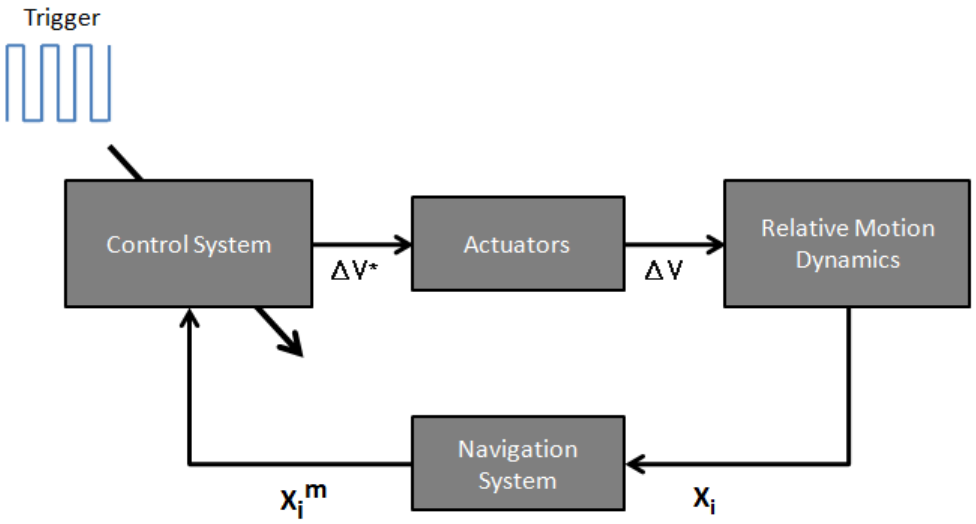

Figure 3.1: Block Diagram of the Control System.

a faster dynamics than the orbital mouvement, thus considering it as instanteneous or impulsive.

Maintaining a satellite in an orbit inside of a polytope is not a classical reference tracking problem as the trajectory is not chosen *a priori*. The algorithm tracks a ensemble of trajectories that respect the constraints rather than a single point. Besides that, the model is linear time varying and the constraints of saturations and budget are taken into account to the control computation. In this case, with all these characteristics, the classical techniques of control, and even the modern techniques, can not be used, so the problem is written as an optmization problem that is specified in the next sections.

3.2 Actuator Saturation and Limit Budget

The follower thrusters can not provide an arbitrary gain of speed. This saturation is a constraint in each of the directions and has a maximum value of ΔV_{max}. In addition, the lifespan of a satellite is defined by the amount of fuel left in its reservoir, so it is considered also a constraint on the total consumption σ. The set of constraints directly related to the actuators can be expressed as:

$$-\Delta V_{max} \leq \Delta V_i \leq \Delta V_{max}, \ \forall i = 1..N \tag{3.1}$$

$$\sum_{i=1}^{N} \|\Delta V_i\|_1 \leq \sigma \tag{3.2}$$

3.3 Constrained spacecraft relative trajectories

The present section shows a mathematical characterization of the set of vectors of parameters D that correspond to spacecraft relative orbits which respect some dimension constraints.

In what follows it will be shown a description of the admissible trajectories for the problem. Furthermore the characterization is made *a priori* continuously in time over a specified interval or in anomaly over a period. This leads to an inifinity number of constraints that need to be verified. A form of translating these constraints into the form of a finite convex description of the admissible spacecraft relative trajectories for a given set of dimensions constraints is also provided.

3.3.1 Definition of admissible trajectories

From a mathematical point of view, the constraints on the dimensions of the spacecraft relative trajectories can be written as linear constraints on the spacecraft relative positions:

$$\begin{array}{l} x_{\min} \leq x(t) \leq x_{\max} \\ y_{\min} \leq y(t) \leq y_{\max} \quad , \ \forall t \in [t_0 \ t_f] \\ z_{\min} \leq z(t) \leq z_{\max} \end{array} \iff \begin{array}{l} \tilde{x}_{\min}(\nu) \leq \tilde{x}(\nu) \leq \tilde{x}_{\max}(\nu) \\ \tilde{y}_{\min}(\nu) \leq \tilde{y}(\nu) \leq \tilde{y}_{\max}(\nu) \quad , \ \forall \nu \in [\nu_0 \ \nu_f] \\ \tilde{z}_{\min}(\nu) \leq \tilde{z}(\nu) \leq \tilde{z}_{\max}(\nu) \end{array} \tag{3.3}$$

Equation (3.3) illustrates the effects of the variable change (2.8) on the dimension constraints: the constant minimum and maximum bounds in the time domain are transformed into bounds that depend on the true anomaly of the leader spacecraft ν. The constraints must be respected continuously on the intervals $[t_0 \ t_f]$ and $[\nu_0 \ \nu_f]$ respectively.

The constraints in (3.3) can be written in a more compact way as:

$$H \, \tilde{X}(\nu) \leq \tilde{V}(\nu), \ \forall \nu \in [\nu_0 \ \nu_f] \tag{3.4}$$

where the matrices H and V define a generic polytopic set. Using the definition from (3.4), the set of spacecraft relative states from which the autonomously propagated trajectories remain inside the polytopic set (H, V) during the specified interval can be defined as:

$$S(H, V, \nu_0, \nu_f) = \left\{ \tilde{X}(\nu_0) \in \mathbb{R}^6 \mid \tilde{X}(\nu) = \Phi(\nu, \nu_0)\tilde{X}(\nu_0),\ H\,\tilde{X}(\nu) \leq \tilde{V}(\nu),\ \forall \nu \in [\nu_0\ \nu_f] \right\} \tag{3.5}$$

An equivalent form can be given to the set of parameters defining relative trajectories that respect the given trajectory constraints during the specified interval:

$$S_D(H, V, \nu_0, \nu_f) = \Big\{ D(\nu_0) \in \mathbb{R}^6 \mid D(\nu) = \Phi_D(\nu, \nu_0)D(\nu_0),\ H\,F(\nu)D(\nu) \leq \tilde{V}(\nu), \\ \forall \nu \in [\nu_0\ \nu_f] \Big\} \tag{3.6}$$

with the matrix $F(\nu) = C^{-1}(\nu)$ defined in (B.4).

The trajectory constraints need to be verified *continuously* on the specified interval. As a consequence, an *infinite* number of conditions need to be checked in order to certify that a state or a vector of parameters correspond to a trajectory which respects the specified requirements.

The set of periodic trajectories that satisfy the polytopic constraint is:

$$S_X^p(H, V) = \left\{ \tilde{X} \in \mathbb{R}^6 \mid \tilde{X}(\nu_0) = \tilde{X}(\nu_0 + 2\pi),\ H\tilde{X}(\nu) \leq \tilde{V}(\nu),\ \forall \nu \in [0\ 2\pi] \right\} \tag{3.7}$$

$$S_D^p(H, V) = \left\{ D \in \mathbb{R}^6 \mid d_0 = 0,\ H\,F(\nu)D \leq \tilde{V}(\nu),\ \forall \nu \in [0\ 2\pi] \right\} \tag{3.8}$$

The interval on which the constraints need to be checked is limited to one orbital period in this case because the trajectory is periodic, but checking that a vector D defines an admissible trajectory is still a hard problem.

Inside a given polytopic set there can be found many trajectories that respect the dimensions constraints (see Figure 3.2). We are interested in obtaining a finite description

of *all* these admissible periodic trajectories.

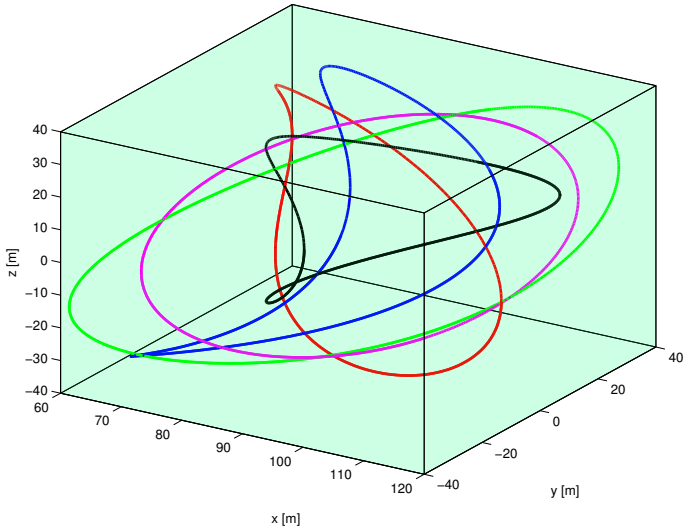

Figure 3.2: Examples of periodic spacecraft relative trajectories that evolve inside a polytopic set.

3.3.2 Finite description of admissible trajectories

Imposing continuous constraints on the spacecraft relative trajectories leads to a description of the admissible trajectories using an infinite number of constraints. The provided description is accurate but not very well suited for our purposes. This is due to the difficulty in certifying that a given trajectory respects all the required conditions. A method for reaching a finite description of the admissible trajectories is presented in what follows.

3.3.3 Finite description using non-negative polynomials

The idea of using the properties of non-negative polynomials to obtain a finite description of the admissible spacecraft relative trajectory came from the desire to exploit the structure of the solution for the relative motion provided by the transition matrix. The expressions (2.23) show that for the periodic relative motion the trajectory is defined by trigonometric polynomials. In this case, the dimension constraints (3.3) can be written as polynomial non-negativity constraints through a change of variable.

Once the dimensions constraints on the spacecraft relative trajectory are transformed into polynomial non negativity constraints of the type:

$$P(w) \geq 0, \ \forall w \in \mathcal{W} \tag{3.9}$$

the results presented by Nesterov in [14] can be used in order to obtain a finite description of the admissible trajectories without relying on discretization. It is shown that polynomial non negativity constraints can be transformed into conditions of existence of one or two constrained positive semi-definite matrices (see Appendix A). The infinite number of points where the polynomial non-negativity constraint needed to be checked can be replaced by one Linear Matrix Inequality (LMI) constraint.

3.3.4 Rational expressions for the spacecraft relative motion

The following variable change can be used in order to transform the trigonometrical terms in the expressions for the propagation of the spacecraft relative trajectory into rational terms:

$$w = \tan\left(\frac{\nu}{2}\right), \quad \cos\nu = \frac{1-w^2}{1+w^2}, \quad \sin\nu = \frac{2w}{1+w^2}, \tag{3.10}$$

Introducing (3.10) into (2.18) leads to the following expressions for the spacecraft relative positions:

$$\begin{aligned} \tilde{x}(w) &= \frac{1}{(1+w^2)^2}\left[P_x(w) + 3\,d_0\,P_{Jx}(w)\,J(w)\right] \\ \tilde{y}(w) &= \frac{1}{1+w^2}P_y(w) \qquad\qquad\qquad\qquad , w \geq w_0 \\ \tilde{z}(w) &= \frac{1}{(1+w^2)^2}\left[P_z(w) + 2\,d_0\,P_{Jz}(w)\,J(w)\right] \end{aligned} \tag{3.11}$$

where the polynomials $P_{Jx}(w)$ and $P_{Jz}(w)$ depend only on the eccentricity of the orbit of the leader satellite and are given by:

$$P_{Jx}(w) = ((1+e) + (1-e)w^2)^2 \quad P_{Jz}(w) = -3e((1-e)w + (1-e)w^3) \tag{3.12}$$

and the term $J(w)$ is obtained by introducing the variable change (3.10):

$$J(w) = \int_{w_0}^{w} \frac{2\,\tau^2 + 2}{((1-e)\tau^2 + e + 1)^2} d\tau \tag{3.13}$$

The polynomials $P_x(w)$, $P_y(w)$ and $P_z(w)$ are defined by:

$$P_x(w) = \sum_{i=0}^{4} p_{xi}\, w^i \quad P_y(w) = \sum_{i=0}^{2} p_{yi}\, w^i \quad P_z(w) = \sum_{i=0}^{4} p_{zi}\, w^i \tag{3.14}$$

and their vectors of coefficients, $p_x = \begin{bmatrix} p_{x0} & p_{x1} & p_{x2} & p_{x3} & p_{x4} \end{bmatrix}^T$, $p_y = \begin{bmatrix} p_{y0} & p_{y1} & p_{y2} \end{bmatrix}^T$ and $p_z = \begin{bmatrix} p_{z0} & p_{z1} & p_{z2} & p_{z3} & p_{z4} \end{bmatrix}^T$ respectively, depend linearly on the vector of parameters $D(\nu_0)$:

$$p_x = C_x\, D(\nu_0) \quad p_y = C_y\, D(\nu_0) \quad p_z = C_z\, D(\nu_0) \tag{3.15}$$

The matrices C_x, C_y and C_z depend only on the eccentricity of the reference orbit and are given by:

$$C_x = \begin{bmatrix} 0 & 0 & -2-e & 1 & 0 & 0 \\ 0 & 4+2e & 0 & 0 & 0 & 0 \\ 0 & 0 & 2e & 2 & 0 & 0 \\ 0 & 4-2e & 0 & 0 & 0 & 0 \\ 0 & 0 & 2-e & 1 & 0 & 0 \end{bmatrix} \quad C_y = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & -1 & 0 \end{bmatrix} \quad C_z = \begin{bmatrix} 0 & e+1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2e+2 & 0 & 0 & 0 \\ 0 & -2e & 0 & 0 & 0 & 0 \\ 0 & 0 & 2-2e & 0 & 0 & 0 \\ 0 & e-1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.16}$$

The final purpose is to reach some polynomial expressions for the constrained spacecraft relative motion that can lead to a finite description of the admissible trajectories. The expressions (3.11) are not entirely rational because of the presence of the term $J(w)$. Without further manipulations, the spacecraft relative trajectory is defined by rational expressions only in the case of periodic motion. However when $d_0 = 0$, the relative trajectory is given by:

$$\tilde{x}(w) = \frac{1}{(1+w^2)^2}\, P_x(w) \quad \tilde{y}(w) = \frac{1}{1+w^2}\, P_y(w) \quad \tilde{z}(w) = \frac{1}{(1+w^2)^2}\, P_z(w) \tag{3.17}$$

3.3.5 Constrained periodic trajectories

The description of the admissible trajectories is simplified in the case of periodic spacecraft relative motion. The periodic trajectories that respect some polytopic constraints can be defined directly in terms of non negativity conditions of some rationals. In the periodic case, the expression is:

$$H\,\tilde{X}(\nu) \leq \tilde{V}(\nu),\ \forall \nu \in [\nu_0\ \nu_f] \tag{3.18}$$

which implies that

$$h_{i,1}\,\tilde{x}(w) + h_{i,2}\,\tilde{y}(w) + h_{i,3}\,\tilde{z}(w) \leq \frac{1+e+(1-e)w^2}{1+w^2}\,v_i,\ \forall w \in [w_0\ w_f],\ i = 1..s \tag{3.19}$$

Then we can define Ξ

$$\Xi_i(w) = -h_{i,1}\,\tilde{x}(w) - h_{i,2}\,\tilde{y}(w) - h_{i,3}\,\tilde{z}(w) + \frac{1+e+(1-e)w^2}{1+w^2}\,v_i \geq 0,\ i = 1..s \tag{3.20}$$

By bringing the terms to the lowest common denominator, a more compact form can be obtained for $\Xi_i(w)$:

$$\Xi_i(w) = \frac{1}{(1+w^2)^2}\,\Gamma_i(w),\ i = 1..s \tag{3.21}$$

where the polynomials $\Gamma_i(w)$ are defined by:

$$\Gamma_i(w) = -h_{i,1}\,[P_x(w)+3\,d_0\,P_{Jx}(w)\,J(w)] - h_{i,2}\,\bar{P}_y(w) - h_{i,3}\,[P_z(w)+2\,d_0\,P_{Jz}(w)\,J(w)] + v_i\,T(w) \tag{3.22}$$

which in the periodic case is:

$$\Gamma_i^p(w) = -h_{i,1}\,P_x(w) - h_{i,2}\,\bar{P}_y(w) - h_{i,2}\,P_z(w) + v_i\,T(w) \tag{3.23}$$

In the previous definition the polynomial $\bar{P}_y(w)$ is obtained as $\bar{P}_y(w) = (1+w^2)P_y(w)$

and the polynomial $T(w)$ is given by $T(w) = \sum_{i=0}^{4} t_i\, w^i$ with the vector of coefficients $t = \begin{bmatrix} 1+e & 0 & 2 & 0 & 1-e \end{bmatrix}^T$. The coefficients of the other polynomials in (3.22) depend on the vector of parameters D, as defined in (3.14). This leads to the definition of the set of constant parameters that correspond to admissible trajectories using a finite number of polynomial non negativity constraints:

$$S_D^p(H,V) = \left\{ D \in \mathbb{R}^6 \mid d_0 = 0,\ \Gamma_i^p(w,D) \geq 0,\ \forall w \in \mathbb{R},\ i = 1..s \right\} \tag{3.24}$$

The non negativity of the polynomials $\Gamma_i^p(w)$ needs to be checked on an infinite interval since the variable change (3.10) maps one orbital period to $\mathbb{R}$.

Using the property of non negative polynomials on infinite intervals given in [14] and in A, the set of parameters corresponding to spacecraft periodic relative trajectories that evolve inside a specified polytopic set can be defined as:

$$S_D^p(H,V) = \left\{ D \in \mathbb{R}^6 \;\middle|\; \begin{array}{l} d_0 = 0 \\ \exists\, Y_i \succeq 0 \text{ s.t } \gamma_i^p = \Lambda^*(Y_i),\ \forall i = 1..s \end{array} \right\} \tag{3.25}$$

where γ_i^p are the vectors of coefficients corresponding to the polynomials $\Gamma_i^p(w)$. Since we are dealing with periodic trajectories, the vector of parameters is constant over the infinite interval. The degree of the polynomials $\Gamma_i^p(w)$ is easy to read in this case and is less or equal to 4 (see (3.23) and (3.14)). This means that the variables Y_i are at most 3 by 3 matrices.

3.4 Problem Formulation

Taking the constraints (3.1), (3.2) and (3.25) along with the transition matrices presented in the previous chapters, the rendezvous problem can be written as follows:

Given an impulse, ν the true anomaly, $X(\nu)$ the spacecraft state at ν and H and V the matrices that define the polytope, determine the existence of a control vector ΔV such that:

$$
\begin{cases}
-\Delta V_{max} \leq \Delta V \leq \Delta V_{max} \\
\|\Delta V\|_1 \leq \sigma \\
\Delta \tilde{V} = \left(\frac{1+ecos(\nu_i)}{\dot{\nu}_i}\right) \Delta V \\
\tilde{X}(\nu) = T(\nu) X(t) \\
\tilde{X}(\nu) = \phi(\nu, \nu_0)\tilde{X}(\nu_0) + \phi(\nu, \nu_0)\tilde{B}\Delta\tilde{V} \\
D = C(\nu)\tilde{X}(\nu) \\
D \in S_D(H, V) \\
Y_i \succeq 0, \ \forall i = 1..s \\
\gamma_i^p = \Lambda^*(Y_i), \ \forall i = 1..s
\end{cases}
\tag{3.26}
$$

The constraints can all be written as non negativity of symmetric matrices for which the coefficients respect some equalities. Given that these constraints are essentially convex, to find a solution for this problem is the same as finding an intersection between two convex subspaces (the equalities define a subspace of the symmetric real matrices and the cone of semi-definite positive matrices).

3.4.1 Conclusion

In this chapter it was defined a predictive impulsive control law by formally defining all the constraints to which the solution must be submitted (3.26). The objective was to provide a vector of velocity (three axes) that makes that, apart from perturbations, the orbit made by the spacecraft lay inside the defined polytope. The final description shows that all the vectors ΔV, that are solutions to the problem, lay inside the intersection of two well defined convex sets. The problem of finding an intersection between two convex sets can be solved in many ways. As this is a problem for embedded systems, there are computational constraints and the solution can not be of a great level of complexity. One of the simplest ways of find a solution to it is the Alternating Projection Algorithm proposed by Dykstra in [7]. This algorithm was used in [10] and in [9] and it is based on it that the analysis in what follows in this work will be done.

Chapter 4

Alternating Projections Algorithm

Previous chapter described the formal problem of rendezvous. As stated, to calculate a control it is necessary to search for an element which is in two closed convex sets, what can be done by finding a point in the intersection of both sets. As a low cost solution from the computational point of view is searched, the idea is to use the Alternating Projections Algorithm proposed by [4] and used in [10] and [9] to solve the problem.

The present and the following chapters will present and use this algorithm exhaustively, so a great importance is given to the knowledge of how it works. Some examples will be shown first and the real implementation in *Matlab* for the resolution of the problem will come next.

4.1 Algorithm Principle

Although many authors worked in this algorithm, here the work of Boyd and Dattoro [4] is used as reference. Alternating projections is a very simple algorithm for computing a point in the intersection of two convex sets using a sequence of projections onto these sets. The main idea behind the algorithm is to trace a path between the sets untill a point of the intersection is reached.

Like many other iterative algorithms, alternating projections can be slow, but if we have a good way of calculating the projections it is very efficient. As used in [4], the Euclidean norm, Euclidean distance, and Euclidean projection will be used as standard.

Suppose C and D are two closed convex sets in R^n , and let P_C and P_D denote projection on C and D, respectively. The algorithm starts with any $x_0 \in C$, and then

alternately projects onto C and D:

$$Y_k = P_D(X_k), X_{k+1} = P_C(Y_k), k = 0, 1, 2, ... \tag{4.1}$$

This generates a sequence of points $X_k \in C$ and $Y_k \in D$. A basic result, shown by Cheney and Goldstein [5] is that: If $C \cap D \neq \varnothing$, then the sequences X_k and Y_k both converge to a point $x^* \in C \cap D$. Roughly speaking, alternating projections finds a point in the intersection of the sets, provided they intersect. Note that the algorithm does not produces a point in $C \cap D$ in a finite number of steps. What is claimed is that the sequence X_k (which lies in C) satisfies $dist(X_k, D) \to 0$, and likewise for Y_k . A simple example is illustrated in figure 4.1.

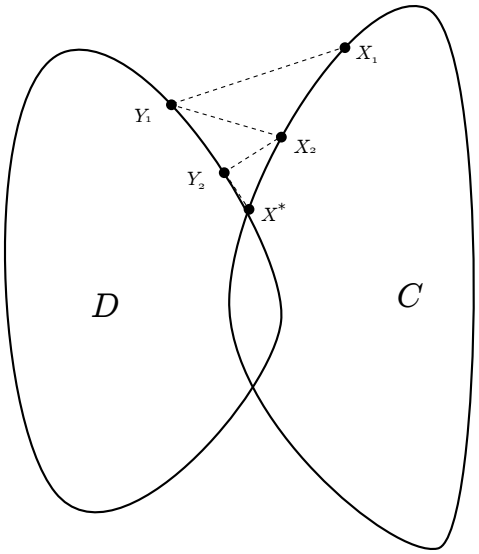

Figure 4.1: Algorithm behaviour when there is an intersection

The alternating projections algorithm can also be used when the sets do not have an intersection. In this case it can be proved that:

Assume the distance between C and D is achieved (i.e., there exist points in C and D whose distance is $dist(C, D)$). Then $X_k \to x^* \in C$, and $Y_k \to y^* \in D$, where $\|x^* - y^*\|_2 = dist(C, D)$. In other words, alternating projections converges to a pair of points in C and D that have the minimum distance between the sets. In this case, alternating projections also yields (in the limit) a hyperplane that separates C and D. A simple example is illustrated in figure 4.2.

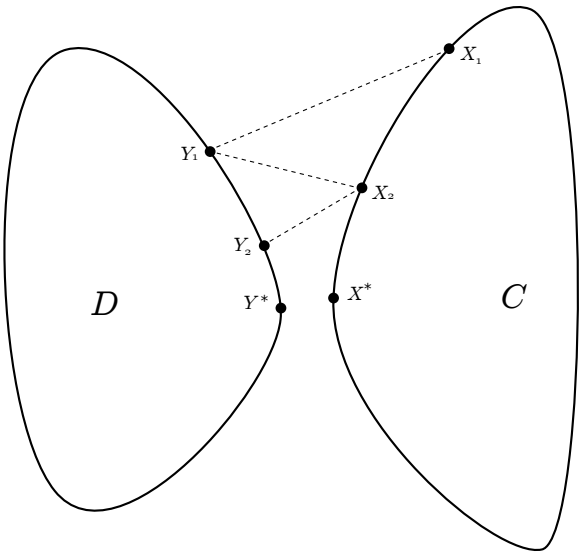

Figure 4.2: Algorithm behaviour when there is no intersection

The proof of convergence is presented in [4].

The Projections

In the work of Boyd and Dattoro [4] they give an example for the feasibility of Positive Semidefinite Definition problems that we are dealing with. The analytical formulas used in their work are what will be used in this work in order to do the projections.

Find $\chi \in S^n$ that satisfies

$$\begin{cases} \chi \succeq 0 \\ tr(A_i\chi) = b_i, \ A_i \in S^n, \ i = 1..m \end{cases} \tag{4.2}$$

Here we take C to be the semidefinite positive cone of S^n, and we take D to be the affine set in S^n defined by the linear equalities. The Euclidean norm here is the Frobenius norm.

The projection of iterate Y_k onto C can be found from the eigenvalue decomposition ($Y_k = VDV^T$)

$$P_C(Y_k) = Vdiag(max\{0, \lambda_1\}, ..., max\{0, \lambda_n\})V^T \tag{4.3}$$

in other words, all the positive eigenvalues are conserved and the others are set to 0.

The projection of iterate X_k onto the affine set is also easy to work out:

$$P_D(X_k) = X_k - \sum_{i=1}^{m} u_i A_i \tag{4.4}$$

where u_i are found from the normal equations:

$$Gu = (tr(A_1 X_k) - b_1, ..., tr(A_m X_k) - b_m),\ G_{ij} = tr(A_i A_j). \tag{4.5}$$

in other words the matrix X_k is forced to solve the system using the residuals.

Convergence Speed

It is good to notice that the speed of convergence of the algorithm depends strongly on the geometry between the two sets. Another parameter that will count is the initial position, these two factors will be important aspects of the numerical analysis made in this work.

Figures 4.3 and 4.4 ilustrate how the angle between the sets will change completely the convergence speed.

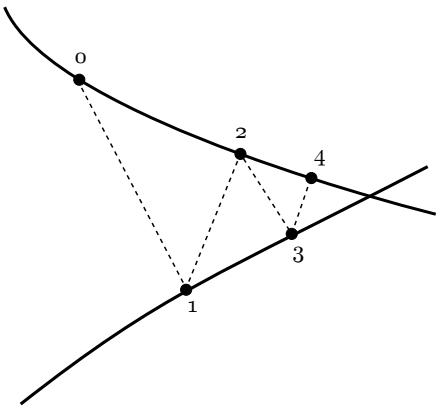

Figure 4.3: Ilustration of a fast convergence

4.2 Example

Determine the existence of a pair (x, y) such that

$$\begin{cases} x + y = 2 \\ x \geq 0,\ y \geq 0 \end{cases} \tag{4.6}$$

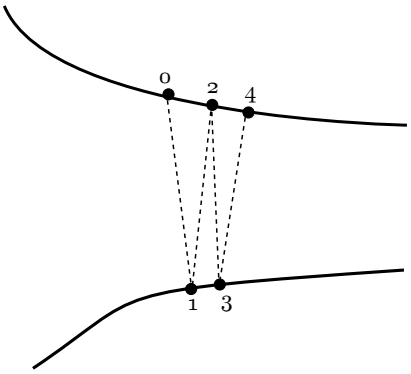

Figure 4.4: Ilustration of a slow convergence

Solving this problem is the same as finding the existence of a point that is in the straight line $x + y - 2 = 0$ and is in the non-negative quarter of the cartesian plane. In order to use the alternating projection algorithm, the problem must be translated into a problem of feasibility of PSD:

$$\begin{cases} \chi \succeq 0 \\ tr(A_i\chi) = b_i,\ A_i \in S^n,\ i = 1..m \end{cases} \tag{4.7}$$

where $\chi = \begin{bmatrix} x & 0 \\ 0 & y \end{bmatrix}$, $A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ and $b = 2$; the problem is in the right way and the projections can be used in the same way as stabilished before.

4.3 Implementation

The implementation of the algorithm itself needs the translation of the constraints of the problem

$$\begin{cases} -\Delta V_{max} \leq \Delta V_i \leq \Delta V_{max}, \forall i = 1..N \\ \sum_{i=1}^{N} \|\Delta V_i\|_1 \leq \sigma \\ \Delta \tilde{V}_i = \left(\frac{1+ecos(\nu_i)}{\dot{\nu}_i}\right) \Delta V_i, \forall i = 1..N \\ \tilde{X}(\nu) = T(\nu)X(t) \\ \tilde{X}(\nu_i) = \phi(\nu, \nu_1)\tilde{X}(\nu_1) + \sum_i \phi(\nu, \nu_i)\tilde{B}\Delta\tilde{V}_i \\ D_i = C(\nu_i)\tilde{X}(\nu_i), \forall i = 1..N \\ D_i \in S_D(H, V), \forall i = 1..N \\ Y_i \succeq 0, \ \forall i = 1..s \\ \gamma_i^p = \Lambda^*(Y_i), \ \forall i = 1..s \end{cases} \tag{4.8}$$

into the form

$$\begin{cases} \chi \succeq 0 \\ tr(A_i\chi) = b_i, \ A_i \in S^n, \ i = 1..m \end{cases} \tag{4.9}$$

The implementation of the algorithm can basically be divided into two steps: the decision variables choice and the process of building the matrices A_i and the vector b. The algorithm receives the data to build the problem in the form $tr(A_i\chi) = b_i, \chi \succ 0$ and from the problem it can start the iterative process, as the formulas for projection depend exclusively on A and b. The algorithm has three different stopping conditions and returns the control value (vector of speed impulse) in the end.

4.3.1 Details of Implementation: Construction of the iteration variable

For the example of section 4.2, it was clear that $\chi = \begin{bmatrix} x & 0 \\ 0 & y \end{bmatrix}$ should be the variable, but when we look at the problem it is not obvious what use as the iterative variable (χ) in the algorithm. Looking through the list of constraints, it is evident that some variables need to be in χ very clearly. As in the problem definition it is stabilished $\chi \succeq 0$, all the variables in the problem that are subjected to positivity constraint must be in χ. Note that, if χ is block diagonal, χ is PSD[1] if and only if every element of it is PSD.

[1] Positive Semidefinite

The saturation constraints can be translated into a positive definitness constraint. For example, using the block matrix:

$$M_x = \begin{bmatrix} \Delta V_{max} & \Delta V_x \\ \Delta V_x & \Delta V_{max} \end{bmatrix} \tag{4.10}$$

Making it semi-positive definite is equivalent to $(\Delta V_x)^2 \leq (\Delta V_{max})^2$ which is one of the inequality constraints.

The budget constraints can also be translated with the help of slack variables in the following way:

$$\begin{cases} Z_i + \Delta V_i = W_i^+ \\ Z_i - \Delta V_i = W_i^- \\ \sigma - \sum_i Z_i = W_0 \\ Z_i, W_i^+, W_i^-, W_0 \geq 0 \end{cases} \tag{4.11}$$

Which would produce 10 variables that need to be semi-definite positive and 7 equality constraints using these variables.

So, at first, the following matrix could be used as the iteration variable in the algorithm:

$$\chi = diag(M_x, M_y, M_z, Z_x, Z_y, Z_z, W_x^+, W_y^+, W_z^+, W_x^-, W_y^-, W_z^-, W_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6) \tag{4.12}$$

The first blocks are all composed by the consumption constraints and the last block are related to the polytope constraints. Each block will be analysed separately.

The first 3 blocks are related to the saturation constraints

$$M_x = \begin{bmatrix} \Delta V_{max} & \Delta V_x \\ \Delta V_x & \Delta V_{max} \end{bmatrix} M_y = \begin{bmatrix} \Delta V_{max} & \Delta V_y \\ \Delta V_y & \Delta V_{max} \end{bmatrix} M_z = \begin{bmatrix} \Delta V_{max} & \Delta V_z \\ \Delta V_z & \Delta V_{max} \end{bmatrix} \tag{4.13}$$

Following in the matrix blocks we have Z_x,Z_y,Z_z,W_x^+,W_y^+,W_z^+,W_x^-,W_y^-,W_z^-,W_0 all scalars and related to the consumption.

In the last 6 blocks there are the matrices $Y_1, .., Y_6$ associated with the polytopic

constraints.

This presents the big variable χ that is used in the problem and must be PSD (note that to a diagonal matrix be PSD it has to have all elements of its diagonal PSD as well).

The saturation constraints are solved in the way the variable X was built. To the budget constraints it is sufficient adding the 7 equality constraints presented in (4.11). The equality constraints which remain are the ones used to create the matrices $Y_1, ..., Y_6$.

4.3.2 Details of Implementation: Translation of the problem

In order to build the matrices A_i and the vector b_i, it is needed to build the equality constraints in the form

$$A_{eq}y = b_{eq} \tag{4.14}$$

The first constraint is

$$d_0 = 0 \tag{4.15}$$

then there is a set of constraints that are linked to the polytopic constraint:

$$\begin{aligned}
tr(Y_1 h_i) &= -p_x(i) + x_{max} * tc(i), \ i = 1, ..., 5 \\
tr(Y_2 h_i) &= -p_y(i) + y_{max} * tc(i), \ i = 1, ..., 3 \\
tr(Y_3 h_i) &= -p_z(i) + z_{max} * tc(i), \ i = 1, ..., 5 \\
tr(Y_4 h_i) &= p_x(i) + x_{min} * tc(i), \ i = 1, ..., 5 \\
tr(Y_5 h_i) &= p_y(i) + y_{min} * tc(i), \ i = 1, ..., 3 \\
tr(Y_6 h_i) &= p_z(i) + z_{min} * tc(i), \ i = 1, ..., 5
\end{aligned} \tag{4.16}$$

where h_i is the i-th Hankel matrix as described in Appendix A. These are 26 constraints, namely

$$p_x = C_x D(\nu) = C_x C X(\nu) = C_x C T X(t) \tag{4.17}$$

and the last 7 constraints are the constraints associated with the introduction of the slack variables in the saturation and budget constraints.

$$
\begin{aligned}
Z_i + \Delta V_i &= W_i^+,\ i = x, y, z \\
Z_i - \Delta V_i &= W_i^-,\ i = x, y, z \\
\sigma - \textstyle\sum_i Z_i &= W_0,\ i = x, y, z
\end{aligned}
\tag{4.18}
$$

These are the 34 equality constraints of the problem. These equations plus the 6 equations that define the 6 values of saturation (one maximum and one minimum for each direction) form 40 constraints. So, in order to complete these development, each equation will be transformed into a constraint in the form of

$$
tr(A\chi) = b \tag{4.19}
$$

where A is a matrix 32×32, χ is the iteration variable defined previously, and b is a scalar. Basically, this process is a reallocation of the terms of the equality and an example is given next.

Transformation of the equations

The following equation system:

$$
\begin{bmatrix} 4 & 2 & 8 & 20 \\ 12 & 14 & 2 & 30 \end{bmatrix} \begin{bmatrix} r \\ s \\ t \\ u \end{bmatrix} = \begin{bmatrix} 2 \\ 7 \end{bmatrix} \tag{4.20}
$$

with the constraint

$$
\begin{bmatrix} r & s \\ s & t \end{bmatrix} \succeq 0,\ u \geq 0 \tag{4.21}
$$

can be translated into

$$
x = \begin{bmatrix} r & s & 0 \\ s & t & 0 \\ 0 & 0 & u \end{bmatrix} \succeq 0 \tag{4.22}
$$

and

$$tr\left(\begin{bmatrix}4 & 1 & 0\\ 1 & 8 & 0\\ 0 & 0 & 20\end{bmatrix}\begin{bmatrix}r & s & 0\\ s & t & 0\\ 0 & 0 & u\end{bmatrix}\right) = 2, tr\left(\begin{bmatrix}12 & 7 & 0\\ 7 & 2 & 0\\ 0 & 0 & 30\end{bmatrix}\begin{bmatrix}r & s & 0\\ s & t & 0\\ 0 & 0 & u\end{bmatrix}\right) = 7 \tag{4.23}$$

4.3.3 Break Conditions

As computers have a finite precision, some tolerances were estabilished to use in the break points of the algorithm. This tolerances are: tol_vp and tol_cons. This way, in fact, our problem is:

$$\begin{cases} min(eig(X) \geq tol\_vp \\ \|tr(A_i X) - b_i\| \leq tol\_cons,\ A_i \in S^n,\ i = 1..m \end{cases} \tag{4.24}$$

The study of how the value of these tolerances influence in the resulting control is made in chapter 7. There are three break conditions in the algorithm: Convergence, maximum number of iterations and a prediction of convergence. The convergence tests if the variable satisfies the conditions of the problem, the maximum number of iterations is a guarantee that the algorithm will stop even if it does not converge and the prediction is made with a linear regression to try to estimate how many iterations will be needed until it converges.

Prediction

[9] Suggests that a form of number of iterations prediction can be used as a break condition. As the convergence evolution is strictly monotonic decreasing, a linear regression can be made to determine how many iterations are needed to arrive at a certain value. This convergence is considered as the difference between the projections, and tol_fro is defined as the desired value that would imply into the convergence.

The linear regression will give a number of iterations that either we can wait until it converges or stop the algorithm because it is known that it will not converge. One option is to compare the number of iterations predicted with the maximum number of iterations defined for the algorithm, if it is higher in any moment the algorithm can stop for it will

necessarily stop because of the maximum number of iterations.

4.3.4 Pseudocode

Data: initial guess,environment,saturation,fuel budget,tolerances,polytope

input: state,anomaly

Result: control

X = initial guess;

make_A();

make_b();

while *true* **do**

  $Y = Projection(X, C)$;

  $X = Projection(Y, D)$;

  $iter++$;

  if $(trace(A_i X) - b_i < tol\_cons,\ i = 1 : I$ ***and*** $min[eigenvalue(X)] > tol\_vp)$

  then

    break;

  else

    if $iter > max\_iter$ **then**

      break;

    else

      Convergence Prediction

    end

  end

end

extract_control();

Algorithm 1: Alternate Projection Algorithm

4.4 Initial Guess

The initial guess has a great importance in the result of the algorithm when it comes to the consumption level. The algorithm starts with a matrice X_0 and tries to find a matrice X^* which is in the intersection of two sets: The cone of the PSD matrices and the set

described by the equality constraints $tr(A_i\chi) = b_i$. The algorithm takes the matrice X_0 and does a sequence of projections into both sets until it converges to the intersection or at least the point of minimal distance between the sets. Figure 4.5 ilustrates one situation for the algorithm.

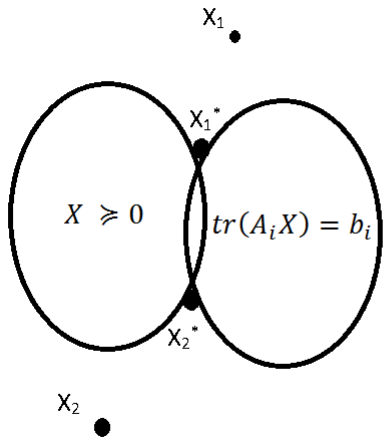

Figure 4.5: Example of the topology the algorithm works in

If X_1 is the initial guess, the algorithm will converge to X_1^*, however if X_2 is the initial guess the algorithm will converge to X_2^*. Figure 4.6 ilustrates another situation for the algorithm.

For a situation with disconnected sets the initial guess is even more important as it can converge to really different regions in the space. This leads to the conjecture that if we search for a global optimal we have to search the optimal initial guess as well.

It is important to see that the constraints depend on time, anomaly, position and actual speed of the aircraft as well. This way there can be different geometric situations for consecutive problems.

At first, to generate the initial guess, some logical choices were made like what is called the Cold Start and the Warm Start. The Cold start consists in using everytime the same initial guess, therefore starting always from the same point. The Warm start, however, consists in using the last solution generated by the algorithm as the initial guess for the following control computation. This two were the only initialisation strategies used in in [9] and in [10].

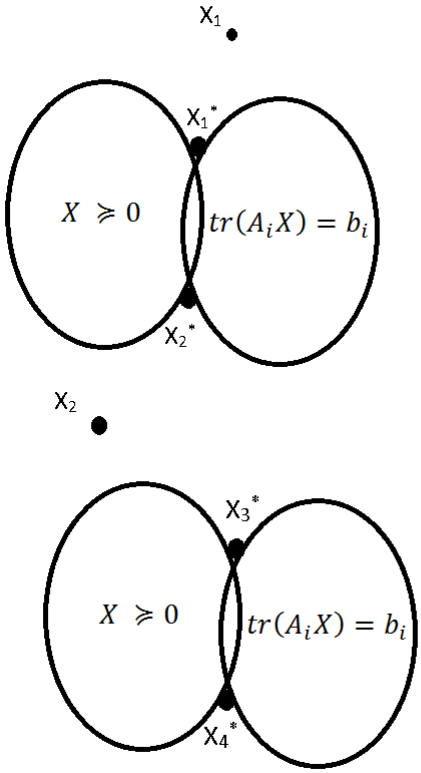

Figure 4.6: Another example of topology the algorithm work in

4.5 Justification of the use of the Alternating Projections Algorithm

The strongest reason to use the Alternating Projections Algorithm instead of a optimization solver in this work is the computational cost of the algorithm. Given that sattelites uses Leon kind of boards and that the memory available for the control algorithm in this application is in the order of hundreds of kilobytes, there is no way to use an already made optimization solver in the satellite. Besides that, there is a time constraint as the satellite is a time critical embedded system and uses a TDMA algorithm to manage its processor, so the algorithm has to give a control in the order of miliseconds.

Nowadays the solution used in this kind of application is to calculate the control remotely and send it to the satellite, however this does not give much independence to the system. So, considering all the constraints we have to deal with when it comes to embed a control algorithm, the Alternating Projections Algorithm is a great option face to optimization solvers.

The next chapter will, then, analyse how we can improve this solution in terms of consumption and feasibility so that it approaches its efficiency to the level of a optimization solver.

4.6 Conclusion

In this chapter the Alternating Projections Algorithm was fully described, so from now on this work will treat all the aspects related to using it as the solution to the problem of rendezvous. Next chapters will describe deeper the issues with the initial guess, the feasibility of the problem as well as a numerical analysis from the iterative point of view of the whole process.

Chapter 5

Initialisation of the Alternating Projections Algortihm

As showed in the last chapter, the initial guess in the algorithm is really relevant to the results. If we compare only the two "classical" initialisations used in the work of Arantes and Louembet in [10] we see already that the consumption in the warm start is really lower than the cold start. In this chapter we study the influence of the initial guess in the final consumption of the mission. It is also compared the results given by the Alternating Projections Algorithm to the results given when using an optimization solver.

Intrigued by the results given by an optimization solver, we did not know if it was possible to arrive at such low consumption using the alternating projection algorithm. So in this chapter:

1. The results obtained by an optimization solver (SeDuMi) are presented.
2. A metaheuristic based on the process of Simulated Annealing was used to find out if there is any initial guess that can result in the optimal consumption.
3. Another method of initialisation was proposed to be an option to the cold and warm start.

5.1 Simulator

The simulator presented in Figure 5.1 was used to evaluate the performance of the algorithm used. It was conceived by Kara-Zaïtri [11] and consists in a *Simulink* Block Diagram which simulates separately the trajectories of the follower and the target using two non

linear equivalent models based on Gauss equations. Although it was used the linearized equations of Tschauner-Hempel when modeling the system to our control system, this simulator considers the angles i, ω and Ω, the non-homogeneous geometry of the Earth, the atmospheric drag and the solar activity.

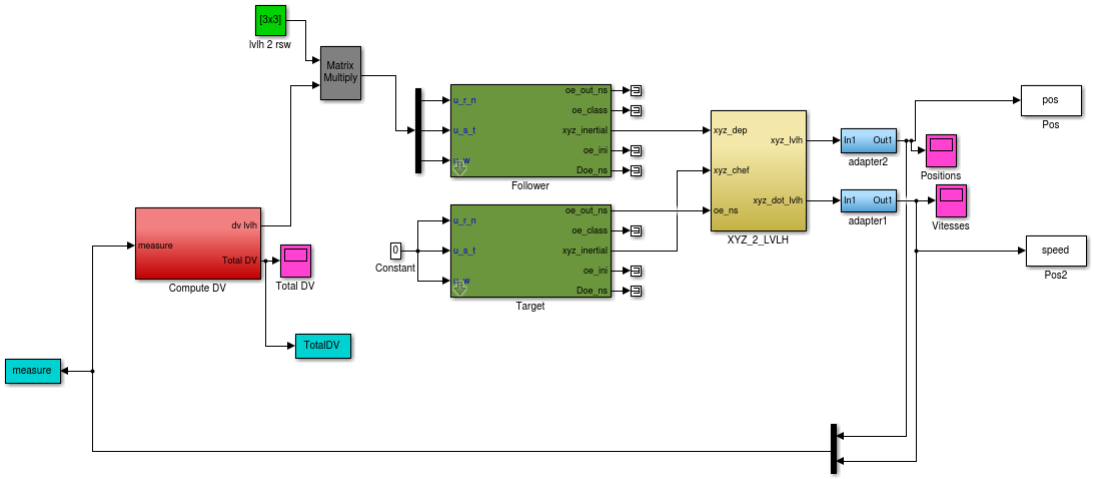

Figure 5.1: Simulink simulation scheme used in this work

The green blocs in Figure 5.1 are responsible for simulating the orbital elements evolution, while the red one is the block which will contain our control algorithm.

For the simulations an initialisation is done before to set a scenario. It is decided as well the frequency we will apply the control as well as the total time of simulation, initial position of the spacecrafts and total budget of the follower.

The simulation keeps track of the whole trajectory traced by the follower in the LVLH (Local Vertical Local Horizontal) framework situated in the mass center of the target.

5.1.1 Scenario of simulation

The scenario of simulation used through the rest of the work is presented in Table 5.1.

Parameter	Value	Parameter	Value
a[km]	6763	Target Mass [kg]	462949
e	0.0052	Follower Mass [kg]	20000
i[°]	52	Drag Coeff Target	3
ω[°]	0	Drag Coeff Follower	2.274
Ω[°]	0	Target Surface [m^2]	1703
t_0	0	Follower Surface [m^2]	50
X_0[m,m/s]	[5;5;5;0;0;0]	max_it	100
$\Delta\ V_{max}$	0.26	Tol_vp	-0.001
σ [m/s]	6763	Tol_cons	0.001
Box	[-40,40,-40,40,-40,40]	Time Between Controls [s]	800

Table 5.1: Data for the scenario simulated

5.2 Classic Initialisations

5.2.1 Cold Start

The cold start used in [10] uses the zero matrix or the identity as initial guess. This gives an almost constant behaviour to the algorithm when we see the iterations graphics. As it starts always at the same place it tends to converge always to a similar solution, unless the set given by the constraints changed a lot from one control calculation to another. The cold start is less efficient when compared to the warm start as it is shown during the numerical analysis in Chapter 7, producing solutions that have a higher consumption and less smooth trajectories. The trajectory traced by the Cold Start is presented in Figure 5.2.

5.2.2 Warm Start

The warm starts consists in using always the solution of the last control calculation as the initial guess. This way it will start from the point the algorithm stopped last time, in other words, it starts from the latest intersection of the two previous sets. If it is considered that the problem does not change drastically from one control calculation to the next, this initial guess should be really near the intersection of the two sets. The intuitive idea

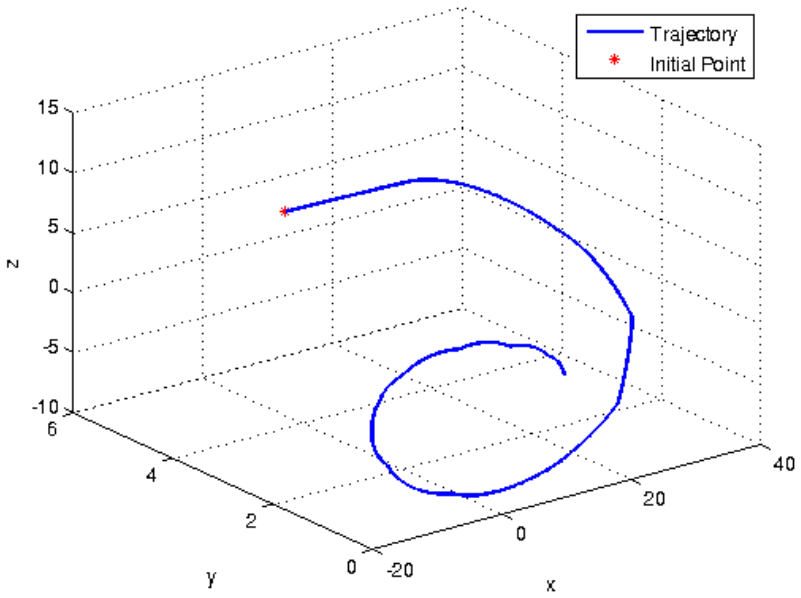

Figure 5.2: Trajectory for Cold start

is to try to maintain always the same trajectory, considering that the perturbations are small, continuing in the same orbit should cost little. The results show that this is a great initial guess indeed, but as will be shown in this chapter, the solution of the warm start is yet far from the optimum. For the first control calculation it is used a cold start, taking the zero or the indentity matrix as initial guess. The trajectory traced by the Cold Start is presented in Figure 5.3.

Figure 5.4 gives a better insight at how different are the trajectories given by the solutions of warm and cold start. In blue it is showed the trajectory resulted of warm start control. In red is the trajectory that would have resulted from a cold start in the same situation. It is possible to note that the solution of cold start tends to diverge from the actual trajectory, while the warm start tends to maintain its orbit.

5.3 Current Point Approach

Another alternative to the initial guess is to use the state mesure and try to build an initial guess matrice as though the current state were the solution. With this we try to force the algorithm to converge to the nearest solution of the instanteneous orbit, which would cost less than the others. This is reasonable since a direct control in speed is used,

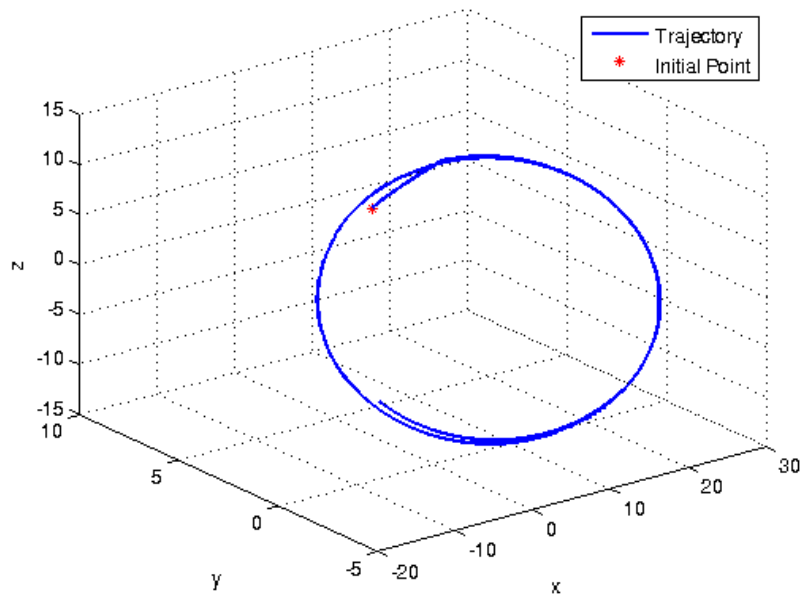

Figure 5.3: Trajectory for Warm start

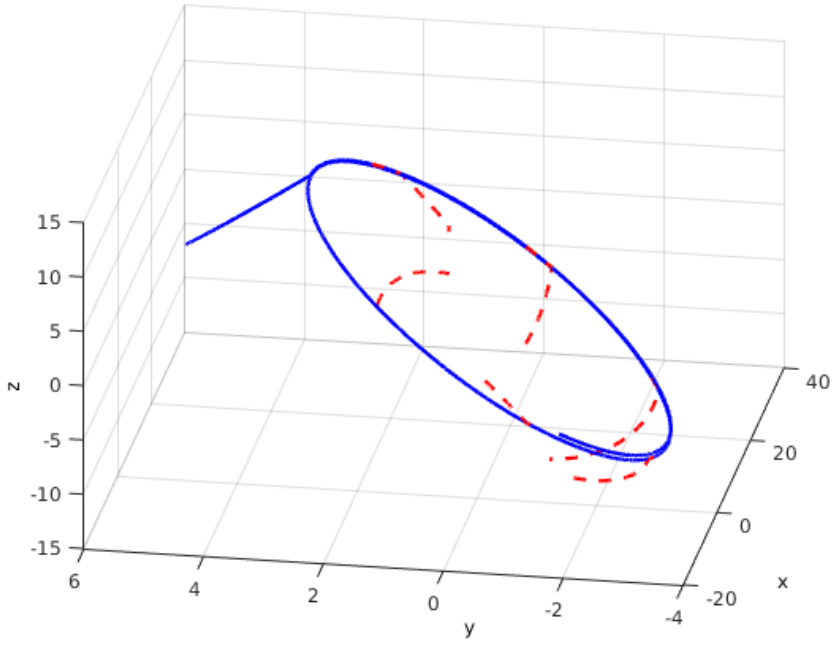

Figure 5.4: Trajectory Comparison Warm Start and Cold Start

so if it starts with the current speed we could find a trajectory that does not need a great impulse.

To understand the real difference between this approach and the warm start, let the following situation: From the last control application the spacecraft suffered a great per-

turbation in terms of speed, so now it was driven away from the orbit planned. If it uses the Warm Start, it will try to come back to its last trajectory, which is far away right now. The lowest cost solution would be to try to find a trajectory that passes through the point it is at now, requiring the smallest control possible.

The mathmetical development is presented next.

5.3.1 Contruction of the initial guess

Let $X_0(t_k) \in \mathbb{R}^6$ the state measure at the instant t_k and ν_k the true anomaly at the instant t_k. The objective is to calculate $Y_0 \in \mathbb{R}^{32x32}$ starting point for the Alternate Projections Algorithm. In order to build Y_0 it is needed to analyse the structure of it: Y_0 is a diagonal block matrix that comes from the problem constraints.

$$Y_0 = diag(M_x, M_y, M_z, Z_x, Z_y, Z_z, W_x^+, W_y^+, W_z^+, W_x^-, W_y^-, W_z^-, W_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6) \tag{5.1}$$

The first blocks are all composed by the consumption constraints and the last block are related to the polytope constraints. Each block will be analysed separately.

The first 3 blocks are related to the saturation constraints

$$M_x = \begin{bmatrix} \Delta V_{max} & \Delta V_x \\ \Delta V_x & \Delta V_{max} \end{bmatrix} M_y = \begin{bmatrix} \Delta V_{max} & \Delta V_y \\ \Delta V_y & \Delta V_{max} \end{bmatrix} M_z = \begin{bmatrix} \Delta V_{max} & \Delta V_z \\ \Delta V_z & \Delta V_{max} \end{bmatrix} \tag{5.2}$$

Since the algorithm gives the closest solution to the initial point and, the objective is to minimize the variable ΔV for the 3 axes, the best is to set them 0 at the initial point. The value of ΔV_{max} is fixed when the problem is defined, so it will be set to this value. This way, the matrices will be

$$M_s = \begin{bmatrix} \Delta V_{max} & 0 \\ 0 & \Delta V_{max} \end{bmatrix}, s = x, y, z \tag{5.3}$$

Following the blocks in matrix Y_0, there is Z_x,Z_y,Z_z,W_x^+,W_y^+,W_z^+,W_x^-,W_y^-,W_z^- all scalars and related to the consumption. For the same reason presented before, they will be set to 0 trying to minimize them.

The next block is W_0 which is the total budget of the mission and is a fixed value from the problem definition.

In the last 6 blocks there are the matrices $Y_1, .., Y_6$ associated with the polytopic constraints. It is here that the measured state value will be used. To build this matrices all the constraint definition process has to be followed. First of all to pass the measure from time to true anomaly, this transformation follows

$$\tilde{X}_0(\nu_k) = T(\nu_k)X_0(t_k) \tag{5.4}$$

Now the state is in function of the true anomaly, the parameters $D_0(\nu_k)$ associated with this state can be calculated:

$$D_0(\nu_k) = C(\nu_k)\tilde{X}_0(\nu_k) \tag{5.5}$$

From this parameter, the polynomial coefficients associated to the motion in each axe of the satellite.

$$p_x = C_x D(\nu_k), \quad p_y = C_y D(\nu_k), \quad p_z = C_x D(\nu_k) \tag{5.6}$$

From the polynomial coefficients the polynomials that we will imposed to be non-negative can be explicited as follows:

$$\Gamma_i = -h_{i,1}[P_x] - h_{i,2}[\tilde{P}_y] - h_{i,3}[P_z] + v_i T \tag{5.7}$$

The politopic constraints that were variable of time have to be transformed into anomaly as well. To give an example of how this polynomials look like we can explicit Γ_1 which will be associated to the constraint Y_1

$$\Gamma_1 = -p_x + x_{max} * t_i, \quad t_i = [(1+e), 0, 2, 0, (1-e)]^t \tag{5.8}$$

where x_{max} is a scalar in the problem defined constraint from the polytope definition. The next step is to transform the polynomial constraints into positive definiteness ones. This is made with the help of the Λ operator defined in [14]. In fact, it is not used the operator itself, but with the property:

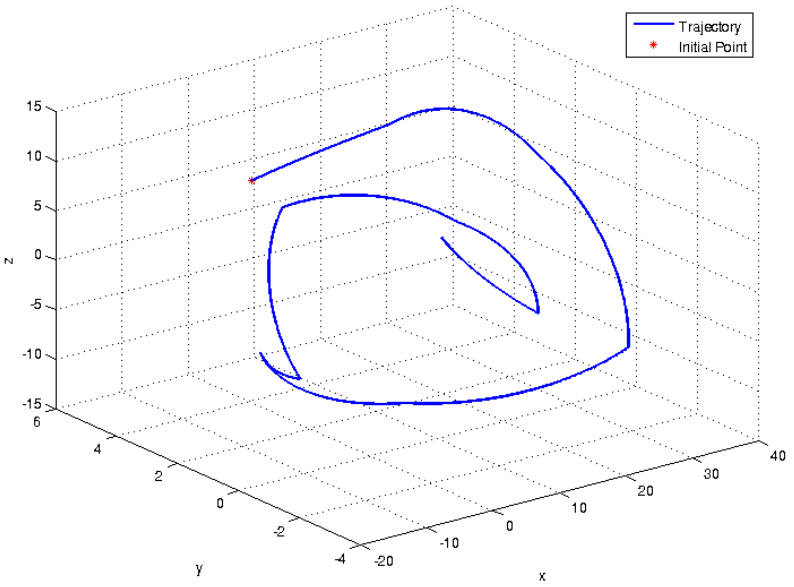

Figure 5.5: Simulated trajectory for the initial point approach.

$$< \Gamma, \Lambda^*(Y) > = < \Lambda(\Gamma), Y > \tag{5.9}$$

it is possible to write the matrice $\Lambda(\Gamma)$ explicitly. In a general form the matrice will be:

$$Y_i = \begin{bmatrix} \Gamma_i(1) & \Gamma_i(2) & \Gamma_i(3) \\ \Gamma_i(2) & \Gamma_i(3) & \Gamma_i(4) \\ \Gamma_i(3) & \Gamma_i(4) & \Gamma_i(5) \end{bmatrix}, \quad i = 1, .., 6 \tag{5.10}$$

and with this the last elements of the Initial Point Matrix are defined.

Results

To compare this approach to the others used previously, the same scenario presented before was used. The trajectory that resulted for a period of 1000 seconds between the control aplications is presented in Figure 5.5.

The consumptions for comparison with the cold and the warm approaches are presented in Table 5.2. which shows that this approach is not optimal.

Period [s]	Consumption		
	cold start	warm start	current point start
10	0.1056	0.0197	0.1687
100	0.0903	0.0222	0.1488
500	0.0677	0.0321	0.1117
600	0.0656	0.0340	0.1047
700	0.0659	0.0361	0.0908
800	0.0652	0.0378	0.0983
900	0.0662	0.0397	0.1028
1000	0.0694	0.0420	0.0981
1100	0.0723	0.0447	0.0981

Table 5.2: Consumption for each initial guess strategy for each control frequence

5.4 Optimization Process

To solve the problem optimally it was used the Yalmip language with the solver SeDuMi. YALMIP is a language for advanced modeling and solution of convex and nonconvex optimization problems and it is implemented as a free toolbox for MATLAB.

The main motivation for using YALMIP is that all the implementation could be used with no changes. The language is consistent with standard MATLAB syntax, thus making it extremely simple to use for anyone familiar with MATLAB. The problem constraints were written in MATLAB language rebuilding the problem in the LMI (PSD) form. Then the problem was solved with the objective function to minimize the consumption (sum of speed variations applied) by the solver SeDuMi.

The controls given by the optimization process are showed in Table 5.3 and using the warm start in the Table 5.4. The results when comparing the SeDuMi solution and the Warm Start are similar.

It can be seen that the solutions given by the optimization process are similar to the warm start, however the optimal control uses only the actuator in X, this way reducing the consumption. Basically the warm start is not far from the optimum solution, it just uses more effort than necessary. The trajectory presented by the optimal solution is really similar to the Warm start as well and it is possible to see it in Figure 5.6.

Comparing the trajectory in Figure 5.6 with the one presented in Figure 5.3 can be concluded that the optimal solution gives a trajectory really similar to the warm approach. The intuitive idea is that the optimal solution is capable of giving an orbit that is nearer

X	Y	Z
0.0107	0	-0.0083
-0.0005	0	0
-0.0003	0	0
-0.0002	0	0
-0.0003	0	0
-0.0005	0	0
-0.0007	0	0
-0.0008	0	0
-0.0006	0	0
-0.0004	0	0
-0.0001	0	0
-0.0003	0	0

Table 5.3: Optimal Controls given by the solver SeDuMi using Yalmip

X	Y	Z
0.0106	0.0046	-0.014
-0.0005	0	0.0005
-0.0003	0	0.0002
-0.0002	0	0.0001
-0.0003	0	0.0003
-0.0005	0	0.0004
-0.0008	0	0.0006
-0.0008	0	0.0005
-0.0006	0	0.0005
-0.0004	0	0.0003
-0.0001	0	0.0001
-0.0003	0	0.0002

Table 5.4: Controls given by the warm approach

to the bounds of the polytope, thus reducing the consumption.

The consumption results comparing warm start, cold start and optimal solutions are presented in the Table 5.5. The column Period refers to the period between two applications of the control. As shown in the work of Arantes and Louembet [10] this parameter (period) can be really important to the consumption in the sense that a short period implicates in many but low controls whereas a great period implicates in fewer but higher controls. This period is *a priori* constant through all the mission.

What leads to a further investigation is that SeDuMi also uses a external path algorithm. This means that *a priori* it would be possible to reach the same solution using the Alternating Projections Algorithm. However to reach this solution it would have to be

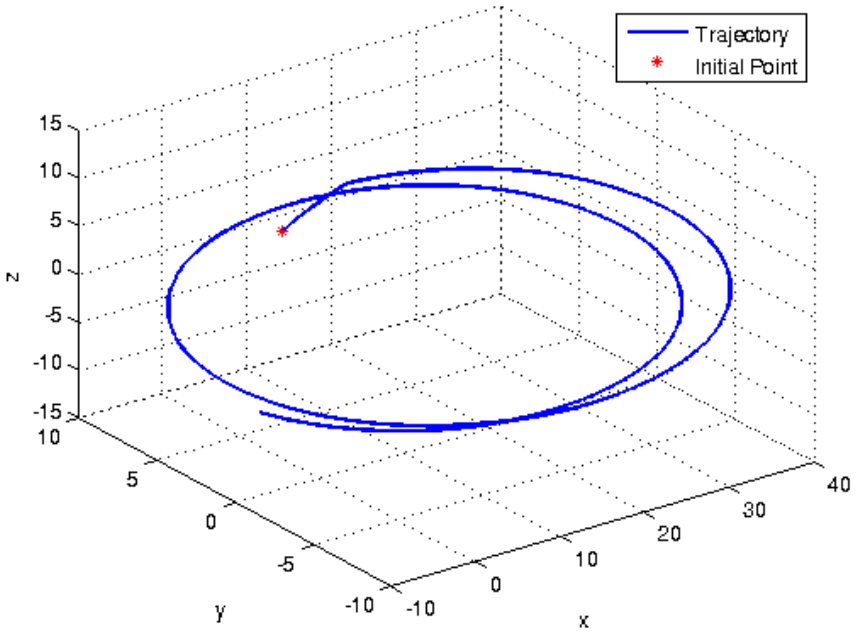

Figure 5.6: Simulated Trajectory using Optimal Controls

used a different initial guess that would lead to the "right" path and, therefore, the optimal solution. In order to search for this *Optimal Initial Guess* is that it was used a metaheuristic varying the initial guess to analyse the solution that it gives. This metaheuristic is presented in the next section.

5.5 Simulated Annealing

In this section it is presented the use of the Simulated Annealing algorithm to know if it is possible to reach the same results produced by the optimization process using the Alternating Projections Algorithm. Note that the Simulated Annealing will not substitute the Alternating Projections, but it will work producing intial guess to the Alternating Projections, searching the initial guess that will produce the best result in terms of consumption.

It is important to note as well, that the Simulated Annealing will not be used online, in the processor of the satellite. This is a analysis made by this work, using it to prove how near the optimality it is possible to get without using a optimization algorithm.

Simulated annealing (SA) is a generic probabilistic metaheuristic for the global op-

Period (s)	Warm	SeDuMi	Cold
10	0.0196	0.0167	0.1044
100	0.0214	0.0166	0.0903
500	0.0318	0.0173	0.0677
600	0.0338	0.0191	0.0668
700	0.0363	0.0215	0.0660
800	0.0375	0.0236	0.0676
900	0.0398	0.0257	0.0661
1000	0.0419	0.0284	0.0695
1100	0.0447	0.0359	0.0727

Table 5.5: Consumption Comparison between Cold, Warm and Optimal controls.

timization problem of locating a good approximation to the global optimum of a given function in a large search space. For certain problems, simulated annealing may be more efficient than exhaustive enumeration — provided that the goal is merely to find an acceptably good solution in a fixed amount of time, rather than the best possible solution.

The name and inspiration come from annealing in metallurgy, a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects. Both are attributes of the material that depend on its thermodynamic free energy. Heating and cooling the material affects both the temperature and the thermodynamic free energy. While the same amount of cooling brings the same amount of decrease in temperature it will bring a bigger or smaller decrease in the thermodynamic free energy depending on the rate that it occurs, with a slower rate producing a bigger decrease. This notion of slow cooling is implemented in the Simulated Annealing algorithm as a slow decrease in the probability of accepting worse solutions as it explores the solution space. Accepting worse solutions is a fundamental property of metaheuristics because it allows for a more extensive search for the optimal solution.

As something to search for a possible *Optimal Initial Guess* was needed, this metaheuristic was ideal to lead the search. It is really good to explore the space and it is robust enough to avoid falling into local optima in the case of the geometry is hard.

5.5.1 Pseudocode

Data: Problem, Optimal Solution

Result: Initial Guess that gives the Optimal Solution (Best Result in the List)

```
Define initial temperature;
Define final temperature;
Define law to update temperature;
for cycles do
    x = generate_initial_guess();
    fx = objective_function(x);
    if worse soluntion then
        generate probability of acceptance based on temperature;
        if rand < probability of acceptance then
            accept new solution;
        else
            do not accept new solution;
        end
    else
        accept new solution;
    end
    if new solution accepted then
        update list of solutions;
    end
    update temperature;
end
```

Algorithm 2: Simulated Annealing Algorithm

An intuitive idea for the algorithm is that it starts with an initial guess. Than it generates an initial guess in the neighbourhood of the first using a random method (but conserving the structure of the matrix) and evaluates it. If the new initial guess is better than the old one it is accepted and put in the list of solutions, however in the case that it is worse than the old one there is also a probability of it being accepted. This scheme with a probability of acceptance enables the algorithm to exit local optimas. As the cycles pass, the temperature is decreased and the probability of acceptance is decreased as well. So

when we approach the last iterations, the temperature converges to 0 and the probability converges to 0 in the same way, considering just improvements in the objective function.

This process is random and offers no optimal garanties, however the optimal values are obtained from the Yalmip analysis (using SeDuMi) and the metaheuristic ran until it finds an initial guess that gives a result near enough to the optimum. This means that from the 100 problems solved with this method, for 90 it found an *Optimal Initial Guess* in the first try; however for a decade of problems it had to run some more times the metaheuristic in order to find it.

5.6 Results Analysis

The results show that it is possible indeed to find the optimal value with the alternating projections algorithm. However there is no unicity of the optimal initial guess and we were not able to find a logical process of building it as it did not ressembled either the strategies we used before. The consumption results are presented in Figure 5.7.

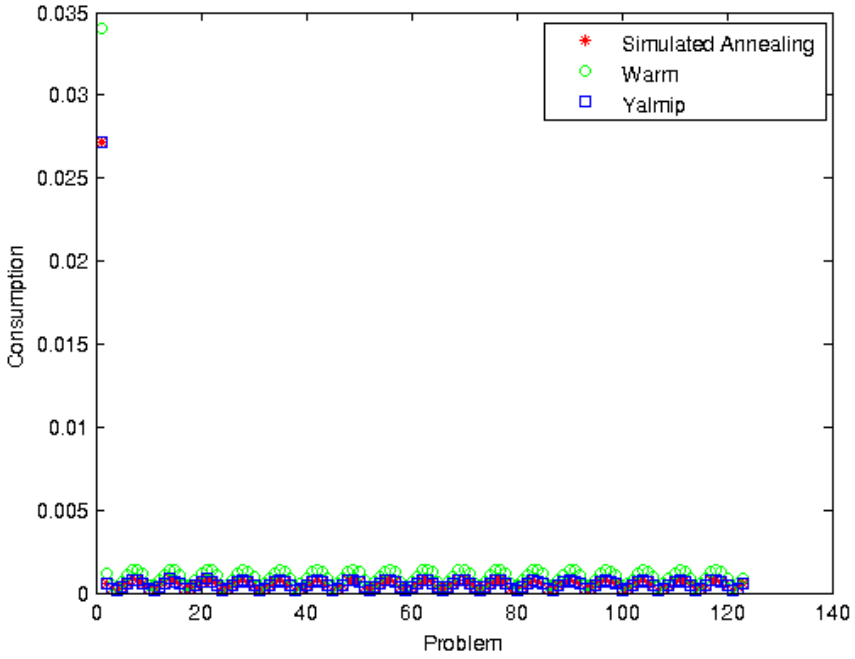

Figure 5.7: Consumption Comparison between Warm, Optimal and Simulated Annealing

The simulated annealing was able to find initial guesses that gave a solution near enough to the optimum. Figure 5.8 shows how far the warm aproach is from the optimal

as well as shows that it is possible to arrive at arbitrarily near the optimal solution using the Alternating Projection Algorithm if it uses the good initial guess. These results show that there is yet at least a 40% margin to gain in consumption when using the alternating projections algorithm when compared to the optimal solution. This margin stays open for future works to try to find a logical process that generates better initial guesses than the warm start or to at least know how much is the loss using this strategy when compared to the use of an optimization process.

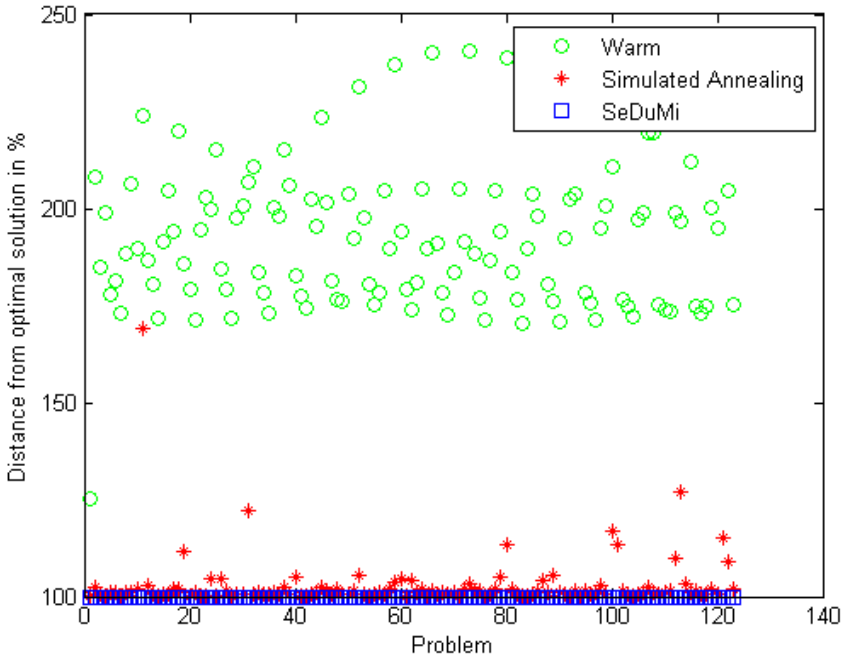

Figure 5.8: Relative Comsumptions Comparison

Chapter 6

Feasibility of the Problem

With the addition of the saturation and budget constraints to the problem, it is not always possible to find a feasible solution to the problem. Two situations can happen here, either there is need of higher actuation than our actuators can provide or there is not enough fuel to apply the control. In these situations it would be nice if there was a test to know if the algorithm can find a feasible solution or not.

Saying that there is a solution to the problem is the same thing as saying that the two sets have an intersection. In other words, to study the feasibility of the problem is the same as studying the topology of both sets. In order to do such study we need to decide which form of the problem will be use. In this chapter the work is always in the space of D (the set of 6 variables that describe the space of trajectories) and with the problem written as polynomials in D that need to be positive definite in respect to $\omega = tan(\frac{\nu}{2})$. There are basically three objectives in this chapter: Create a simple test to know if there is intersection between the sets, find a better way to decide when to apply the control rather than fixed time intervals and, study the topology where the problem is solved (if the sets are connected or not and other characteristics).

6.1 Feasibility of a state

The problem can be built in a way that for each vector D there is a test to say if it respects the constraints of the problem. In other words, the vector D defines a trajectory and, if this trajectory lay inside the polytope forever than it is an admissible trajectory. Another way to see this is to build the polynomials defined in Chapter 2 to see if, for the

vector D, it is positive definite.

For the algorithm, it was used the criterion of [14] to see if the polynomials were positive definite or not. Another way to see if the polynomial has this characteristic is the following: Let $p(x)$ be a polynomial of even degree, if it has only imaginary roots then either $p(x) > 0$ or $p(x) < 0$ $\forall x \in \Re$. So as our six polynomials are two of degree 2 (the ones related to the axis Y) and four of degree 4 (the ones related to the axis X and Z), if there is a criterion that tells when the roots are all imaginary based on the coefficients, the same criterion could be used to say when they are positive or negative definite.

The coefficients of the polynomials are functions of the parameters $[d_0, d_1, ..., d_5]$. This way, each vector D can be related to a polynomial and this criterion can be used to know if it is positive definite or not.

For degree 2 there is Bhaskara's Discriminant that is well know. For degree 4 there is a simular structure. Appendix C shows the process for polynomials of degree 4.

The conditions presented were used based on the polynomials. The objective is to have a positive definite polynomial of degree 4, so the test used was:

- $\Delta > 0$
- $P > 0$ or $\delta > 0$

these conditions are all functions of the variables $[d_1, d_2, d_3]$. So was made a grid of points in the space made by these 3 variables to find the region that satisfies the conditions and is positive.

It is known that the movement in Y is decoupled of the movement in X and Z. From the vector $[d_0, d_1, ..., d_5]$ the parameters d_4 and d_5 define all the trajectory in Y. Besides that, d_0 is set to 0 as a problem constraint (to take only periodic trajectories). So d_1,d_2 and d_3 define the trajectory in the plan $X - Z$. As the two problems are analogue here it only presents results for the trajectory $X - Z$.

Feasible Region

We can define a region that contains all the vectors $[d_1, d_2, d_3]$ that are a solution to our problem, based on the contraints for a quartic function be positive definite. This region is presented in Figures 6.1, 6.2 and 6.3

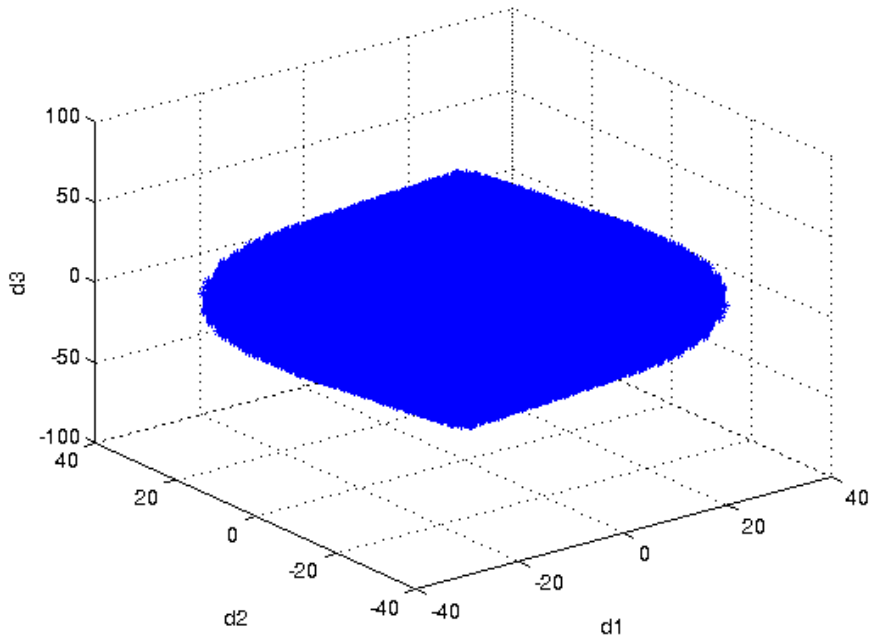

Figure 6.1: Region of feasible solutions

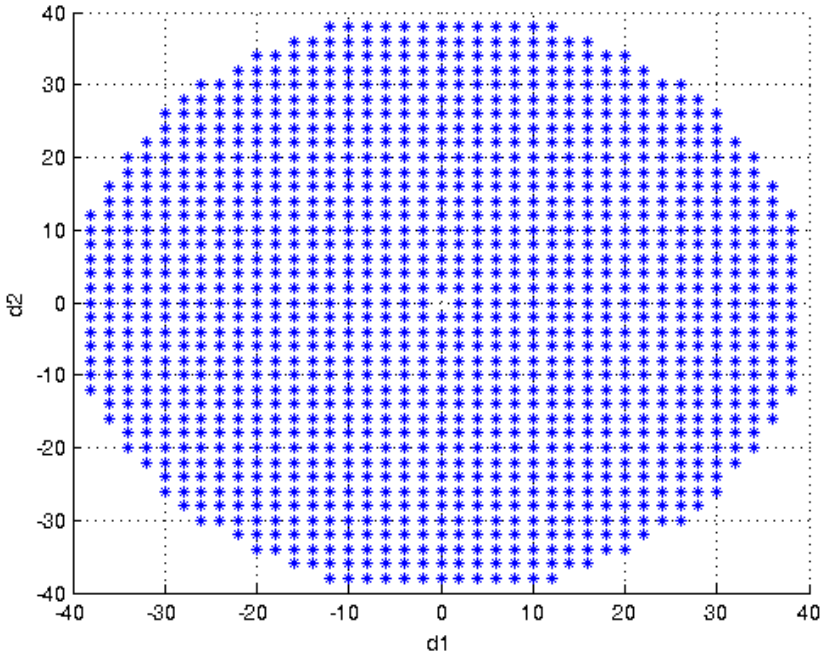

Figure 6.2: Top view of the region of feasible solutions

This results are for a problem with 0 excentricity. The case for excentricity greater than 0 will be discussed later in this chapter.

The region ressembles a diamond form, the result of a revolution of the polygone shown

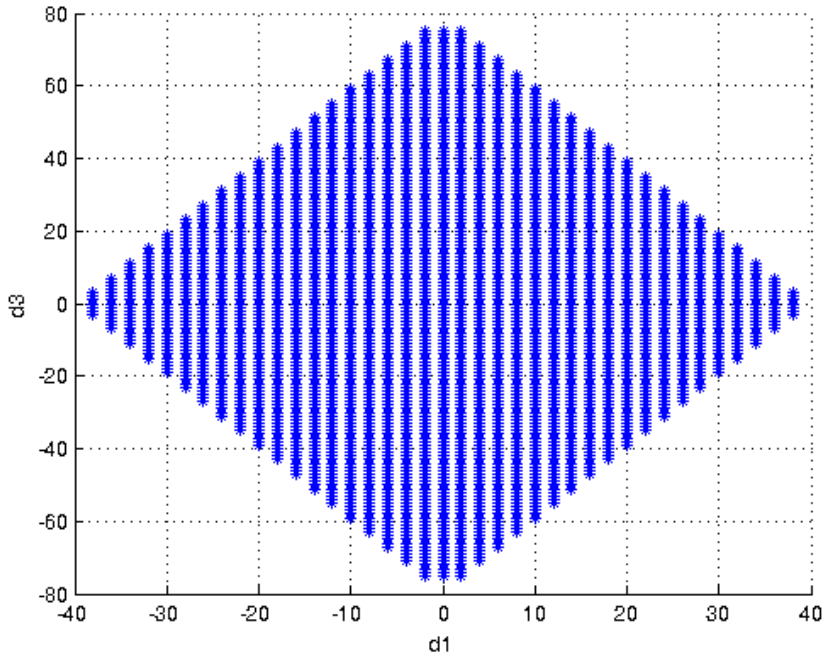

Figure 6.3: Front/Lateral view of the region of feasible solutions

in Figure 6.3. It is possible to see as well, that for a fixed d_3 value there is a circular figure in $d_1 \times d_2$. Figure 6.4 shows the cuts for some values of d_3.

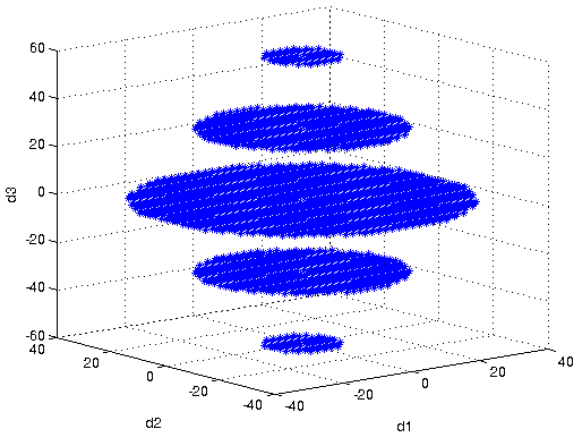

Figure 6.4: Cuts of the region for some d_3 values.

It can be seen as well that the radius of the circles has a linear behaviour with d_3. Analyzing this results and Equation (6.1) it is possible to conclude that this region can be modeled by a superquadric. Taking d_0 equals to 0 and varying the anomaly (ν) between 0 and 2π, if the direction, for example, x is fixed less than a x_{max} there is a constraint on d_1, d_2 and d_3 that will result in a circle for $e = 0$. So the results given by the grid in the

space D and the equations are consistent.

$$\begin{aligned}
\tilde{x}(\nu) &= (2 + e\cos\nu)(d_1\ \sin\nu - d_2\ \cos\nu) + d_3 + 3\,d_0\,J(\nu)(1 + e\cos\nu)^2 \\
\tilde{y}(\nu) &= d_4\ \cos\nu + d_5\ \sin\nu \qquad\qquad ,\nu \geq \nu_0 \\
\tilde{z}(\nu) &= (1 + e\cos\nu)(d_2\ \sin\nu + d_1\ \cos\nu) - 3\,e\,d_0\,J(\nu)\sin\nu(1 + e\cos\nu) + 2\,d_0
\end{aligned} \tag{6.1}$$

It is trivial to see how Equations (6.1) originate the circles. For $e = 0$ it is possible to write the equations for $x_{min} \leq x \leq x_{max}$ and it is analogue to z. Then take d_3 fixed, so the constraints will be on d_1 and d_2 in a similar way to figure 6.3. Taking the intersection for all values of anomaly ($\nu \in [0, 2\pi]$) it will generate the circle observed in 6.4.

In mathematics, the superquadrics or super-quadrics (also superquadratics) are a family of geometric shapes defined by formulas that resemble those of ellipsoids and other quadrics, except that the squaring operations are replaced by arbitrary powers. They can be seen as the three-dimensional relatives of the Lamé curves ("Superellipses").

The superquadrics include many shapes that resemble cubes, octahedra, cylinders, lozenges and spindles, with rounded or sharp corners. Because of their flexibility and relative simplicity, they are popular geometric modeling tools, especially in computer graphics.

The basic superquadric has the formula

$$|x|^r + |y|^s + |z|^t = 1 \tag{6.2}$$

where r, s, and t are positive real numbers that determine the main features of the superquadric. Namely:

1. less than 1: a pointy octahedron with concave faces and sharp edges.

2. exactly 1: a regular octahedron.

3. between 1 and 2: an octahedron with convex faces, blunt edges and blunt corners.

4. exactly 2: a sphere

5. greater than 2: a cube with rounded edges and corners.

6. infinite (in the limit): a cube

Each exponent can be varied independently to obtain combined shapes. For example, if $r = s = 2$, and $t = 4$, one obtains a solid of revolution which resembles an ellipsoid with round cross-section but flattened ends. This formula is a special case of the superellipsoid's formula if (and only if) $r = s$. To a deeper knowledge of superquadrics [2] is a good reference.

The importance of finding a model to the *Feasible Region* is that a simple test can be made to know wheter or not a point is inside of it. The superquadrics make it easy, as to know if a point is in its interior it is sufficient to test if a specific expression is superior to 0 for that point. In this case the region defined for $e = 0$ can be modeled by a superquadric with $r = s = 1$ and $t = 2$ resulting in the shape presented in Figure 6.5.

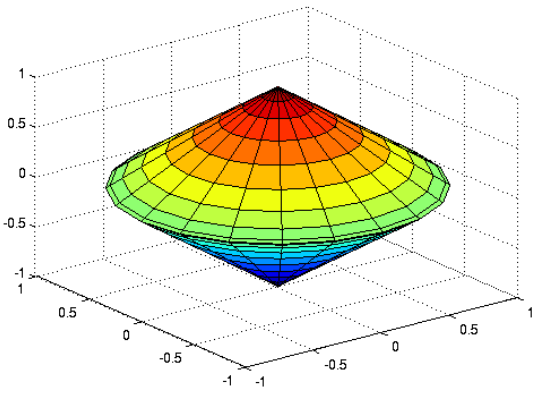

Figure 6.5: Superquadric that model the feasible region

for other values of excentricity we find similar figures, but we could not yet model the region. Figures 6.6, 6.7 and 6.8 show the region for an excentricity of 0.3.

6.2 Control Action in D Space

Now that the trajectories that satisfy the problem are well defined, it is possible to analyse how the algorithm works in this space. Since there is a saturation constraint, when applied a control the change in the trajectory is limited. In a situation where the spacecraft is really far from the feasible region it will not be possible to, with the application of only one control to come back to a feasible solution (return to a periodic orbit that respect

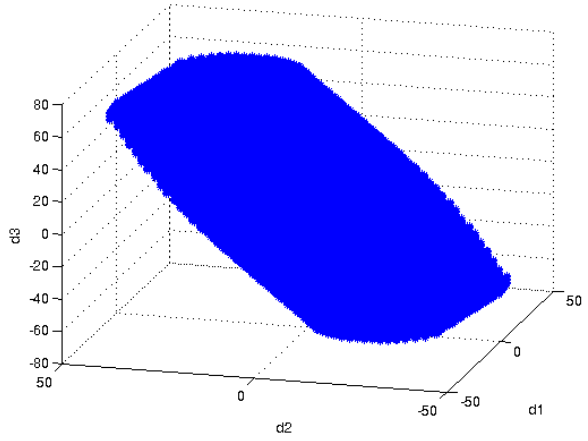

Figure 6.6: Feasible region for $e = 0.3$

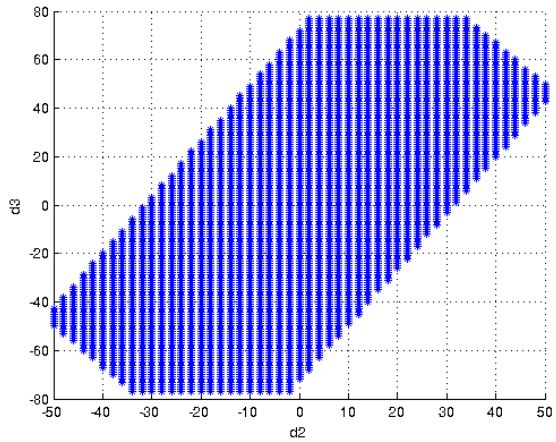

Figure 6.7: Lateral view of the feasible region for $e = 0.3$

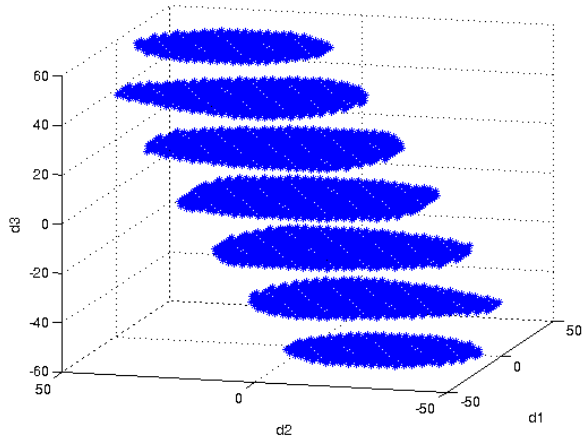

Figure 6.8: Cuts for the feasible region for $e = 0.3$

the polytopic constraints). The question is now how far can the spacecraft be from the feasible region and be able to come back with a single impulse.

It is possible to define a region that will be the region that can be reached in the D space if it is applied every possible control value. Using (6.3) and fixed maximum/minimum speed values we can trace the region in D it could reach. Like before, the movement is decoupled and it will be analysed only in terms of d_1, d_2 and d_3. This region is the plane shown in Figure 6.9.

$$C(\nu)=\begin{bmatrix} 0 & 0 & \frac{-(3\,e\cos\nu+e^2+2)}{e^2-1} & \frac{(1+e\cos\nu)^2}{e^2-1} & 0 & \frac{-e\sin\nu(1+e\cos\nu)}{e^2-1} \\ 0 & 0 & \frac{3(e+\cos\nu)}{e^2-1} & \frac{-(2\cos\nu+e\cos^2\nu+e)}{e^2-1} & 0 & \frac{\sin\nu(1+e\cos\nu)}{e^2-1} \\ 0 & 0 & \frac{3\sin\nu(1+e\cos\nu+e^2)}{(e^2-1)(1+e\cos\nu)} & \frac{-\sin\nu(2+e\cos\nu)}{e^2-1} & 0 & \frac{-(\cos\nu+e\cos^2\nu-2e)}{e^2-1} \\ 1 & 0 & \frac{-3\,e\sin\nu(2+e\cos\nu)}{(e^2-1)(1+e\cos\nu)} & \frac{e\sin\nu(2+e\cos\nu)}{e^2-1} & 0 & \frac{e^2\cos^2\nu+e\cos\nu-2}{e^2-1} \\ 0 & \cos\nu & 0 & 0 & -\sin\nu & 0 \\ 0 & \sin\nu & 0 & 0 & \cos\nu & 0 \end{bmatrix} \tag{6.3}$$

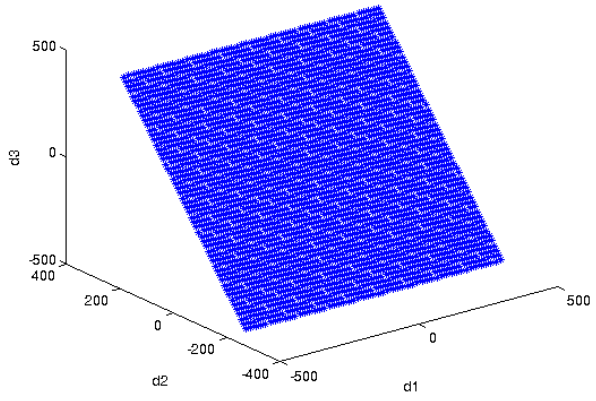

Figure 6.9: Attainable region from an arbitrary point using one control.

From 6.3 it is possible to see that the influence of the control into $[d_0, d_1, d_2, d_3]$ is a matrix with, at best, rank 3. If the periodicity constraint is applied and forces d_0 to 0, this rank is reduced to 2. So even if there are actuators in the 3 axes, only 2 directions can be, in fact, controled. It is important to note that matrix 6.3 depends on the anomaly, so

for each anomaly value the plan is different, in the sense that the direction of its normal vector changes.

So in the situation that we need to control a spacecraft in the rendezvous stage. If there is access to its state in D it is possible to know if it is inside the feasible region. If it is inside, there is no need to correct anything since the trajectory will be periodic and will rest inside the polytope. Now suppose that the state is not inside the feasible region, it is possible to trace the plan for the anomaly value and see if it intersects the feasible region. If there is an intersection so there is a control value that can be applied to bring our spacecraft to a good trajectory. Now take the region defined by all the points from which is possible to reach the feasible region for a determined anomaly. If a point is inside this region there is always a control that will bring it to a good trajectory, however, if it is outside it is clear that for that value of anomaly there is no solution to the problem.

The region described above can be defined as the minkowski sum of both the plan and the feasible region [13],[16],[12]. Figure 6.10 show this region for excentricity 0 and a value of anomaly (0) and figure 6.11 compares for two anomaly values (0 and $\pi/3$).

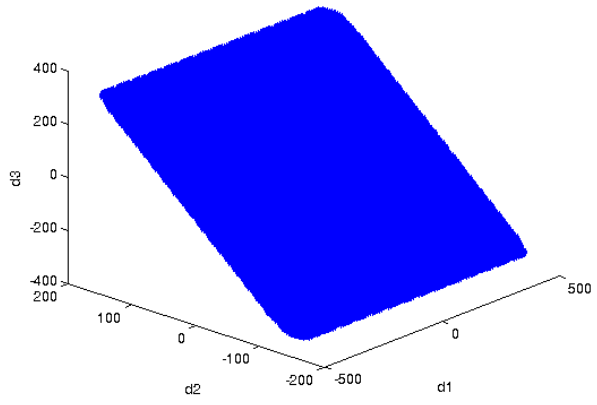

Figure 6.10: Region result of the Minkowski sum for anomaly 0

So the problem to know if there is a solution to the problem is equal to the problem to determine if the spacecraft is in this region or not, which can be easy if there is a analitical formula for it.

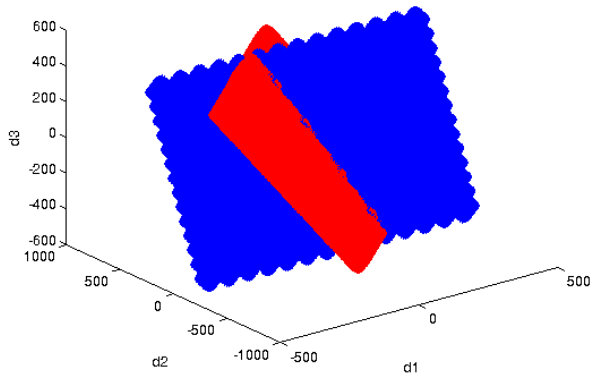

Figure 6.11: Region result of the Minkowski sum for two different anomalies (0 and $\pi/3$)

6.3 Conclusion

In this chapter was made a deeper analysis of the feasibility of the problem. Given the state of the spacecraft and the constraints definition it is possible to know if there exists a speed impulse that will bring the spacecraft to an orbit that respect the constraints. If there is no solution to the problem, another strategies can be used like the method using two impulses presented in [6]. Another solution if the problem is not feasible is to wait until it comes to feasibility, as explained before, all the constraints and the plane of action depend on the anomaly, so the problem can go from the feasibility to the unfeasibility only with the evolution of the anomaly.

Other applications of what was developed here is when it comes to the period between applications of control. If in some way the state of the spacecraft is monitored, it is possible to know when the problem will become unfeasible. As the dynamics of the vector D are well explained, the only thing that can be a problem are the perturbations. In a way it is possible to create a flag that decides when there is need to actuate in the spacecraft trajectory to avoid it going into the unfeasible region.

Chapter 7

Numerical Analysis

With the algorithm finished and its theoric aspects studied, a numerical analysis is important to know the behavior of the algorithm facing different situations. Many questions can be answered with this analysis, like: How many iterations are needed to converge using a warm start? And for the cold start? How is the constraints evolution during the iterative process? How can be set the breaking tolerances? The convergence prediction really works? Is there any problematic point for the algorithm?

In this chapter all this questions are answered and a logical explanation is given.

7.1 Example

This section describes an example of use of the algorithm in a complete simulation. Using the scenario presented in Table 7.1 we have simulated the system and acquired the data from the Algorithm of Alternating Projections.

7.1.1 Trajectory

Just analysing the trajectories we can see that both approaches shown in Chapter 5 solve the problem and maintain the satellite inside the especific box set as a constraint. Although both solutions are accepted we see that the control given as a result depends heavily on the initial condition as stated before. The warm start tends to maintain the satellite in the same trajectory given by the last step as we can see in Figure 5.3. On

Parameter	Value	Parameter	Value
a[km]	6763	Target Mass [kg]	462949
e	0.0052	Follower Mass [kg]	20000
i[°]	52	Drag Coeff Target	3
ω[°]	0	Drag Coeff Follower	2.274
Ω[°]	0	Target Surface [m^2]	1703
t_0	0	Follower Surface [m^2]	50
X_0[m,m/s]	[5;5;5;0;0;0]	max_it	100
$\Delta\ V_{max}$	0.26	Tol_vp	-0.001
σ [m/s]	6763	Tol_cons	0.001
Box	[-40,40,-40,40,-40,40]	Time Between Controls [s]	800

Table 7.1: Data for the scenario simulated

the other hand the cold start will begin always from the same initial point and will give diferent trajectories each step as showed in Figure 5.2.

7.1.2 Iterations

In the first step the warm approach initiates empty, what means that the initial guess is the identity matrix, so the convergence takes a greater number of iterations. After the first step though, the convergence is really fast and takes only a couple of iterations to find the feasible solution. For example, we can see this process as transient and steady regime, in a way that the number of iterations, similar to a state, will converge to a constant number (equilibrium). This process is shown in Figures 7.1 and 7.2. Steps 1 and 2 can be considered as the transitory regime for Figure 7.1 and steps from 1 to 6 are the transitory for Figure 7.2.

For the cold start we observe, in oposition to the warm, a constant number of iterations to converge. Since we start always from the same point, there is no division between temporary and steady regime. This results are presented in Figures 7.3 and 7.4. Both graphs are for the same simulation but Figure 7.4 represents a simulation of 50000 seconds instead of the 10000 seconds stabilished for this scenario.

Note that there are some points that did not converge, this issue will be treated later

in this chapter.

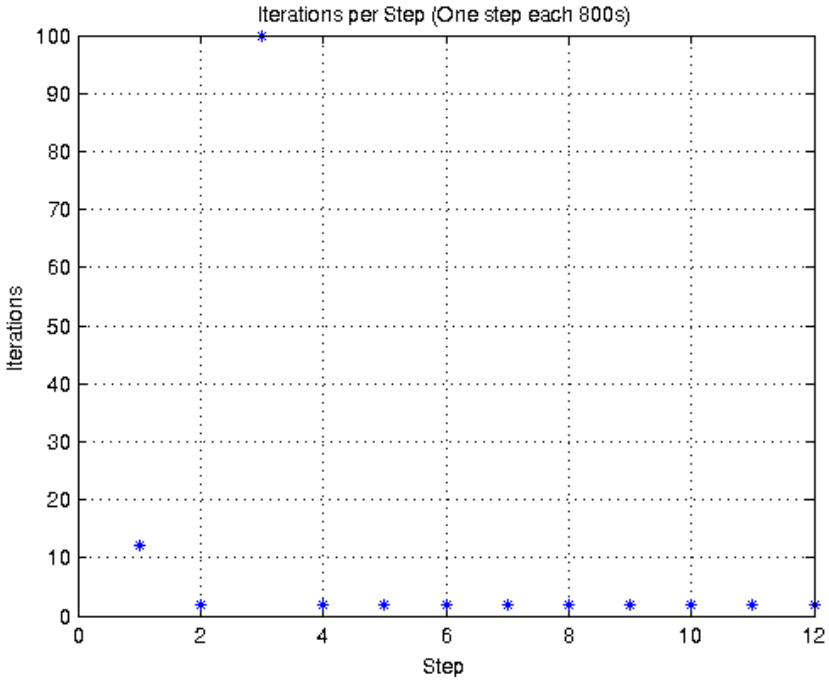

Figure 7.1: Iterations per Step Warm Start

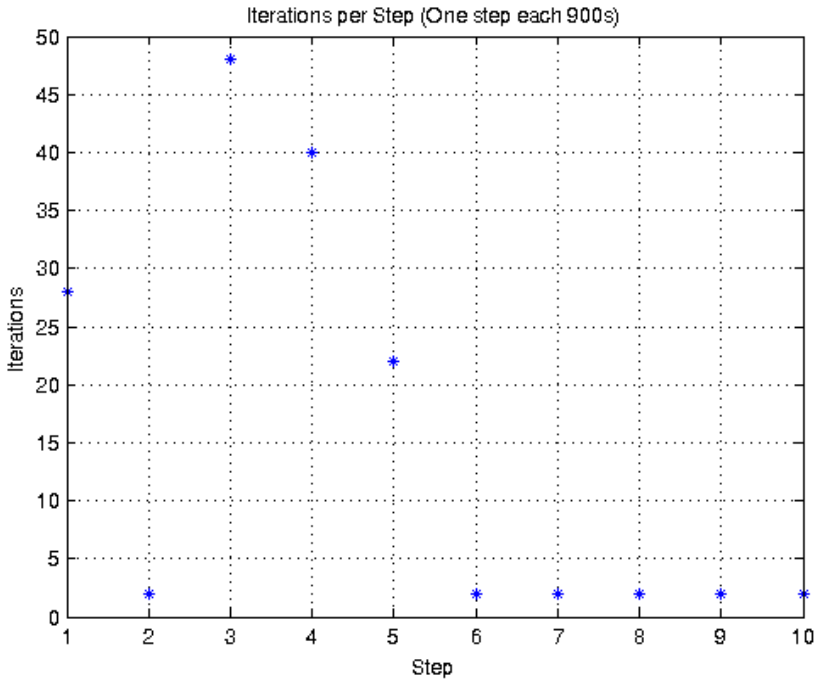

Figure 7.2: Iterations per Step Warm Start

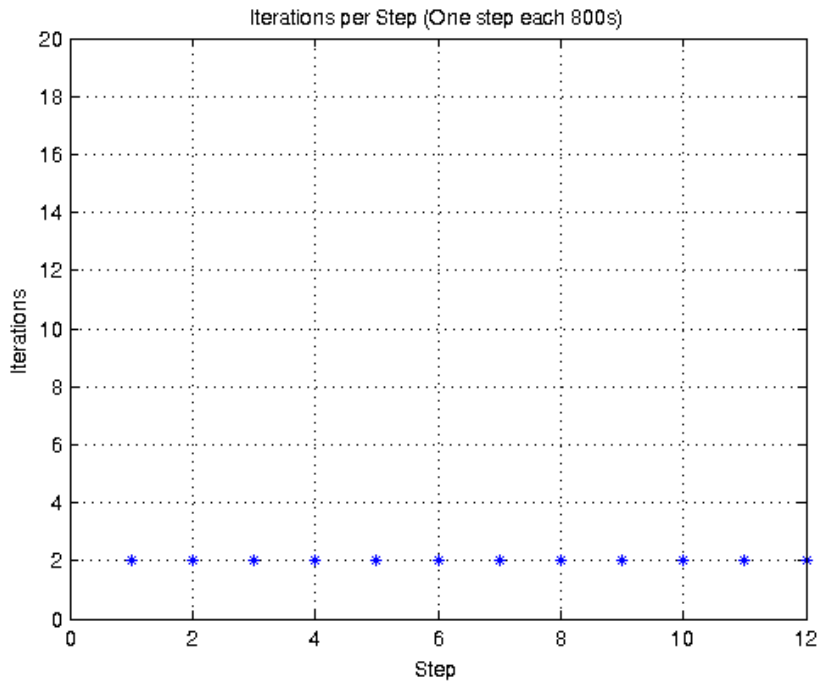

Figure 7.3: Iterations per Step Cold Start

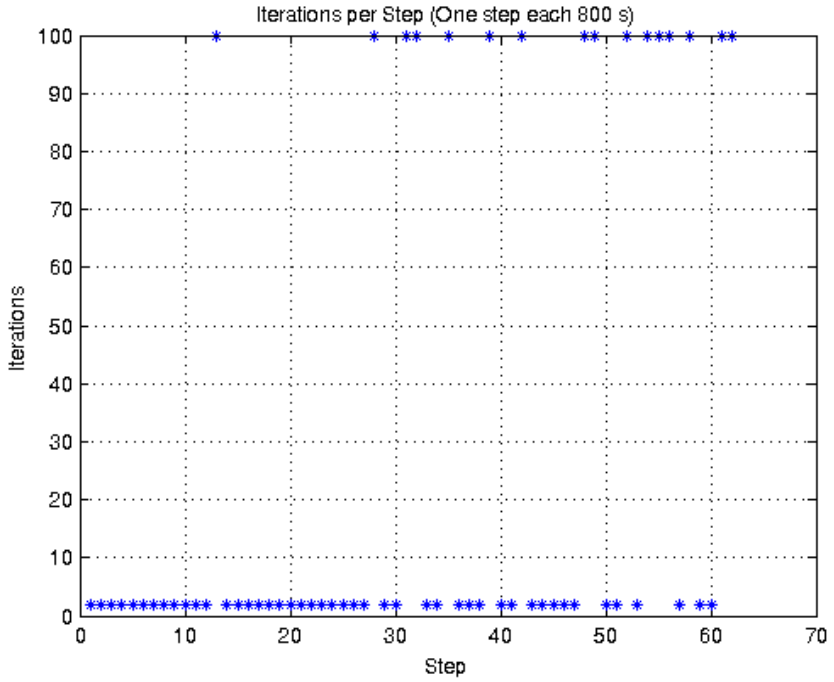

Figure 7.4: Iterations per Step Cold Start

7.1.3 Control Output

Analysing the control output, given by the warm start, along the time we see that it gives a first great effort to put the satellite on a trajectory and after that the control is just to maintain the trajectory. The satellite will diverge from the planned trajectory due to many factors: perturbations, model simplifications and errors; so the following controls will compensate these factors to maintain the trajectory stated in the first control calculation. This can be seen in the final trajectory made by the satellite. The control outputs given by ther warm start for the scenario described as example are in the Table 7.2.

Step	1	2	3	4	6	8	12
DV_x	0.0106	-0.0005	-0.0003	-0.0002	-0.0005	-0.0008	-0.0003
DV_y	0.0046	0	0	0	0	0	0
DV_z	-0.014	0.0005	0.0002	0.0001	0.0004	0.0005	0.0002

Table 7.2: Control Values for Warm Start

Similar to the warm, the cold start will require a greater effort in the first control application and then it will decreases. We can see that although the strategy is similar, the control efforts are bigger in the cold start what will reflect the greater consumption a priori and the different trajectory that it will trace. The control outputs given by the cold start for the scenario described as example are in the Table 7.3.

Step	1	2	3	4	6	8	12
DV_x	0.0106	-0.0005	-0.0003	-0.0002	-0.0005	-0.0008	-0.0002
DV_y	0.0046	0.0027	0.0017	0.0011	0.0005	0.0002	0
DV_z	-0.014	0.0042	0.001	-0.0022	-0.0033	-0.0008	0.002

Table 7.3: Control Values for Cold Start

7.1.4 Constraints

Warm

We can see that our initial solution is unfeasible and violates both constraints and as the projections are made the solution gets closer to the feasibility. In the second calculation however, as we use the last solution as our initial guess one of the constraints is already respected and the other is not far from being respected. The constraints evolution is showed in Tables 7.4 and 7.5.

min[eig(X)]	norm(v)
-0.85	262.2496
-0.425	0.85
-0.2125	0.425
-0.1062	0.2125
-0.0531	0.1062
-0.0266	0.0531
-0.0133	0.0266
-0.0066	0.0133
-0.0033	0.0066
-0.0017	0.0033
-0.0008	0.0017
-0.0004	0.0008

Table 7.4: Constraints Evolution for the Transitory Regime

min[eig(X)]	norm(v)
0	2.0221
0	0

Table 7.5: Constraints Evolution for the Steady Regime

Cold

For the cold start we have a different situation. As we start always with a matrice of zeros we are always far from one of the constraint (the equality) but we are near the

positive-definiteness. the constraint evolution is showed in Tables 7.6, 7.7 and 7.8.

min[eig(X)]	norm(v)
0.0468	220.7375
0.0468	0.0000

Table 7.6: Constraints Evolution Cold Start

min[eig(X)]	norm(v)
0.0467	217.7904
0.0467	0.0000

Table 7.7: Constraints Evolution Cold Start

min[eig(X)]	norm(v)
0.0471	216.0448
0.0471	0.0000

Table 7.8: Constraints Evolution Cold Start

7.2 Break Conditions

In order to study better the algorithm behaviour, we will now vary some parameters that affect the algorithm outcome. As explicited before, there are 3 conditions to stop the algorithm: Convergence in the sense of the tolerances, number of iterations is greater than its maximum or we predicted that the algorithm won't converge. These 3 cases have to be studied deeply because they are the only anomaly find in the algorithm performance.

7.2.1 Convergence

The convergence is determined when the minimum eigenvalue of the matrix X is greater than the tolerance tol_vp and the norm of the vector whic contains the residual from the equalities, $v_i = trace(A_i X) - b_i$, is inferior to the tolerance tol_cons. So if we change this tolerances to greater values we should have values that are less precise, on the other hand if we decrease the tolerances we will require a more precise solution from the algorithm.

First we will work only with the first tolerance and see its influence in the results, than the other and finally both. The simulation scenario will be the same from the beginning of this chapter and we will vary only the time of application. When needed other scenario will be explicited.

tol_cons

A priori tol_cons is set for 10^{-3} which is a great reference. We woul like to see what happens when we require more precision, so we will decrease the tolerance and compare the control output when solved the problem using the tolerance of 10^{-4} and 10^{-6}. Although the control will be calculated for 3 different tolerances we will apply only the one calculated with the reference value, so the trajectory and other results will be exactly the same as shown in the example. We increased the tolerance to see what happens when we relax the problem. We tried the values 10^{-2} and 1, and we expect to see less iterations than the normal but worst results. The difference in iterations is shown in Figure 7.5. In case of non convergence the last iteration is used as result.

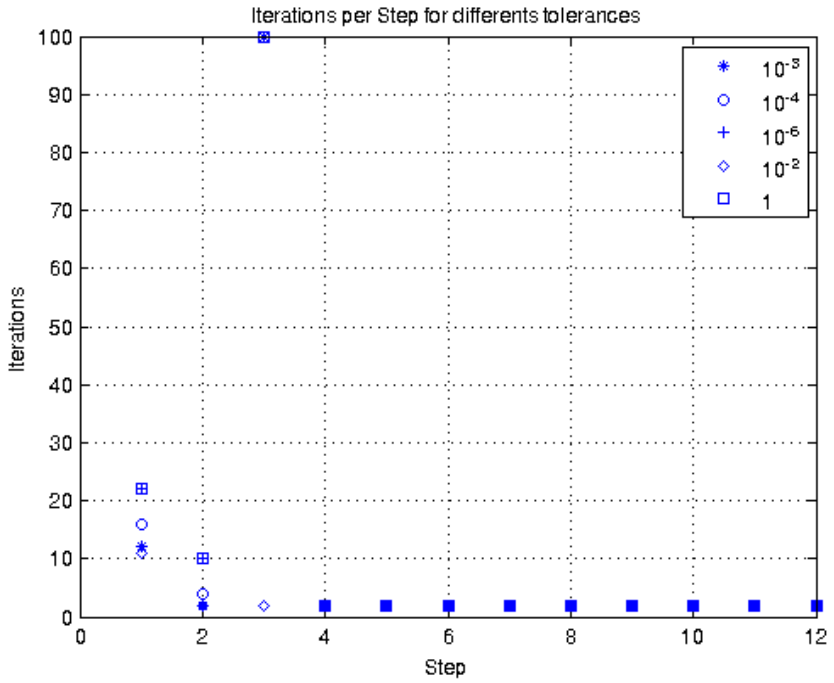

Figure 7.5: Iterations per Step when Varying Tolcons

We can see that if we decrease the tolerance we will have more iterations in the transitory regime but in the steady regime it remains the same number. Increasing the

tolerance can have good results, in the figure we can see that with 10^{-2} the algorithm converged in the third step. However if we increase too much we will get many iterations as well. Although the number of iterations have changed the control hasn't, the results are given in Table 7.9.

Comparation	Results
$10^{-3} - 10^{-4}$	-1.2387 10^{-17}
$10^{-3} - 10^{-6}$	-6.8236 10^{-18}
$10^{-3} - 10^{-2}$	4.0086 10^{-11}
$10^{-3} - 1$	3.9370 10^{-11}

Table 7.9: Difference in the Control Output when Varying Tolcons

The results are de sum of the difference in the control value, in fact, this difference is just in the transient regime and is caused by the difference in iterations. As the difference is virtually zero we conclude that changes in this tolerance won't affect the satellite's trajectory or consumption except in these cases of non-convergence that will be explained later. All these results are for the warm start, as cold start has no transitory regime in its evolution.

tol_vp

The same way as tol_cons, tol_vp is -10^{-3} as reference. To observe the algorithm's behaviour when we vary this tolerance we ran the simulation with the values -10^{-4}, -10^{-6}, -10^{-2} and -1 the same way it was done with tol_cons.

As we can see in Figure 7.6, when we change tol_vp the transitory regime evolve in the same way tol_cons did. However, as showed in the example, this constraint is easily satisfied in the begining, so if we change the tolerance to high values such as -1 it will continue to have many iterations caused by the other constraint. This show as well that we can relax one of the constraints if we maintain the other, the control results are virtually the same as showed in Table 7.10.

Both

Figure 7.7 and Table 7.11 show that when we change both tolerances we have a "combined" effect in terms of iterations and control output. Although we changed significantly the

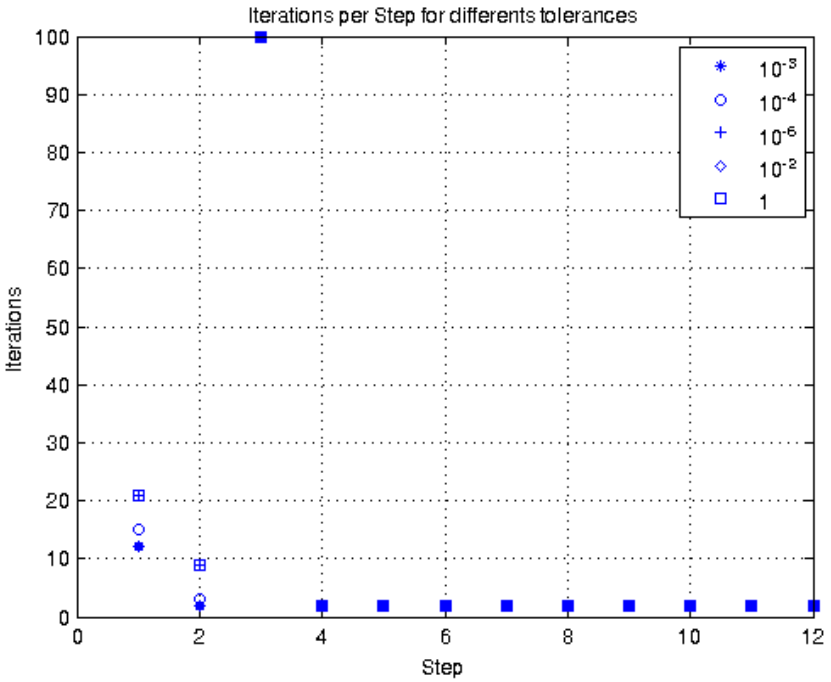

Figure 7.6: Iterations per Step when Varying Tolvp

Comparation	Results
$-10^{-3} - -10^{-4}$	-1.9441 10^{-17}
$-10^{-3} - -10^{-6}$	-3.0459 10^{-17}
$-10^{-3} - -10^{-2}$	0
$-10^{-3} - -1$	0

Table 7.10: Difference in the Control Output when Varying Tolvp

tolerances, the output is virtually the same (greater diference of 10^{-11}. So a priori we could solve a well relaxed problem that we would obtain the same performance.

Comparation	Results
$10^{-3} - 10^{-4}$	-1.3660 10^{-17}
$10^{-3} - 10^{-6}$	-6.1596 10^{-18}
$10^{-3} - 10^{-2}$	4.0086 10^{-11}
$10^{-3} - 1$	3.9870 10^{-11}

Table 7.11: Difference in the Control Output when Varying both Tolerances

Cases of Non Convergence

As observed in the last section we have some situations where the algorithm won't converge for some tolerances. To study this deeply we solved the same problem with the solver

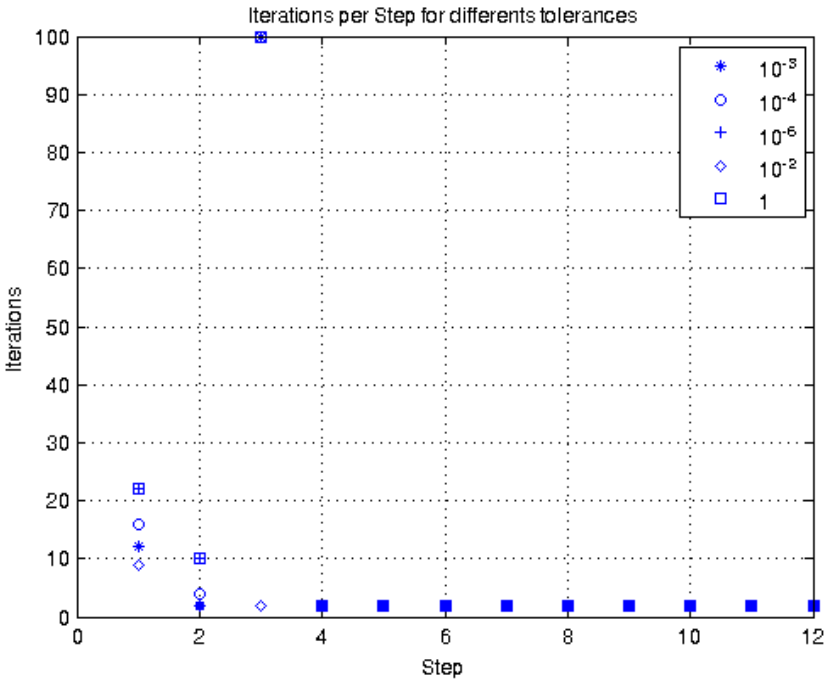

Figure 7.7: Iterations per Step when Varying both Tolerances

SeDuMi using the language *YALMIP* to see if the problem was of our algorithm or it was a case of infeasability. *YALMIP* showed that the problem was feasible, so we looked the constraints dynamics just for this case, the results are sohwed in Table 7.12.

The constraints show that eventually the algorithm will converge, but the rate $norm(v)$ is decreasing is too slow. If we use the tolerance of 10^{-2} it will converge fast as showed before, but this is due to the fact that the norm we are using here is not the best. When we use the Euclidean norm we will in fact sum all the 40 errors we have in each equality, so $norm(v)$ will be an information about the sum of errors and not about the greateast error itself. So here we change the euclidean norm for the infinity norm which would suit better. Doing again the simulation but now with the new norm and fixing both tolerances at 10^{-3} we obtain the results showed in Figure 7.8.

The results prove that changing the norm we correct the problem for this scenario, but maybe a better norm could be chosen. We can see in Figure 7.9 the example used for cold start with a simultaion time of 50000s that the new norm solves many of the non convergence cases but some are still not converging.

If we observe the constrains' dynamics we see the same evolution we had before. The norm will converge fast for a number greater than the tolerance, altough small, and will

Iteration	min[eig(X)]	norm(v)
1	-5.1879882812 10^{-5}	4.0409368482
2	-2.5939941406 10^{-5}	0.0015116952
3	-1.297 10^{-6}	0.0015110264
4	-6.485 10^{-6}	0.0015108585
10	-1.01327896 10^{-7}	0.001510797
11	-5.0663948 10^{-8}	0.001510796
12	-2.5331974 10^{-8}	0.0015107951
18	-3.95812 10^{-10}	0.0015107893
19	-1.97906 10^{-10}	0.0015107883
20	-9.8953 10^{-11}	0.0015107874
21	-4.9477 10^{-11}	0.0015107864
22	-2.4738 10^{-11}	0.0015107854
23	-1.2369 10^{-11}	0.0015107845
24	-6.185 10^{-12}	0.0015107835
35	-3 10^{-15}	0.0015107729
36	-2 10^{-15}	0.0015107719
37	-1 10^{-15}	0.001510771
38	0	0.00151077
39	0	0.001510769
40	0	0.0015107681
98	0	0.0015107121
99	0	0.0015107112
100	0	0.0015107102

Table 7.12: Constraints Evolution in a Non Convergence Case

start to converge very slowly to zero. This dynamic is presented in Table 7.13.

7.2.2 Prediction

The prediction done propose that with a linear regression of the evolution of the distance between the projections and the iterations we could foresee when the algorithm would converge. The idea is simple, if we know how many iterations it lasts to converge, we have a better control of the algorithm. To apply this method, we save the 10 last values of the distance which is calculated with the frobenius norm and take their logarithm. The theory proves that if the algorithm converge, its dynamic will be linear and if it doesn't converge this distance will go to the least distance bewtween the sets (will remain constant). So we set a tolerance for this distance and we calculate with the function gave by the linear regression how many iterations we would need to calculate to arrive to this

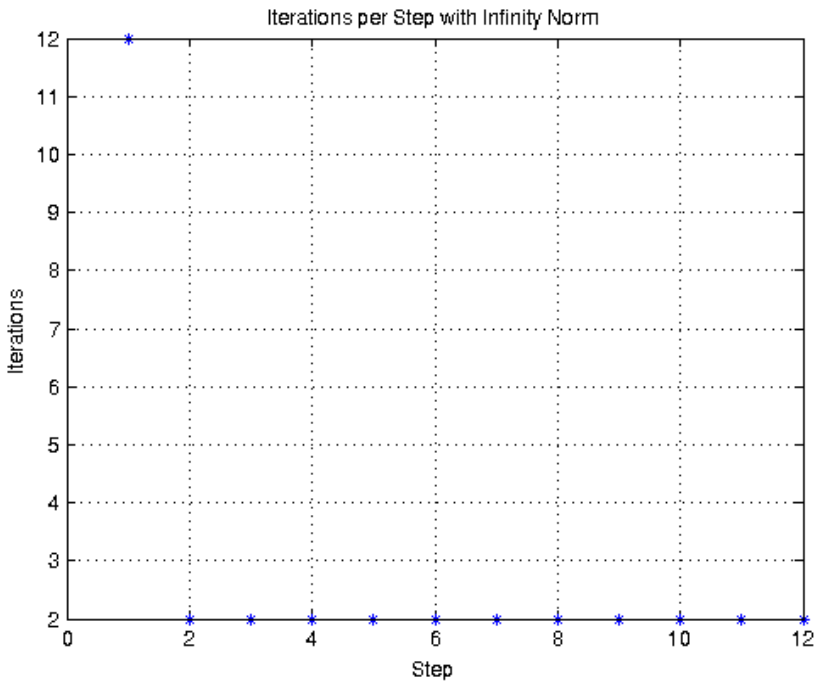

Figure 7.8: Iterations per Step for Infinity Norm

distance.

The problem here is that we do not use the distance between iterations as a break criteria, so we do not have a direct relation between a value for the distance when the algorithm stops. The prediction in fact is false as we wont do this number of iterations and we can't interpretate this that way.

We can, however, use this value to conclude about the convergence. If the distance is almost constant and we calculate a linear regression, the solution will be nearly parallel to a constant function, so the "prediction" will explose and tend to infinite. On the other hand if the distance has a real linear behaviour we will obtain a prediction value that will be proportional to our tolerance but will not diverge.

The decision we made at this point is to use not the prediction of how many iterations we need to certain distance, but rather use the number of iterations to gain one more decimal case of precision. This way, instead of have a fixed value we had to achieve, which has not necessarily correlation with the convergence, we have a different value each iteration, which is $log_{10}(norm(X - Y)) - 1$.

In Table 7.14 we can see the values predicted for 3 cases, the first is a transistory regime iterations, the second is one of the cases when the algorithm has a very slow rate

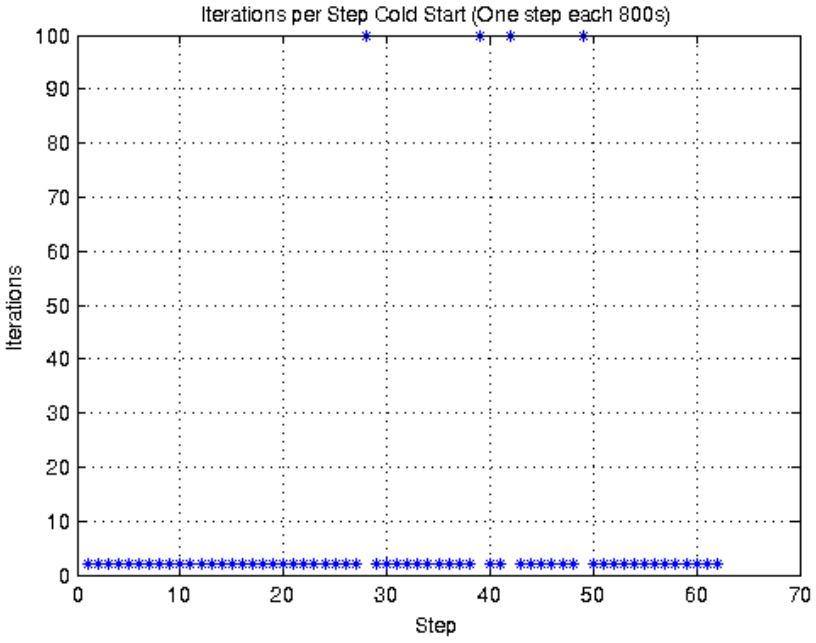

Figure 7.9: Iterations per step for Cold Start with Inf. Norm

of convergence and the last one is a case of non convergence.

As we see from the data, the three cases have really different values for the prediction and we will use this criteria to detect when the algorithm will converge and when it won't.

7.2.3 Maximum Iterations

Given the results presented in this chapter, the convergence if occurs is fast, so the number maximum of iterations must be a bound just for the transitory regime. Based on these same results a value of 100 is enough, but higher values could be used if the prediction works fine.

7.3 Conclusions

In this chapter a numerical analysis of the algorithm was made showing that:

- The algorithm converges really fast in general, but the number of iterations to converge depends heavily on the initial guess.
- For the warm start is possible to divide the mission in 2 stages:

Iteration	min[eig(X)]	norm(v)
1	0.04919442	212.14268699
2	0.04919442	$2.303833705\ 10^{-3}$
3	0.04919442	$2.3038321886\ 10^{-3}$
4	0.04919442	$2.3038306887\ 10^{-3}$
21	0.04919442	$2.3038051911\ 10^{-3}$
58	0.04919442	$2.303749696\ 10^{-3}$
59	0.04919442	$2.3037481962\ 10^{-3}$
60	0.04919442	$2.3037466964\ 10^{-3}$
98	0.04919442	$2.3036897021\ 10^{-3}$
99	0.04919442	$2.3036882023\ 10^{-3}$
100	0.04919442	$2.3036867025\ 10^{-3}$

Table 7.13: Constraints Evolution in a Non Convergence Case

1. A transient regime where the number of iterations oscillate.
2. A steady regime when the number of iterations is constant and low.

- The control output has a different evolution for different initialisation strategies.
- There is a phenomenum that, for a feasible point, the algorithm converges to a solution, however this convergence becomes very slow, causing a false "non-convergence".
- As the convergence is, in general, fast, when small changes are made to the break conditions (tolerances) there is no difference in the results.
- The prediction is a good way to separate the fast convergence, the slow convergence and the non convergence cases. However, it can not really predict when the algorithm will converge (stop).

Iteration	Case Non Convergence	Case Fast Convergence	Case Slow Convergence
11	1287396	21.7005	10.372074504
12	3171419	32.9043	14.772304552
14	13224645	34.9346	20.683916708
15	22029066	35.9475	27.128929675
16	33217492	36.9591	41.086584297
17	47039102	37.9696	79.535599668
18	64220403	38.9791	212.357647429
19	85818793	39.9876	727.332298743
30	1241754288	51.0404	3601993.73633929
31	1539938237	52.0428	3605405.19474435
32	1904277883	53.045	3606231.42814846
33	2348867834	54.047	3606404.45062156
34	2890821657	55.0488	3606425.9677981
35	3550887814	56.0504	3606396.86061344
36	4354222876	57.0518	3606406.52483441
37	5331410024	58.0531	3606422.13400877
42	14437248913	63.0579	3606556.74352201
43	17580305645	64.0586	3606544.26828223
44	21395465049	65.0592	3606502.97471186
45	26027468875	66.0598	3606429.45345462
46	31651073756	67.0603	3606407.8043123
47	38475101743	0	3606408.75590034
48	46754759729	0	3606401.77375098
49	56811402158	0	3606406.58288186
50	69008191130	0	3606406.31002688
51	83817723609	0	3606452.09854168
63	857532447644	0	3606464.77438039
64	1039522116077	0	3606460.26417575
65	1265167132675	0	3606502.24297534
66	1531641245609	0	3606511.61180991
67	1855432949852	0	3606577.66466431
68	2258080099026	0	3606560.86251601
69	2726459345405	0	3606523.30683028
70	3313754612067	0	3606515.27755723
95	320255973501994	0	3606544.85934599
96	1310138073416970	0	3606498.39468827
97	661078844384760	0	3606476.46378038
98	4503599627370610	0	3606445.53829439
99	1601279867509610	0	3606414.9316065
100	-800639933754664	0	3606408.03268244

Table 7.14: Prediction of Iterations Evolution

Conclusion

The purpose of this work was to analyse a model predictive control strategy for the problem of orbital rendezvous. The MPC was translated as a constraint satisfaction problem, formally described as to find a point in an intersection of two convex sets. The computational resources in satellites are not great so it was searched for a solution that could be executed with the computational constraints given by embedded systems. With this computational constraint we had to discard all the comercial optimization softwares, for example.

The algorithm chosen was the Alternating Projection and in the work of Arantes and Louembet [10],[9] all the development to write the algorithm in *Matlab* and the simulation environment in *Simulink* was done. This work was based on both the code and the simulator and it consists of a numerical analysis of the limitations in this solution.

Chapter 2 and 3 presented the mathematical development of the model used and all the process to write the problem into the constraints. Chapter 4 presented the algorithm and the simulator and its details. Chapter 7 describes all the numerical analysis of the code including variations in the entry parameters and changings in the break conditions.

Chapter 5 presented how sensitive the algorithm is in relation to its initial guess and explored the bounds of its solution. Comparing with an optimization method it was found out that the Alternating Projections Algorithm is capable of giving the optimal solution but for this it has to have a specific initial guess.

Chapter 6 presented a analysis of feasibility of the algorithm that can be used in future works including a robust analysis or the model of the system as a hybrid system.

In general this work shows that it is completely possible to use a low computational cost algorithm to control the time variant system, however it has many limitations. The main contribituion of this work is to explain and show the cons of using this kind of algorithm

when compared to highly precise optimization methods. Sometimes it is not possible to use complex optmization methods in embedded systems, when these cases arrive it is good to know the boundaries of the capacities for algorithms like the Alternating Projections.

Future works can be based on the physical implementation of the control law presented in this work. The physical implementation can be used to compare the numerical analysis and also analysis time constraints.Besides that, this work demonstrated that it is possible to achieve optimal consumptions using the Alternating Projections Algorithm, however it has not shown how to achieve it. From the theoric point of view there is a lot to develop in the direction of Hybrid Systems, using its tools to prove good characteristics of the control system proposed.

Appendix A

Properties of non negative polynomials

The results presented in Chapter 3 are based on the properties of non-negative polynomials given by Nesterov in [14]. Nesterov proves that the cone of coefficients of univariate polynomials which are non-negative on some segment of the real axis can be represented as the linear image of the cone of positive semi-definite matrices. This result enables the usage of the semi-definite programming for optimization problems with polynomial non-negativity constraints.

The definitions presented here are extracted from [14] and they concern only the concepts needed in order to understand the mathematical development in Chapter 3.

A.1 Checking polynomials non negativity on a finite interval

let $\kappa_{a,b}$ be the convex, closed and pointed cone of the coefficients of polynomials that are non negative on a finite interval $[a, b] \in \Re$:

$$\kappa_{a,b} = \left\{ p \in \Re^{n+1}, P(\omega) = \sum_{i=0}^{n} p_i \omega^i, \ \forall \omega \in [a, b] \right\} \tag{A.1}$$

Nesterov [14] shows that a polynomial $P(\omega)$, represented through its vector of coefficients $p = [p_0, ..., p_n]^T$, belongs to $\kappa_{a,b}$ if and only if there exist two symmetric positive semi-definite matrices Y_1 and Y_2 such that:

$$p \in \kappa_{a,b} \Leftrightarrow \exists Y_1, Y_2 \succeq 0 s.t. p = \Lambda^*(Y_1, Y_2) \tag{A.2}$$

the definition of the linear operator Λ^* and the dimensions of the matrices Y_1 and Y_2 depend on whether the polynomial $P(\omega)$ has an odd or even degree.

For n odd take Y_1,$Y_2 \in \Re^{(m+1)x(m+1)} \succeq 0$, where $m = (n-1)/2$. Let $H_{k,i} \in \Re^{(k+1)x(k+1)}$ be some Hankel matrices that contain ones on the i-th anti-diagonal and zeros everywhere else:

$$H_{k,1} = \begin{bmatrix} 1 & 0 & 0 & \cdots \\ 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & \cdots \\ \vdots & & & \ddots \end{bmatrix} H_{k,2} = \begin{bmatrix} 0 & 1 & 0 & \cdots \\ 1 & 0 & 0 & \cdots \\ 0 & 0 & 0 & \cdots \\ \vdots & & & \ddots \end{bmatrix} H_{k,3} = \begin{bmatrix} 0 & 0 & 1 & \cdots \\ 0 & 1 & 0 & \cdots \\ 1 & 0 & 0 & \cdots \\ \vdots & & & \ddots \end{bmatrix} \tag{A.3}$$

In this case, the operator Λ^* is defined as:

$$\Lambda^*(Y_1, Y_2) = \begin{bmatrix} tr(Y_1(-aH_{m,1})) + tr(Y_2(bH_{m,1})) \\ tr(Y_1(H_{m,1} - aH_{m,2})) + tr(Y_2(bH_{m,2} - H_{m,1})) \\ \vdots \\ tr(Y_1(H_{m,i-1} - aH_{m,i})) + tr(Y_2(bH_{m,i} - H_{m,i-1})) \\ \vdots \\ tr(Y_1 H_{m,2m+1}) + tr(Y_2(-H_{m-1,2m-1})) \end{bmatrix} \tag{A.4}$$

For n even take $Y_1 \in \Re^{(m+1)x(m+1)} \succeq 0$ and $Y_2 \in \Re^{mxm} \succeq 0$, where $m = \frac{n}{2}$. In this case, the operator Λ^* is defined by:

$$\Lambda^*(Y_1, Y_2) = \begin{bmatrix} tr(Y_1 H_{m,1}) + tr(Y_2(-abH_{m-1,1})) \\ tr(Y_1 H_{m,2}) + tr(Y_2((b+a)H_{m-1,1} - abH_{m-1,2})) \\ tr(Y_1 H_{m,3}) + tr(Y_2((b+a)H_{m-1,2} - H_{m-1,1} - abH_{m-1,3})) \\ \vdots \\ tr(Y_1 H_{m,i}) + tr(Y_2((b+a)H_{m-1,i-1} - H_{m-1,i-2} - abH_{m-1,i})) \\ \vdots \\ tr(Y_1 H_{m,2m}) + tr(Y_2((b+a)H_{m-1,2m-1} - H_{m-1,2m-2})) \\ tr(Y_1 H_{m,2m+1}) + tr(Y_2(-H_{m-1,2m-1})) \end{bmatrix} \tag{A.5}$$

A.2 Checking polynomials non negativity on an infinite interval

The necessary and sufficient conditions for non negativity of univariate polynomials on infinite intervals have also been given in [14]. A polynomiaml $P(\omega)$ is non negative on $\Re$ if and only if there exists a symmetric positive semi-definite matrix $Y \in \Re^{(m+1)x(m+1)}$ such that p, the vector of coefficients of $P(\omega)$, verifies:

$$p \in \kappa_\infty \Leftrightarrow \exists Y \succeq 0 \ s.t. \ p = \Lambda^*(Y) \tag{A.6}$$

where the linear operator Λ^* is defined by:

$$\Lambda^*(Y)(j) = tr(Y H_{m,j}), j = 1, .., 2m+1 \tag{A.7}$$

Appendix B

Dynamics of the vector of parameters

The variable change defined by:

$$D(\nu) = C(\nu)\tilde{X}(\nu) \tag{B.1}$$

represents a valid state transformation since the matrix $C(\nu)$ is always invertible on the domain on which the spacecraft closed trajectories are defined:

$$\det(C(\nu)) = \frac{1}{1-e^2} \neq 0, \ \forall\, 0 \leq e < 1 \tag{B.2}$$

The passage from the space of the D parameters back to the Cartesian relative state is given by the inverse matrix:

$$\tilde{X}(\nu) = C^{-1}(\nu)D(\nu) = F(\nu)D(\nu) \tag{B.3}$$

where $F(\nu) \in \mathbb{R}^{6\times 6}$ is defined as:

$$F(\nu) = \begin{bmatrix} 0 & \sin\nu(2+e\cos\nu) & -\cos\nu(2+e\cos\nu) & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cos\nu & \sin\nu \\ 2 & \cos\nu(1+e\cos\nu) & \sin\nu(1+e\cos\nu) & 0 & 0 & 0 \\ 3 & 2e\cos^2\nu + 2\cos\nu - e & 2\sin\nu(1+e\cos\nu) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\sin\nu & \cos\nu \\ -\dfrac{3e\sin\nu}{1+e\cos\nu} & -\sin\nu(1+2e\cos\nu) & 2e\cos^2\nu - e + \cos\nu & 0 & 0 & 0 \end{bmatrix} \tag{B.4}$$

The dynamics of the vector of parameters $D(\nu)$ can be deduced from the dynamics defining the spacecraft relative motion. When the relative state is represented using local Cartesian coordinates, the relative dynamics can be modelled by a linear periodic dynamic equation:

$$\tilde{X}'(\nu) = \tilde{A}(\nu)\tilde{X}(\nu) \tag{B.5}$$

where the matrix $\tilde{A}(\nu)$ is defined as in (2.10). After differentiating (B.1) with respect to the independent variable ν, we obtain:

$$D'(\nu) = C'(\nu)\tilde{X}(\nu) + C(\nu)\tilde{X}'(\nu) \tag{B.6}$$

Introducing (B.3) and (B.5) in the previous equations leads to:

$$D'(\nu) = A_D(\nu)D(\nu) \tag{B.7}$$

with the matrix $A_D(\nu)$ defined by:

$$A_D(\nu) = C'(\nu)C^{-1}(\nu) + C(\nu)\tilde{A}(\nu)C^{-1}(\nu) \tag{B.8}$$

The expression for the dynamic matrix $A_D(\nu)$ can be obtained through direct computation:

$$A_D(\nu) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -\dfrac{3e}{(1+e\cos\nu)^2} & 0 & 0 & 0 & 0 & 0 \\ \dfrac{3}{(1+e\cos\nu)^2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{B.9}$$

A state transition matrix can be easily computed for the dynamical system (B.7). Assuming that the spacecraft relative motion is propagated using the Yamanaka-Ankersen transition matrix as in (2.11) and using the transformation (2.20), we obtain:

$$D(\nu) = C(\nu)\Phi(\nu,\nu_0)C^{-1}(\nu_0)D(\nu_0) = \Phi_D(\nu,\nu_0)D(\nu_0) \tag{B.10}$$

where the state transition matrix $\Phi_D(\nu,\nu_0)$ is given by:

$$\Phi_D(\nu,\nu_0) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ -3eJ(\nu,\nu_0) & 0 & 1 & 0 & 0 & 0 \\ 3J(\nu,\nu_0) & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.11}$$

The term $J(\nu,\nu_0)$ is the same integral term defined in (3.13).

The dynamic matrix A_D and the transition matrix Φ_D highlight some interesting properties of the spacecraft relative motion. It can be seen that the parameters d_4 and d_5 that define the motion on the y axis are always constant in time, implying that the motion on the y axis is always bounded. This is consistent with the fact that the motion on the y axis is known to be periodic. The parameters d_0 and d_1 are also constant while the values of d_2 and d_3 change over time. The evolution of d_2 and d_3 is conditioned by the value of d_0. It can be seen that in the general case their modulus grows linearly with respect to time. The parameters remain constant only when $d_0 = 0$.

Appendix C

Nature of the roots of a quartic function

This appendix is based on [1]. Given the general quartic equation

$$ax^4 + bx^3 + cx^2 + dx + e = 0 \tag{C.1}$$

with real coefficients and $a \neq 0$, the nature of its roots is mainly determined by the sign of its discriminant

$$\begin{aligned}\Delta = {} & 256a^3e^3 - 192a^2bde^2 - 128a^2c^2e^2 + 144a^2cd^2e - 27a^2d^4 \\ & +144ab^2ce^2 - 6ab^2d^2e - 80abc^2de + 18abcd^3 + 16ac^4e \\ & -4ac^3d^2 - 27b^4e^2 + 18b^3cde - 4b^3d^3 - 4b^2c^3e + b^2c^2d^2\end{aligned} \tag{C.2}$$

This may be refined by considering the signs of four other polynomials:

$$P = 8ac - 3b^2 \tag{C.3}$$

such that $\frac{P}{8a^2}$ is the second degree coefficient of the associated depressed quartic

$$Q = b^3 + 8da^2 - 4abc \tag{C.4}$$

such that $\frac{Q}{8a^3}$ is the first degree coefficient of the associated depressed quartic

$$\Delta_0 = c^2 - 3bd + 12ae \tag{C.5}$$

which is 0 if the quartic has a triple root; and

$$\delta = 64a^3e - 16a^2c^2 + 16ab^2c - 16a^2bd - 3b^4 \tag{C.6}$$

which is 0 if the quartic has two double roots.

The possible cases for the nature of the roots are as follows [15]:

1. If $\Delta < 0$ then the equation has two real roots and two complex conjugate roots.
2. If $\Delta > 0$ then the equation's four roots are either all real or all complex.
   - If $P < 0$ and $\delta < 0$ then all four roots are real and distinct.
   - If $P > 0$ or $\delta > 0$ then there are two pairs of complex conjugate roots.
3. If $\Delta = 0$ then either the polynomial has a multiple root, or it is the square of a quadratic polynomial. Here are the different cases that can occur:
   - If $P < 0$ and $\delta < 0$ and $\Delta_0 \neq 0$, there is a real double root and two real simple roots.
   - If $\delta > 0$ or ($P > 0$ and ($\delta \neq 0$ or $Q \neq 0$)), there is a real double root and two complex conjugate roots.
   - If $\Delta_0 = 0$ and $\delta \neq 0$, there is a triple root and a simple root, all real.
   - If $\delta = 0$, then:
     - If $P < 0$, there are two real double roots.
     - If $P > 0$ and $Q = 0$, there are two complex conjugate double roots.
     - If $\Delta_0 = 0$, all four roots are equal to $-\frac{b}{4a}$

Bibliography

[1] Wikipedia quartic function. https://en.wikipedia.org/wiki/Quartic_function. Accessed: July 2015.

[2] A.H. Barr. Superquadrics and angle-preserving transformations. *IEEE CGA*, 1(1):11–23, 1981.

[3] R.H. Battin. *An Introduction in the Mathematics and Methods of Astrodynamics*. American Institute of Aeronautics and Astronautics, 1999.

[4] S. Boyd and J. Dattoro. *Alternating Projections*. Stanford University, 2003.

[5] W. Cheney and A. Goldstein. Proximity maps for convex sets. *Proceedings of the AMS*, 10:448–450, 1959.

[6] G. I. Deaconu. *On the trajectory design, guidance and control for spacecraft rendezvous and proximity operations*. PhD thesis, Université Paul Sabatier - Toulouse III, 2013.

[7] R.L. Dykstra. An algorithm for restricted least squares regression. *Journal of the American Statistical Association*, 78(384):837–842, 1983.

[8] C. Louembet G. Deaconu and A. Théron. Constrained periodic spacecraft relative motion using non negative polynomials. *American Control Conference (ACC)*, 2012.

[9] P.R. Arantes Gilz. Contrôle impulsionnel pour une classe de systèmes linéaires à temps variant. Technical report, LAAS - CNRS, 2014.

[10] P.R. Arantes Gilz and C. Louembet. Predictive control algorithm for spacecraft rendezvous hovering phases. In *European Control Conference*.

[11] M. Kara-Zaïtri. *Modélisation et guidage robuste et autonome pour le problème du rendez-vous orbital*. PhD thesis, Université Paul Sabatier - Toulouse III, 2010.

[12] H. Mann. *Addition Theorems: The Addition Theorems of Group Theory and Number Theory*. Robert E. Krieger Publishing Company, 1976.

[13] M. B. Nathanson. *Additive Number Theory: Inverse Problems and Geometry of Sumsets*. Springer, 1996.

[14] Y. Nesterov. *Squared Functional Systems and Optimization Problems*. Kluwer Academic Publishers, 2000.

[15] E. L. Rees. Graphical discussion of the roots of a quartic equation. *The American Mathematical Monthly*, 29(2):51–55, 1922.

[16] R. Tyrrell Rockafellar. *Convex analysis. Princeton landmarks in mathematics*. Princeton University Press, 1997.

[17] J. Tschauner. Elliptic orbit rendezvous. *AIAA Journal*, 5(6):1110–1113, 1967.

[18] K. Yamanaka and F. Ankersen. New state transition matrix for relative motion on an arbitrary elliptical orbit. *Journal of Guidance, Control and Dynamics*, 25(1):60–66, 2002.