Embedding a SDP-based control algorithm for the orbital rendezvous hovering phases
Frédéric Camps, Paulo Ricardo Arantes Gilz, Mioara Joldes, Christophe Louembet

To cite this version:
Frédéric Camps, Paulo Ricardo Arantes Gilz, Mioara Joldes, Christophe Louembet. Embedding a SDP-based control algorithm for the orbital rendezvous hovering phases. International Conference on Integrated Navigation Systems, Jun 2018, Saint Petersburg, Russia. <hal-01729956>

HAL Id: hal-01729956
https://hal.laas.fr/hal-01729956
Submitted on 13 Mar 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Embedding a SDP-based control algorithm for the orbital rendezvous hovering phases

Frédéric CAMPS  
LAAS-CNRS  
Université de Toulouse  
CNRS, Toulouse, France  
fcamps@laas.fr

Paulo Ricardo ARANTES GILZ  
LAAS-CNRS  
Université de Toulouse  
CNRS, Toulouse, France  
prarante@laas.fr

Mioara JOLDEŞ  
LAAS-CNRS  
Université de Toulouse  
CNRS, Toulouse, France  
mjoldes@laas.fr

Christophe LOUEMBET  
LAAS-CNRS  
Université de Toulouse  
CNRS, Toulouse, France  
louembet@laas.fr

Abstract

In this paper we present the procedures for embedding a semidefinite programming-based control algorithm for the impulsive rendezvous hovering phases problem on a board containing a synthesized LEON3 microprocessor. The performance of this algorithm is benchmarked by means of simulations against traditional linear programming-based constraint discretization approaches.

Keywords: FPGA embedded control law, semidefinite programming, optimization-based control, polynomial non-negativity, impulsive rendezvous, aerospace

1 Introduction

Spacecraft autonomy has become an important feature in the development of recent space missions, especially when the communication times make impossible any control from the ground. Embedded algorithms dedicated to autonomous decision making and maneuvering have already been used in many missions, even in deep space exploration or Mars missions (e.g., the Japanese Hayabusa asteroid touchdown mission [11], the Soyuz and ATV automated docking systems [7, 9], the Rosetta deep space exploration mission [18], etc). Indeed, mastering the implementation of these algorithms and assessing their efficiency are crucial for the accomplishment of such mission goals, but also open a venue of economical opportunities in Earth orbiting missions, such as orbit servicing and debris removal.

In this work, we discuss the embedding of a novel control algorithm dedicated to the hovering phases of the impulsive orbital rendezvous missions. The spacecraft rendezvous consists of a sequence of maneuvers performed by an active follower satellite with the goal of getting closer or even docking to a leader satellite. These approaching maneuvers are divided into several phases which are defined according to the inter-satellite distance, communication, visibility and other constraints. One of these phases is the so-called hovering phase, during which the follower spacecraft is required to remain in a delimited region of the space relatively to the leader, while the mission control awaits for other events to be accomplished (measurements, synchronization, go/no go decision, visibility, etc).

In the literature, fuel-optimal spacecraft flying-formation problems and model predictive controllers related to space applications are traditionally formulated as Linear Programs (LP) (see [10, 16, 15]) thanks to discretization procedures. In the works of Deaconu and Louembet (see [5, 4]), a different approach for modeling the space constraints of the rendezvous problems was proposed. It consists in characterizing the periodic relative trajectories that are enclosed by polytopes defined in the local frame of the leader satellite by polynomials non-negativity constraints and employing a result demonstrated by Nesterov [13] to convert these polynomial constraints into linear matrix inequalities.

Beside providing a finite and exact description of such trajectories, this method allows for formulating the fuel-optimal problems under trajectory constraints as a semidefinite program (SDP). Thus, model
predictive control strategies can be considered to address various proximity operations (approaching under cone of visibility constraints, collision avoidance, hovering phases, etc) without loss of information due to discretization. However, these SDP problems associated to this approach have not been extensively tested on low-performance computation environments, such as the devices dedicated to space applications.

The goal of this work is to detail how the above-mentioned approach can be implemented on a space dedicated board containing a synthesized LEON3 microprocessor to solve the fuel-optimal impulsive spacecraft rendezvous hovering phases problem, exhibiting the performances of the proposed algorithm/code for different scenarios and comparing it against the traditional LP-based approach.

This paper is organized as follows: in the first part, the models adopted for the relative motion between spacecraft and its constraints are presented and the mathematical formulation of the rendezvous hovering phases problem is provided. In the second part, the approach employed on the numerical resolution of the above-mentioned mathematical problem is exposed (code, libraries, compilation chains and hardware). Finally, the rendezvous scenarios chosen for the simulations and the obtained results are presented.

2 Theoretical aspects

In order to mathematically formulate the rendezvous hovering phases problem, the adopted models for the relative motion between spacecraft and its constraints are first presented. Then, the formulation of the constrained rendezvous problem as a SDP problem is briefly described. Since this work focuses on the implementation and practical aspects of the problem, some details are omitted on purpose. However, the reader is provided with the relevant and precise references whenever it is necessary.

2.1 Relative motion

Let \( S_l \) be a non-actuated leader spacecraft and \( S_f \), an actuated follower spacecraft (see Fig. 1). The relative motion between these spacecraft is modeled in the frames \( \{O, \hat{I}, \hat{J}, \hat{K}\} \) and \( \{S_l, \hat{x}, \hat{y}, \hat{z}\} \) (see [7, section 3.1] for details). Considering Keplerian mechanics, the orbital evolution of the leader is described by its orbital elements: its elliptic orbit shape is given by the semi-major axis \( a \) and the eccentricity \( 0 < e < 1 \) and its angular position is given by the true anomaly \( \nu \).

Let be \( X(t) = [x(t), y(t), z(t), \dot{x}(t), \dot{y}(t), \dot{z}(t)]^T \) the vector that, for a given instant \( t \), represents the relative position and velocity between spacecraft. Under the hypothesis \( \|OS_l\| \gg \|S_lS_f\| \), the relative unforced dynamics can be modeled by the linearized Tschauner-Hempel equations (more details in [19, 20]):

\[
\dot{X}(\nu) = A(\nu)X(\nu),
\]

Then, thanks to a change of the independent variable from time \( t \) to true anomaly \( \nu \) and \textit{ad hoc} simplification, a simplified version of the linearized dynamics (1) is obtained:

\[
\dot{X}'(\nu) = A(\nu)X(\nu),
\]
where \( \bar{X}(\nu) = [\bar{x}(\nu), \bar{y}(\nu), \bar{z}(\nu), \bar{x}'(\nu), \bar{y}'(\nu), \bar{z}'(\nu)]^T \) is the state obtained by performing the following variable change:

\[
\bar{X}(\nu) = T(\nu)X(t).
\]  

The matrices \( \bar{A}(\nu) \) and \( T(\nu) \) are given in [4, page 20].

### 2.2 Space constraints

We assume that the rendezvous hovering regions are described by a rectangular cuboid (see Fig. 2), although more general polytopes can be similarly employed. These space constraints can be represented by the following inequalities:

\[
\underline{x} \leq x(t) \leq \overline{x}, \quad \underline{y} \leq y(t) \leq \overline{y}, \quad \underline{z} \leq z(t) \leq \overline{z}, \quad \forall t > t_0.
\]  

In the sequel, we focus on the description of periodic relative constrained trajectories. The main reason for considering these particular orbits is that, once reached, a periodic relative trajectory is tracked by the chaser spacecraft, with no extra fuel-consumption (considering an ideal model). These orbits are particularly advised for maintaining the relative motion securely enclosed in a delimited region of the space.

In order to obtain a straightforward characterization of these particular periodic relative orbits enclosed in tolerance polytopes, another variable change is performed:

\[
D(\nu) = C(\nu)\bar{X}(\nu).
\]  

This new transformation consists in using the inverse of the matrix of fundamental solutions of the dynamical system (2) proposed by Yamanaka and Ankersen in [20] as a linear application \( C(\nu) \) that maps the state vector \( \bar{X}(\nu) \) to a new vector \( D(\nu) \). This new vector provides a parametric description of the shape and evolution of relative orbits, unlike the states \( X \) and \( \bar{X} \) that only give the instantaneous relative position and velocity between spacecraft. Moreover, this variable change allows us to express the periodicity property by simply imposing that the first entry of \( D(\nu) \) is zero: \( d_0(\nu) = 0 \) (further details in [5, section II.B]).

![Figure 2: Periodic relative trajectory included in a rectangular cuboid.](image)

The periodic \((d_0 = 0)\) relative trajectories respecting the space constraints are described by the vectors \( D \in \mathbb{R}^6 \) such that, for all \( \nu:\n
\[
\underline{x} \leq M_x(\nu)D \leq \overline{x}, \quad \underline{y} \leq M_y(\nu)D \leq \overline{y}, \quad \underline{z} \leq M_z(\nu)D \leq \overline{z},
\]  

where:

\[
M_x(\nu) = \begin{bmatrix}
0, & (2+\epsilon_\nu)c_\nu, & -\frac{(2+\epsilon_\nu)c_\nu}{1+\epsilon_\nu}, & 0, & 0, & 0
\end{bmatrix}, \\
M_y(\nu) = \begin{bmatrix}
0, & 0, & 0, & \frac{1}{1+\epsilon_\nu}, & \frac{1}{1+\epsilon_\nu}, & 0
\end{bmatrix}, \\
M_z(\nu) = \begin{bmatrix}
0, & c_\nu, & s_\nu, & 0, & 0, & 0
\end{bmatrix}
\]  

with \( c_\nu = \cos(\nu) \) and \( s_\nu = \sin(\nu) \) (see [1, section 2.3.4]).

The inequalities in (6) represent infinitely many constraints. In the literature, the traditional approach to address this constraints satisfaction problem is through discretization: the constraints are only accounted for at a finite number \( N_{disc} \) of instants \( \nu_1, \ldots, \nu_{N_{disc}} \) instead of infinitely many values of \( \nu \). This approach produces an outer-approximation of the set of relative trajectories that are enclosed in the hovering region, leading to systematical violations of the original space constraints. In fact, the solution of minimization of a convex function over polytopes described by finitely many affine inequalities generally lay on one of its vertices, which do not belong to the original admissible set (see Fig. 3).
Deaconu presents in [5] a finite mathematical description of the periodic relative trajectories included in such a polytope. This description is obtained by converting each one of the rational trigonometric inequality constraints (6) into a polynomial non-negative constraint by employing the tangent half-angle substitution $w = \tan(\nu/2)$:

$$
\Gamma^l_x(D, w) \geq 0, \quad \Gamma^l_y(D, w) \geq 0, \quad \Gamma^l_z(D, w) \geq 0, \quad \forall w \in \mathbb{R}.
$$

Then, by applying the results demonstrated by Nesterov in [13, Theorem 17.10], these polynomial non-negativity constraints are replaced by the research of semidefinite positive matrices whose entries are related to the coefficients $\gamma^k_j(D)$ of the respective polynomials $\Gamma^k_j(D, w)$. The set of periodic relative trajectories respecting the space constraints can then be defined as:

$$
S_p^D = \{ D \in \mathbb{R}^6 \mid d_0 = 0 \exists Y^k_j \succeq 0 \text{ s.t. } \gamma^k_j(D) = \Lambda^*(Y^k_j), \quad j \in \{x, y, z\}, \quad k \in \{l, u\} \}
$$

The reader should refer to [4, chapter 3 and appendix B] for further details.

2.3 Control action and saturation constraint

Due to the chemical nature of the propellers, the actions performed by the actuators are modeled as impulsive velocity corrections. Given a true anomaly value $\nu$, $\Delta V(\nu) = [\Delta V_x, \Delta V_y, \Delta V_z]^T \in \mathbb{R}^3$, denotes the instantaneous impulse control at $\nu$. For sake of brevity, the impulse control at $\nu_i$ is noted $\Delta V_i$ and $\Delta V$ is the vector of all impulses such that:

$$
\Delta V = [\Delta V_1^T, \ldots, \Delta V_N^T]^T \in \mathbb{R}^{3N}.
$$

Thus, the state right after an impulse, $X^+(\nu_i)$ can be computed by:

$$
X^+(\nu_i) = X(\nu_i) + B \Delta V_i, \quad B = [0_3 \mid 3_3]^T.
$$

Equation (11) can be rewritten in function of $D$ by performing the similar transformation $X \xrightarrow{T} \tilde{X} \xrightarrow{C} D$, which results in:

$$
D^+(\nu_i) = D(\nu_i) + B_D(\nu_i) \Delta V_i,
$$

where $B_D(\nu_i) = T(\nu_i)C(\nu_i)B$. The vector of parameters obtained after a sequence of impulses applied at times $\nu_1, \nu_2 \ldots \nu_{N-1}, \nu_N$ can be expressed as follows:

$$
D^+(\nu_N) = \Phi^{\nu_N}_{\nu_1} D(\nu_1) + \sum_{i=1}^{N} \Phi^{\nu_N}_{\nu_i} B_D(\nu_i) \Delta V_i,
$$
where $\Phi^{\nu}_{i0}$ is the state-transition matrix describing the propagation of the vector $D(\nu)$ (see [4, equation 2.16]).

Considering that the follower spacecraft has six thrusters, two-by-two symmetrically disposed along each axis of its body (see [17, section II] for details), the fuel-consumption can be defined as the sum of the intensity of the impulsive velocity corrections performed by each propeller:

$$J(\Delta V) = \sum_{i=1}^{N} \|\Delta V_i\|_1 = \sum_{i=1}^{N} |\Delta V_{i,x}| + |\Delta V_{i,y}| + |\Delta V_{i,z}|.$$  \hspace{1cm} (14)

Moreover, the actuators have physical limitations that do not allow for the application of impulses of magnitude greater that a certain threshold $\bar{\Delta} V$. This constraint can be mathematically formulated as:

$$|\Delta V_{i,j}| \leq \bar{\Delta} V, \; i \in \{1, \ldots, N\}, \; j \in \{x, y, z\}.$$  \hspace{1cm} (15)

2.4 Optimization problem formulations

The LP and SDP versions of fuel-optimal impulsive rendezvous hovering phases problem can then be formulated as follows:

**Problem 1 (LP).** Given $e$, $a$, $N$, $\nu_1, \ldots, \nu_N$ and $\bar{X}(\nu_1) \in \mathbb{R}^6$, find $\Delta V \in \mathbb{R}^{3N}$ such that:

$$\min_{\Delta V} J(\Delta V) \quad \text{s.t.} \quad \left\{ \begin{array}{l} D(\nu_1) = C(\nu_1)\bar{X}(\nu_1), \\ D^+(\nu_N) = \Phi^{\nu_n}_{1n} D(\nu_1) + \sum_{i=1}^{N} \Phi^{\nu_n}_{i0} B_D(\nu_i) \Delta V_i, \\ d_{i0}^{+}(\nu_N) = 0 \\ \|\Delta V_{i,j}\| \leq \bar{\Delta} V, \quad i \in \{1, \ldots, N\} \\ \|\Delta V_{i,j}\| \leq \bar{\Delta} V, \quad j \in \{x, y, z\} \\ \|\Delta V_{i,j}\| \leq \bar{\Delta} V, \quad i \in \{1, \ldots, N\} \\ \|\Delta V_{i,j}\| \leq \bar{\Delta} V, \quad j \in \{x, y, z\} \\ D^+(\nu_N) \in S^+_D \end{array} \right.$$  \hspace{1cm} (LP)

**Problem 2 (SDP).** Given $e$, $a$, $N$, $\nu_1, \ldots, \nu_N$ and $\bar{X}(\nu_1) \in \mathbb{R}^6$, find $\Delta V \in \mathbb{R}^{3N}$ such that:

$$\min_{\Delta V} J(\Delta V) \quad \text{s.t.} \quad \left\{ \begin{array}{l} D(\nu_1) = C(\nu_1)\bar{X}(\nu_1), \\ D^+(\nu_N) = \Phi^{\nu_n}_{1n} D(\nu_1) + \sum_{i=1}^{N} \Phi^{\nu_n}_{i0} B_D(\nu_i) \Delta V_i, \\ d_{i0}^{+}(\nu_N) = 0 \\ \|\Delta V_{i,j}\| \leq \bar{\Delta} V, \quad i \in \{1, \ldots, N\} \\ \|\Delta V_{i,j}\| \leq \bar{\Delta} V, \quad j \in \{x, y, z\} \\ D^+(\nu_N) \in S^+_D \end{array} \right.$$  \hspace{1cm} (SDP)

3 Practical aspects

Hereafter we outline the approach we employ to address the optimization problems previously formulated. We also present the software and hardware environment on which their performance is assessed. First of all, let us begin by demonstrating how to employ the CSDP library (see [2, 3]) to solve (SDP):

3.1 Solving the the SDP problem via the CSDP library

The CSDP library solves semidefinite programs of the form:

$$\max_X \quad \text{tr}(C X) \quad \text{subject to} \quad \text{tr}(A_i X) = \alpha_i, \; i \in \{1, \ldots, m\}$$  \hspace{1cm} (CSDP)

where $\alpha_i \in \mathbb{R}$ and all the matrices $C$, $A_i$, and $X$ are real and symmetric matrices (see [3] for details). To write (SDP) in the form of (CSDP), the decision variables must be identified:

$$Y^l_{z}, \; Y^u_{z}, \; Y^l_{z}, \; Y^u_{z} \in \mathbb{R}^{3 \times 3}, \; Y^l_{y}, \; Y^u_{y} \in \mathbb{R}^{2 \times 2}, \; \Delta V \in \mathbb{R}^{3N}.$$  \hspace{1cm} (16)
Since only equalities of the type $\text{tr}(A_n X) = \alpha_n$ and semi-definiteness constraints of the type $X \succeq 0$ can be used in (CSDP), the variables of (SDP) must be modified in order to obtain only symmetric semidefinite positive matrices (or positive scalars). The problem is then reformulated as follows:

1) First we split each of the saturation constraints into two inequalities:

$$|\Delta V_{i,j}| \leq \Delta V \Leftrightarrow -\Delta V_{i,j} + \Delta V \geq 0$$

$$\Delta V_{i,j} + \Delta V \geq 0$$

(17)

2) Then we introduce the variables $W_{i,j}^- = \max\{0, -\Delta V_{i,j}\}$, $W_{i,j}^+ = \max\{0, \Delta V_{i,j}\}$ (which will be used instead of $\Delta V_{i,j}$, that will be lately reconstructed via the relation $\Delta V_{i,j} = -W_{i,j}^- + W_{i,j}^+$) and the auxiliary variables $Z_{i,j}$ and $\bar{Z}_{i,j}$, obtaining:

$$J(\Delta V) = \sum_{i=1}^N |\Delta V_{i,x}| + |\Delta V_{i,y}| + |\Delta V_{i,z}|$$

$$-\Delta V_{i,j} + \Delta V \geq 0, \quad \Delta V_{i,j} + \Delta V \geq 0$$

(18)

$$J(Z) = \sum_{i=1}^N Z_{i,x} + Z_{i,y} + Z_{i,z}$$

$$W_{i,j}^- + W_{i,j}^+ = Z_{i,j}, \quad Z_{i,j} + \bar{Z}_{i,j} = \Delta V$$

with $W_{i,j}^-, W_{i,j}^+, Z_{i,j}, \bar{Z}_{i,j} \geq 0$. The variables $W_{i,j}^-$ and $W_{i,j}^+$ play the role of the norm of the negative and the positive part of $\Delta V_{i,j}$, respectively; $Z_{i,j}$ are slack variables that assume the value of $|\Delta V_{i,j}|$; $\bar{Z}_{i,j}$ are the complementary of $Z_{i,j}$ with respect to $\Delta V$ ($Z_{i,j} + \bar{Z}_{i,j} = \Delta V$).

3) Now that all the necessary variables have been introduced, $X$ can be structured as a symmetric semidefinite positive block-diagonal matrix:

$$X = \text{diag}(Y_{x}^T, Y_{y}^T, Y_{z}^T, Y_u^T, Y_v^T, Y_w^T, W^-, W^+, Z, \bar{Z}) \in \mathbb{R}^{(16+12N) \times (16+12N)},$$

where $W^- = \text{diag}(W_{1,x}^-, \ldots, W_{N,z}^-)$ and $W^+, Z, \bar{Z}$ are defined analogously. The fact that this matrix is semidefinite positive accounts for the inequality constraints previously presented.

4) The equality constraints can now be written in the form $\text{tr}(A_n X) = \alpha_n$. The total number of these constraints adds up to $27 + 6N$, as we show in the sequel:

4a) For each coefficient of the polynomials $\Gamma^k_j$ presented in (8) there is one equality constraint $\gamma_j^k = \Lambda^*(Y_j^k)$ involving a semidefinite positive matrix. The polynomials related to the $x$ and $z$ constraints are of degree 4, with 5 coefficients; the polynomials related to $y$ are of degree 2, with 3 coefficients; each axis has 2 inequality constraints, which results in a subtotal of $2(5 + 5 + 3) = 26$ equality constraints;

4b) The periodicity constraint is expressed via one single equality constraint $d_0 = 0$, which results in a subtotal of $26 + 1 = 27$ equality constraints;

4c) There are $3N$ equality constraints $W_{i,j}^+ + W_{i,j}^- = Z_{i,j}$, which results in a subtotal of $27 + 3N$ constraints;

4d) Finally, there are $3N$ equality constraints $Z_{i,j} + \bar{Z}_{i,j} = \Delta V$, which results in a total of $27 + 3N + 3N = 27 + 6N$ constraints.

5) Given that the criterion in (CSDP) is maximized and that we want to minimize $J(Z) = \sum_{i=1}^N Z_{i,x} + Z_{i,y} + Z_{i,z}$, the matrix $C$ will be filled with $-1$ in the entries that occupy the same positions as the variables $Z_{i,j}$ in $X$ and with zeros elsewhere.

3.2 The test environment

The tests are performed on an AEROFLEX Gaisler GR-XC6S board that contains a synthesized LEON3 microprocessor (see [14] for specifications) and supports a IEEE-754 compliant floating-point unit with single and double precision (32- and 64-bit floats), running a Linux 2.6 environment that simulates the performance of devices usually employed in space applications (see [6]). The compilation chain and the used libraries are detailed hereafter.

3.3 Cross-compilation chains and Libraries

1) Cross-compiler: since we aim to perform the tests in an environment as close as possible to those employed in space systems, we use an embedded synthesized LEON3 processor that requires a particular X86/SPARC cross-compiler. The first step to use this environment is to install the cross-compilers
proposed by AEROFLEX GAISLER, which are then used for the cross-compilation of computation libraries such as LAPACK. The cross-compiler\(^1\) must be installed on a Linux machine in the /opt directory. Here are the installation commands:

\[
\begin{align*}
\$ & \text{ cd } /\text{opt} \\
\$ & \text{ tar xvjf sparclinuxctmultilib0.0.7.tar.bz2} \\
\$ & \text{ export } \text{PATH}=/\text{opt}/\text{sparclinux4.4.2toolchains/multilib/bin:$PATH}
\end{align*}
\]

2) LAPACK and GLPK: LAPACK is a widely used Linear Algebra PACKage. The GLPK package is a linear programming kit intended for solving large-scale linear programming (LP), mixed integer programming (MIP), and other related problems (see [8]). Since LAPACK is written in Fortran, it requires a Fortran cross compiler\(^2\). The procedure of cross-compilation of LAPACK for SPARC/LEON3 is the following one:

2.1) Shell variables for LAPACK cross compilation:

\[
\begin{align*}
\$ & \text{ export CPPFLAGS=":-I/opt/sparc-linux-4.4.2-toolchains/multilib/sparc-leon-linux-gnu/sys-root/usr/include"} \\
\$ & \text{ export CFLAGS=":-g -O2"} \\
\$ & \text{ export LD="/opt/sparc-linux-4.4.2-toolchains/multilib/sparc-leon-linux-gnu/bin/ld"} \\
\$ & \text{ export LDFLAGS=":-L/opt/sparc-linux-4.4.2-toolchains/multilib/lib/gcc/sparc-leon-linux-gnu/4.4.2"} \\
\$ & \text{ export CC=sparclinuxgcc}
\end{align*}
\]

2.2) CFLAG integrates the\(^{-g}\) option for debugging reasons, then use only the \(^{-O2}\) option. In the glpk-4.60 directory is the configure file that generates the makefile:

\[
\begin{align*}
\$ & ./configure host = sparcsunsunos4.1.1 \\
\$ & \text{ make}
\end{align*}
\]

2.3) Once the previous steps are executed, it is possible to compile a test program with the LAPACK library:

\[
\begin{align*}
\$ & \text{ sparclinuxg++ Wall test.c o test L./.li}bs\text{ lglpk}
\end{align*}
\]

2.4) At this point the libglpk.a library is created in the directory glpk-4.60/src/.libs.

3) CSDP: a C Library for Semidefinite Programming, it also uses the LAPACK library\(^3\). The usual CSDP compilation chain must be substantially modified for cross-compilation. Each Makefile file must be modified to integrate the LAPACK library and the paths to the compilers. We use the Linux diff command to highlight new instructions in makefiles. A complete CSDP archive with all the changes for a cross-compilation is available at http://homepages.laas.fr/fcamps/Csdp/Csdp.tar. Each Make.diff file makes it possible to apply a patch to the Makefile of the original archive. CSDP source code is available with the Linux command:

\[
\begin{align*}
\text{svn co https://projects.coinor.org/svn/Csdp/trunk}
\end{align*}
\]

Once the makefiles have been modified, a simple run of the Make command is needed to generate the CSDP library.

4) Embedding the libraries: The computation libraries must be installed on the board (in the case of a non-static compilation) in the standard directory of the libraries. The variable LD_LIBRARY_PATH must be modified, for example:

\[
\begin{align*}
\$ & \text{ export LD_LIBRARY_PATH=}$LD\_LIBRARY\_PATH:/usr/lib
\end{align*}
\]

4 Simulations and Results

Hereafter we assess and compare the practical performances of both SDP and LP approaches on the resolution of the rendezvous hovering phases problem by carrying out simulations on the space dedicated

\(^1\)http://gaisler.com/anonftp/linux/linux-2.6/toolchains/sparc-linux-4.4.2/sparc-linux-ct-multilib-0.0.7.tar.bz2

\(^2\)LAPACK official site: http://www.netlib.org/lapack/, Archives: http://www.netlib.org/lapack/lapack-3.7.0.tgz

\(^3\)CSDP official site: https://projects.coin-or.org/Csdp/
test environment described in Sec. 3.2. We simulate four distinct ISS rendezvous missions in which the follower spacecraft is maneuvered from four different initial states \( X_{01} - X_{04} \) to a periodic orbit enclosed by the hovering zone described by the bounds given in the 'Space constraints' section of Table 1. We employ the actual orbital elements obtained from [12] for the vector time GMT 2018/064/12:00:00.000 (section 'Parameters' of Table 1).

<table>
<thead>
<tr>
<th>Table 1: Scenarios</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
</tr>
<tr>
<td>( a \ [\text{m}] )</td>
</tr>
<tr>
<td>6777280</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Initial states ([m,m/s])</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_{01}(\nu_1) = [400, 300, -40, 0, 0, 0] )</td>
</tr>
<tr>
<td>( X_{02}(\nu_1) = [-800, 600, 200, 0, 0, 0] )</td>
</tr>
<tr>
<td>( X_{03}(\nu_1) = [-1500, 1300, 150, 0, 0, 0] )</td>
</tr>
<tr>
<td>( X_{04}(\nu_1) = [5000, 1300, 500, 0, 0, 0] )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Space constraints ([m])</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x = 50, x = 150, y = -25, y = 25, z = -25, z = 25 )</td>
</tr>
</tbody>
</table>

\( \Delta \nu \) represents the true-anomaly interval between two impulsive velocity corrections.

The C codes and the data files for the resolution of each scenarios with both SDP and LP approaches are available at [http://homepages.laas.fr/fcamps/CSDP/test_files.zip](http://homepages.laas.fr/fcamps/CSDP/test_files.zip).

For the LP tests, the executable instruction took 435098 bytes, the initialized static variables occupied 324 bytes and the uninitialized data occupied 120 bytes. The required libraries were `libstdc++.so.6`, `libm.so.6`, `libgcc_s.so.1`, `libc.so.6` and `glpk`. For the scenario \( X_{01} \) with 120 discretization points, the maximum resident set size was 8048 kbytes and the percent of CPU used by the job was 95%.

For the SDP tests, the executable instruction took 117549 bytes, the initialized static variables occupied 256 bytes and the uninitialized data occupied 136 bytes. The required libraries were `libgfortran.so.3`, `libc.so.6`, `libm.so.6` and `sdp`. For the scenario \( X_{01} \), the maximum resident set size was 4496 kbytes and the percent of CPU used by the job was 92%.

In Table 2 we present the total computation time to solve the optimization problems (Time, in seconds), the fuel-consumption (Cons., in meters per second) and the maximal violation of the space constraints (Viol., in meters) for a single run of each simulated case.

Figure 4 depicts the obtained periodic relative trajectory after the application of the 5 impulsive velocity corrections computed by the SDP algorithm departing from the initial state \( X_{01} \).

**Figure 4:** Obtained relative trajectory for the initial state \( X_{01} \).

We observe that both methods result in an equivalent fuel-consumption up to the third decimal case.
Table 2: Results

<table>
<thead>
<tr>
<th>X_01</th>
<th>(LP)</th>
<th>(SDP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>N_disc</td>
<td>40 80 120 160 200</td>
<td>—</td>
</tr>
<tr>
<td>Time</td>
<td>1.156 3.714 7.309 12.228 18.843</td>
<td>5.043</td>
</tr>
<tr>
<td>Cons.</td>
<td>0.402 0.402 0.402 0.402 0.402</td>
<td>0.402</td>
</tr>
<tr>
<td>Viol.</td>
<td>0.152 0.037 0.016 0.009 0.006</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X_02</th>
<th>(LP)</th>
<th>(SDP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>N_disc</td>
<td>40 80 120 160 200</td>
<td>—</td>
</tr>
<tr>
<td>Cons.</td>
<td>1.103 1.103 1.103 1.103 1.103</td>
<td>1.103</td>
</tr>
<tr>
<td>Viol.</td>
<td>0.152 0.037 0.016 0.009 0.006</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X_03</th>
<th>(LP)</th>
<th>(SDP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>N_disc</td>
<td>40 80 120 160 200</td>
<td>—</td>
</tr>
<tr>
<td>Time</td>
<td>0.892 2.815 5.895 10.910 16.353</td>
<td>4.684</td>
</tr>
<tr>
<td>Cons.</td>
<td>1.781 1.781 1.781 1.781 1.781</td>
<td>1.781</td>
</tr>
<tr>
<td>Viol.</td>
<td>0.015 0.015 0.015 0.009 0.006</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X_04</th>
<th>(LP)</th>
<th>(SDP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>N_disc</td>
<td>40 80 120 160 200</td>
<td>—</td>
</tr>
<tr>
<td>Time</td>
<td>0.941 2.915 5.413 8.703 12.746</td>
<td>3.391</td>
</tr>
<tr>
<td>Cons.</td>
<td>4.204 4.204 4.204 4.204 4.204</td>
<td>4.204</td>
</tr>
<tr>
<td>Viol.</td>
<td>0.154 0.039 0.017 0.010 0.006</td>
<td>0</td>
</tr>
</tbody>
</table>

Concerning the computation timings. the time spent on the computation of the solution of the (SDP) problem is of the same order of magnitude of those of the resolution of the (LP) problem with 80≈120 discrete values of \(\nu\). However, while the (SDP) approach produces an exact solution that does not violate the space constraints, the (LP) approach always produces a residual violation of the space constraints, even when a high number of discrete values of \(\nu\) are employed (this is explained by Fig. 3 and the discussion presented in Section 2.2).

5 Conclusions

We presented the mathematical models employed on the formulation of a SDP-based approach to resolve the fuel-optimal spacecraft rendezvous hovering phases problem and the procedure to compute solution on a board dedicated to space applications via the CSDP library. The performance of the proposed approach was compared against the traditional LP discretization method. During numerical tests the SDP approach showed a computation time of the order of magnitude of those obtained for the LP method, with the advantage of not violating the space constraints. We can conclude from the results that, even though the SDP algorithm scales badly for big problems, for a small number of impulses the time taken to the computations to be performed is completely acceptable for practical space applications. An implementation of this algorithm on RTEMS could possibly show better performances (given that the board where the tests were performed was running a Linux 2.6 environment during the simulations).

6 Acknowledgments

This work was supported by the FastRelax project (ANR-14-CE25-0018-01).

References


