POLITECNICO DI TORINO
Repository ISTITUZIONALE

Suboptimal LQR-based spacecraft full motion control: Theory and experimentation

Original
Suboptimal LQR-based spacecraft full motion control: Theory and experimentation / Guarnaccia, Leone; Bevilacqua, Riccardo; Pastorelli, STEFANO PAOLO. - In: ACTA ASTRONAUTICA. - ISSN 0094-5765. - STAMPA. - 122:(2016), pp. 114-136. [10.1016/j.actaastro.2016.01.016]

Availability:
This version is available at: 11583/2656831 since: 2016-11-22T00:44:13Z

Publisher:
Elsevier Ltd

Published
DOI:10.1016/j.actaastro.2016.01.016

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

02 May 2024

Acta Astronautica 122 (2016) 114–136

Contents lists available at ScienceDirect

Acta Astronautica

journal homepage: www.elsevier.com/locate/actaastro

Suboptimal LQR-based spacecraft full motion control: Theory and experimentation

Leone Guarnaccia [b,1], Riccardo Bevilacqua [a,*], Stefano P. Pastorelli [b,2]

[a] Department of Mechanical and Aerospace Engineering, University of Florida 308 MAE-A building, P.O. box 116250, Gainesville, FL 32611-6250, United States
[b] Department of Mechanical and Aerospace Engineering, Politecnico di Torino, Corso Duca degli Abruzzi 24, Torino 10129, Italy

ARTICLE INFO

Article history:
Received 19 January 2015
Received in revised form
18 November 2015
Accepted 18 January 2016
Available online 2 February 2016

Keywords:
Spacecraft
Optimal control
Linear quadratic regulator
Six-degree-of-freedom
Real-time
Experimental test

ABSTRACT

This work introduces a real time suboptimal control algorithm for six-degree-of-freedom spacecraft maneuvering based on a State-Dependent-Algebraic-Riccati-Equation (SDARE) approach and real-time linearization of the equations of motion. The control strategy is sub-optimal since the gains of the linear quadratic regulator (LQR) are re-computed at each sample time. The cost function of the proposed controller has been compared with the one obtained via a general purpose optimal control software, showing, on average, an increase in control effort of approximately 15%, compensated by real-time implementability. Lastly, the paper presents experimental tests on a hardware-in-the-loop six-degree-of-freedom spacecraft simulator, designed for testing new guidance, navigation, and control algorithms for nano-satellites in a one-g laboratory environment. The tests show the real-time feasibility of the proposed approach.

© 2016 IAA. Published by Elsevier Ltd. on behalf of IAA. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

1. Introduction

As spacecraft technology has evolved, robust and efficient automated control has become an essential mission capability. Over the last fifty years, in fact, worldwide aerospace research environments have addressed their work to the optimization of guidance, navigation and control performances, also paying attention to the propellant consumption and time-to-launch costs.

It is clear, then, how high efficiency controls, ensuring both position accuracy and propellant optimization, have become a critical issue in the control systems scope and specifically in the aerospace sector.

The spacecraft six degrees of freedom optimal control problem has been widely analyzed in the literature, lately focusing on spacecraft relative motion, usually requiring numerical methods [1]. Spacecraft formation and the optimization of relative maneuvers are becoming increasingly important topics of investigation. This is due to the benefits

Abbreviation: ADAMUS, Advanced Autonomous Multiple Spacecraft; APF, artificial potential function; AS, attitude stage; BP, balancing platform; CPU, central processing unit; DARPA, Defense Advanced Research Projects Agency; DCM, direction cosine matrix; EKF, extended Kalman filter; GDC, guidance dynamics corporation; IBPS, intelligent battery and power system; IPOPT, interior point optimizer; KKT, Karush–Kuhn–Tucker; LGR, Legendre Gauss Radau; LKF, linear Kalman filter; LQE, linear quadratic estimator; LQR, linear quadratic regulator; LVLH, local vertical local horizontal; MIMO, multi-input multi-output; MP, moving platform; NLP, nonlinear programming; ONR, Office of Naval Research; PWM, pulse width modulator; RTAI, Real-Time Application Interface; S/C, spacecraft; SNOPT, Sparse Nonlinear OPTimizer; TS, translational stage

[*] Corresponding author. Tel.: +1 352 392 6230.
E-mail addresses: leone.guarnaccia@gmail.com (L. Guarnaccia), bevilr@ufl.edu (R. Bevilacqua), stefano.pastorelli@polito.it (S.P. Pastorelli).

[1] Present address: 48 Via Salvatore Quasimodo, Acicatena 95022, Italy.

[2] Politecnico di Torino, Department of Mechanical and Aerospace Engineering, Room 6939.

http://dx.doi.org/10.1016/j.actaastro.2016.01.016
0094-5765/© 2016 IAA. Published by Elsevier Ltd. on behalf of IAA. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

Nomenclature

$\mathbf{0}_{m\times n}$	zero matrix with dimensions m, n
$\boldsymbol{A}$	state matrix (Jacobian)
$\boldsymbol{A}_{rot}$	state matrix rotational contribute
$\boldsymbol{A}_{transl}$	state matrix translational contribute
$\boldsymbol{B}$	input matrix (Jacobian)
$\boldsymbol{B}_{transl}$	input matrix translational contribute
$\boldsymbol{B}_{rot}$	input matrix rotational contribute
$\boldsymbol{C}$	output matrix
$\boldsymbol{D}$	feedforward matrix
d	thrusters moment arm with respect to the center of rotation of the AS [d=0.32 m]
${}^{E}\boldsymbol{DCM}_{B}$	direction cosine matrix from the body (B) to the inertial reference frame (E) [23]
F	propellant cost (–)
$\mathcal{F}$	generalized force vector ($\boldsymbol{F},\boldsymbol{M}$)
F_t	nominal thrust (F_t=0.3 N)
$\boldsymbol{F}_{thrust}$	force generated by the onboard thrusters (N)
$\mathcal{F}_{LQR}$	optimal generalized force, output of the LQR Simulink block
G	universal gravitational constant ($\text{m}^3\ \text{kg}^{-1}\ \text{s}^{-2}$)
$\boldsymbol{H}$	thruster distribution matrix or mapping matrix
$\boldsymbol{H}_f$	force term of the thruster distribution matrix $\boldsymbol{H}$
$\boldsymbol{H}_m$	torque term of the thruster distribution matrix $\boldsymbol{H}$ (m)
$\boldsymbol{I}_{m\times n}$	identity matrix with dimensions m, n
$\boldsymbol{J}$	inertia matrix (kg m^2)
J	global maneuver cost (–)
$\boldsymbol{K}$	Kalman gain
K_{pos}, K_{rot}	additional dimensionless gains characterizing the adaptive tuning, applied to the state weighting matrix position and angular terms only
$\boldsymbol{M}$	angular momentum vector (N m)
$M_{\oplus}$	Earth mass (kg)
$\mu_{\oplus}$	Earth gravitational parameter $\mu=GM_{\oplus}$ ($\text{m}^3\ \text{s}^{-2}$)
m	spacecraft simulator mass (kg)
$n=\sqrt{\mu_{\oplus}/\boldsymbol{R}_0^3}$	generic orbital rate (rad/s)
$\boldsymbol{\omega}$	angular velocity vector (rad/s)
$\omega_x,\omega_y,\omega_z$	angular velocity vector components (rad/s)
$\boldsymbol{P}$	solution of the Riccati differential equation
P	position cost (–)
p	thrusters inclination with respect to the imaginary square circumscribed to the attitude stage basis ($p=\cos(45^{\circ})=\sin(45^{\circ})$)
$\boldsymbol{Q}$	state weighting matrix (mixed dimensions to generate dimensionless cost)
$\boldsymbol{Q}_{transl}$	state weighting matrix translational contribute
$\boldsymbol{Q}_{rot}$	state weighting matrix rotational contribute
$\boldsymbol{R}$	input weighting matrix (mixed dimensions to generate dimensionless cost)
$\boldsymbol{R}_0$	generic orbit radius (m)
${}^{\omega}\boldsymbol{kin}_{\dot{\theta}}$	kinematics matrix relating the time derivative of the Euler angles with the angular velocity [23]
ρ	additional gain influencing the input weighting matrix (–)
$\boldsymbol{\theta}$	Euler angles vector (rad)
$\theta_x,\theta_y,\theta_z$	Euler angles (rad)
$\boldsymbol{U}^{*}$	generic LQR optimal solution
$\boldsymbol{U}_{cont}$	normalized continuous thrust vector (N)
U_{cost}	net propellant expenditure (–)
$\boldsymbol{u}_{10}$	normalized binary thrust vector (–)
u_a	dimensionless parameter with magnitude correspondent to the nominal thrust
v_x,v_y,v_z	linear velocity vector components (m/s)
$\boldsymbol{X}$	generalized state vector ($\boldsymbol{x},\dot{\boldsymbol{x}},\boldsymbol{\theta},\boldsymbol{\omega}$)
$\boldsymbol{X}_{des}$	desired generalized state
$\boldsymbol{X}_{err}$	actual error vector
$\boldsymbol{x}$	position vector (m)
$\dot{\boldsymbol{x}}$	linear velocity vector (m/s)
x,y,z	position vector components (m)
$\boldsymbol{Y}$	output vector

in cost, responsiveness, and flexibility of a multi-spacecraft system versus the classical monolithic satellite.

Particularly new is the use of continuous on–off engines, appearing on small spacecraft. This control constraint adds new complexity to finding the optimal solution. In fact, most of the literature on spacecraft optimal control assumes that the thrust can be finely modulated, partially to mitigate the aforementioned problem. Unfortunately, this is not the case with real engines, which are usually limited to some sustained value for thrust. As a partial response to this problem, a new methodology has been presented in [2] with the aim to control spacecraft rendezvous maneuvers assuming multi-level continuous thrusters and impulsive thrusters on the same vehicle. Furthermore, very recently, the interests of the Department of Defense have been focusing on time/propellant optimal rendezvous and capture maneuvers of a non-cooperative target satellite [3–5]. This research has been pushing the envelope with regards to fast computation of practical optimal/sub-optimal trajectories.

The recent numerical approaches to the optimal control problem could be in short classified in two main categories:

1. the indirect methods which employ the calculus of variations to obtain the first-order optimality conditions [6], where the resulting boundary-value problem is sometimes impossible or time-consuming to solve;
2. direct methods which approximate the trajectory via parameterization, and transform the cost functional into a cost function [7–9].

These second methods appear to be a viable tool for real-time spacecraft optimal control. However, the major issues with direct methods relate to the difficulties of defining parameters to represent feasible trajectories. The chosen parameterization may lead, for example, to unsatisfactory final conditions, or, it may satisfy the final conditions but violate some of the constraints on the controls. Adding penalties to the cost function may help, but it provides no guarantee of improvement as in [5,9]. Also, the trajectory computation must be as fast as possible, since it is iterated by an outer optimizer looping on the parameters. The rendezvous problem to a tumbling object analyzed in [5] is a very good example, where a direct method may not converge to feasible solutions or may converge in an unacceptable amount of CPU time for on board implementation.

Addressing the linear quadratic control problem, LQR-based algorithms were widely used to control space systems, specifically satellites and spacecraft assemblies. Yang proved the effectiveness of a quaternion-based LQR method for the design of nonlinear spacecraft control systems in [10], demonstrating how the designed controller globally stabilized the nonlinear spacecraft system, whereas it locally optimized the spacecraft performance.

Beatty, in [11], successfully demonstrated the superiority of a linear quadratic regulator with respect to a proportional-derivative (PD) controller. His work consisted of a comparison of spacecraft attitude control methodologies that use reaction wheels for torque actuation and star trackers to infer spacecraft orientation and angular rate.

Another proof of LQR reliability and effectiveness was given by Walker and Spencer [12]: this work focused on a system for relative navigation and automated proximity operations for a microsatellite using continuous thrust propulsion and low-cost visible and infrared imagers. In this case the state error was employed, together with the thrust vector, to define the cost function to be minimized by the LQR optimal gains; moreover the parametric weighting matrices defined as functions of the actual distance to the goal and of the maximum available thrust, while Pulse-Width-Modulation was used to determine thrust control.

Bevilacqua et al. worked on multiple spacecraft control by developing an autonomous distributed control algorithm for close proximity operations of multiple spacecraft systems, including rendezvous and docking scenarios. In order to validate the proposed control approach, both theoretical simulations [13] and experimental tests [14–16] were carried out, choosing the LQR as the system controller.

The LQR control effort served as the attractive force toward goal positions, while APF repulsive functions provided collision avoidance for both fixed and moving obstacles. Previous experiments, by the above authors, assumed that each spacecraft is equipped with an attitude control system, thus focusing only on the translational control. Regarding the control strategy, which represents the main subject of the present paper, as in [12], the weighting matrices were defined introducing a parametric structure depending on the maximum allowable values of the states and control effort. Simulations proved the LQR/APF to be both effective and efficient in conducting simultaneous spacecraft missions and in disturbance rejection.

The previous results obtained in [13] were then implemented and verified in [14], where both theoretical developments and experimental validation of the hybrid LQR/APF were presented. In this case propellant consumption was sub-optimized in real-time through re-computation of the LQR at each sample time, while the APF performed collision avoidance and a high level decisional logic.

There is a gap that has not been addressed directly in the past literature: real-time techniques for optimal control of both attitude and position. This is partly due to the fact that attitude and position are commonly dealt with independently; satellites primarily control their attitude to communicate to ground stations, to point to targets, to collect measurements or absorb energy from the Sun, whereas translational control is activated infrequently as it is only required during specific maneuvers such as orbit insertion and station-keeping. Nevertheless, current spacecraft rendezvous and docking maneuvers require the cooperation of multiple spacecraft deliberately designed to work together; hence both attitude and position need to be controlled constantly and simultaneously to maintain the formation and avoid collisions.

The present paper, in keeping with this context, illustrates a control algorithm taking into account both position and attitude, thus allowing a global control of a full six-degree-of-freedom spacecraft with a unique controller, in view of an implementation on multiple spacecraft assembly for the execution of rendezvous and docking maneuvers.

The adoption of a single controller would simplify the system's architecture, reducing the computational burden, without sacrificing required performances. Specifically, the control strategy is based on a LQR, whose gains are re-computed at each time sample, hence generating a sub-optimized solution of the six-degree-of-freedom problem. A comparison with an optimal control problem solver, based on a Radau pseudospectral Gaussian quadrature method, constitutes the theoretical validation of the proposed algorithm, evaluating the real optimization level. Finally, the presented LQR-based control system was also experimentally verified through a series of tests carried out on the test bed at the Advanced Autonomous Multiple Spacecraft (ADAMUS) laboratory [17,18]. This work particularly showcased the feasibility of a real-time approach.

The main contribution of this research to the state-of-the-art on spacecraft optimal relative motion control consists in the application of a SDARE approach to obtain a real-time control algorithm for six-degree-of-freedom spacecraft. SDARE approaches have been successfully applied in the past to several engineering problems [19–22] but, to the authors' knowledge, the complete spacecraft control problem has not received the attention it deserves. The paper completely characterizes the methodology via a comparison of its performances versus those of a general purpose software for solving nonlinear optimal control problems: GPOPS-II. GPOPS-II is based on an adaptive Radau pseudospectral Gaussian quadrature method. Furthermore, this work presents experimental validation of the proposed optimizer, through hardware-in-the-loop experimentation on a full six-degree-of-freedom test bed, showing real-time feasibility.

This paper is organized as follows. Section 2 illustrates the equations of the six-degree-of-freedom spacecraft motion, the dynamics linearization and deals with the control strategy by means of a linear quadratic regulator. Section 3 introduces the comparison with the optimal control software GPOPS-II, paying specific attention to the transformation of the results in order to have a meaningful matching with those from the LQR controller, thus ensuring an effective and truthful comparison. Section 4 illustrates the robotic spacecraft simulator employed for the experimental verification. Section 5 presents the results of the experimental tests. Section 6 concludes the paper.

2. Theory

The complete dynamics of spacecraft relative motion, as it will be expanded in (2.1) and (2.2), consists of the Euler equations and the Clohessy–Wiltshire equations, describing the spacecraft attitude and translation respectively. When dealing with the spacecraft simulator, used for the experiments, the same equations have been used, simplifying the Clohessy–Wiltshire equations to a double integrator, thus obtaining a direct correspondence between an orbiting spacecraft's dynamics and the robot dynamics itself. The resulting dynamics is represented by linear equations for the translation and nonlinear Euler's equations for the rotation. In the next subsections we will simplify the dynamics to represent the spacecraft simulator and we will use the terms spacecraft (S/C) simulator or robot interchangeably.

Turning to detail, the position and attitude motion is represented by the following twelve component vector:

- three relative position coordinates: x, y, z;
- the relative three linear velocity components: v_x, v_y, v_z or equivalently $\dot{x}$, $\dot{y}$, $\dot{z}$;
- a set of three Euler angles: θ_x, θ_y, θ_z;
- three angular velocity components associated with the rotation rate about the three axes constituting the body reference frame: ω_x, ω_y, ω_z.

Fig. 1 summarizes the conventional reference frames (body B, Local Vertical/Local Horizontal LVLH, inertial E) commonly employed to describe the relative motion of a spacecraft with respect to a generic target and expresses the sets of translational and rotational coordinates mentioned above.

The most common solution employed to control spacecraft entails a set of body-mounted thrusters providing unidirectional forces. This requires thruster mapping strategies, converting the tri-axial forces and torques, obtained from the equations of motion, into a series of unidirectional forces correspondent to each employed thruster.

2.1. Translational dynamics

The translational equations of motion of the spacecraft simulator have been derived from the more general Clohessy–Wiltshire equations describing the relative motion of a chase vehicle with respect to a target vehicle lying on a circular orbit of radius $\boldsymbol{R}_0$ and orbital rate $n = \sqrt{\mu_\oplus / \boldsymbol{R}_0^3}$. Starting from the Newton's second law,

$$m\ddot{\boldsymbol{R}} = \boldsymbol{F} = m\boldsymbol{g} + \boldsymbol{F}_{thrust} \tag{1}$$

where m is the mass of the vehicle, that is, in the present work, the spacecraft simulator, $\boldsymbol{g}$ is the gravitational acceleration and $\boldsymbol{F}_{thrust}$ is the control force, [23] illustrates how to derive the linear equations of relative motion, obtaining the well known expressions:

$$\begin{aligned} &\ddot{x} - 2n\dot{y} - 3n^2x = \frac{\boldsymbol{F}_{thrust\,x}}{m} \\ &\ddot{y} + 2n\dot{x} = \frac{\boldsymbol{F}_{thrust\,y}}{m} \\ &\ddot{z} + n^2 z = \frac{\boldsymbol{F}_{thrust\,z}}{m} \end{aligned} \tag{2}$$

The spacecraft simulator is not orbiting, thus implying the coincidence of the local reference frame with the inertial one in the laboratory: hence the terms related to the angular rate n, clearly null, vanish, finally obtaining:

$$\ddot{x} = \frac{\boldsymbol{F}_{thrust\,x}}{m} \tag{3}$$

$$\ddot{y} = \frac{\boldsymbol{F}_{thrust\,y}}{m} \tag{4}$$

$$\ddot{z} = \frac{\boldsymbol{F}_{thrust\,z}}{m} \tag{5}$$

In the sequel, the forces will be assumed to be expressed in the coordinate system fixed to the spacecraft, requiring a rotation matrix to obtain then in the inertial reference frame. This choice is convenient to simplify the procedure of thruster mapping.

2.2. Attitude dynamics

When the reference frame is the laboratory's floor, the spacecraft simulator attitude dynamics can be modeled employing the Euler's rotational equation of absolute motion in vector/dyadic form,

$$\boldsymbol{M} = \boldsymbol{J} \cdot \dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \boldsymbol{J} \cdot \boldsymbol{\omega} \tag{6}$$

Since the principal axes are also a central triad for the robot, that is the reference frame is baricentric and the inertial tensor is diagonal, expanding along the three axial directions all the products of inertia cancel out and Eq. (6) becomes

$$\begin{cases} J_x\dot{\omega}_x + (J_z - J_y)\omega_z\omega_y = M_x \\ J_y\dot{\omega}_y + (J_x - J_z)\omega_x\omega_z = M_y \\ J_z\dot{\omega}_z + (J_y - J_x)\omega_y\omega_x = M_z \end{cases} \tag{7}$$

The kinematics equation, using the Euler angles parametrization with chosen rotation sequence yxz, is [23]

$${}^E\boldsymbol{\omega}_{Byxz} = \dot{\theta}_y\boldsymbol{e}_2^{*} + \dot{\theta}_x\boldsymbol{e}_1^{**} + \dot{\theta}_z\boldsymbol{e}_3^{***}, \quad \boldsymbol{e}_3^{***} = \boldsymbol{b}_3 \tag{8}$$

where $\boldsymbol{e}_i^j$ are the basis vectors of different reference frames ($\boldsymbol{b}$ body frame) the subscript of which defines the rotation axis, whereas the superscript refers to the partial configuration obtained by each rotation. Expanding along the three axial components x, y, z, and then introducing the matrix notation, the previous equation assumes the form

$$\boldsymbol{\omega} = {}^{\omega}\boldsymbol{kin}_{\dot{\theta}}\dot{\boldsymbol{\theta}} \tag{9}$$

with

$${}^{\omega}\boldsymbol{kin}_{\dot{\theta}\,(Yxz)} = \begin{bmatrix} cz & cx * sz & 0 \\ -sz & cx * cz & 0 \\ 0 & -sx & 1 \end{bmatrix} \tag{10}$$

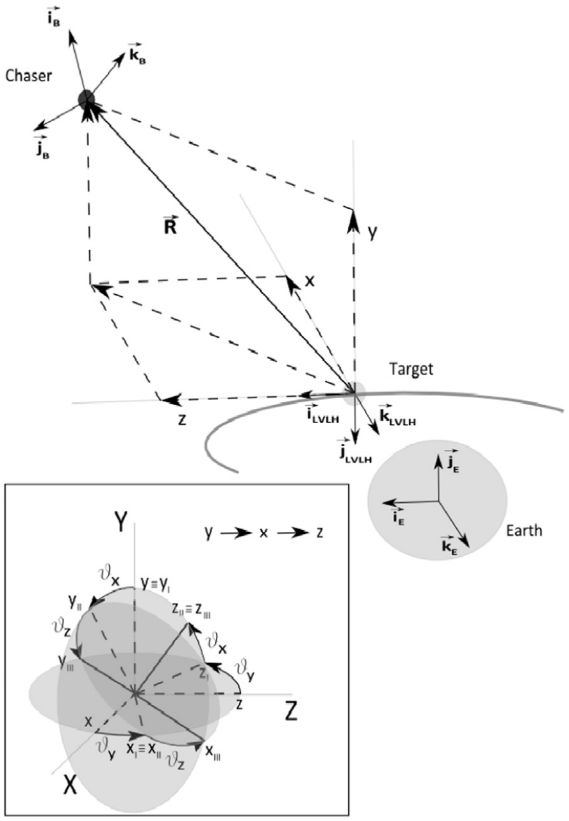


Fig. 1. Vectorial representation of the chaser position with respect to the target, including the inertial frame E, the local vertical local horizontal frame LVLH and the body frame B; attitude representation through the Euler angles parametrization, according to the sequence *yxz* (box).

where *ci* and *si* represent the shorten form of $\cos(\theta_i)$ and $\sin(\theta_i)$ respectively.

2.3. The linearized dynamics

A linearization process has been applied to the non-linear rotational dynamics of the robot to obtain the matrices representing the required inputs of the LQR to determine the optimal Kalman gain. The following derivations address decoupled translational and rotational motions, and the LQR solver generates independently the sub-optimal force and torque 3-component vectors to be mapped to the body mounted thrusters. Despite the ability of the LQR solver to generate directly the 12-component vector of the forces to be generated by the thrusters, thus addressing the coupled rototranslational dynamics, the authors chose to work with decoupled dynamics to retain the ability of a comparison with GPOPS-II. In fact, GPOPS-II has proven to have several convergence difficulties with the coupled dynamics, and to be very sensitive to the desired initial and final conditions. Given the twelve component state vector $\boldsymbol{X}$ expressed by

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x} \\ \dot{\boldsymbol{x}} \\ \boldsymbol{\theta} \\ \boldsymbol{\omega} \end{bmatrix} \tag{11}$$

where $\boldsymbol{x} = \{x, y, z\}^T$, $\dot{\boldsymbol{x}} = \{\dot{x}, \dot{y}, \dot{z}\}^T$, $\boldsymbol{\theta} = \{\theta_x, \theta_y, \theta_z\}^T$ and $\boldsymbol{\omega} = \{\omega_x, \omega_y, \omega_z\}^T$, the decoupled rotational and translational equation of motion can be summarized with the following vectorial expression:

$$\begin{aligned} \ddot{\boldsymbol{x}} &= \boldsymbol{F}/m \\ \dot{\boldsymbol{\theta}} &= {}^{\theta}\boldsymbol{kin}_{\omega}\boldsymbol{\omega} = {}^{\omega}\boldsymbol{kin}_{\dot{\theta}}^{-1}\boldsymbol{\omega} \\ \dot{\boldsymbol{\omega}} &= \boldsymbol{J}^{-1}(-\boldsymbol{\omega} \times \boldsymbol{J} \cdot \boldsymbol{\omega} + \boldsymbol{M}) \end{aligned} \tag{12}$$

or equivalently

$$\dot{\boldsymbol{X}}(t) = \mathcal{G}(\boldsymbol{X}(t)) + \boldsymbol{B}\mathcal{F}(t) \tag{13}$$

where $\boldsymbol{B}$ is a constant matrix. This represents a multi-input multi-output (MIMO) nonlinear system, a function both of the 12-dimensional state vector $\boldsymbol{X}$ and the 6-dimensional input $\mathcal{F}$, including forces $\boldsymbol{F}$ and torques $\boldsymbol{M}$. The system is then linearized at the desired state, at every time step. Proceeding in this way for every time step, the whole trajectory will finally consist of a sequence of linearized sections, and will approach the non-linear one as the time sample tends to zero. In light of these considerations the linearized equations are expressed as

$$\dot{\boldsymbol{X}}(t) = \dot{\boldsymbol{X}}_{des} + \boldsymbol{A}(\boldsymbol{X}_{des})[\boldsymbol{X}(t) - \boldsymbol{X}_{des}] + \boldsymbol{B}\mathcal{F}(t) \tag{14}$$

where the subscript *des* specifies the desired state terms and the matrix $\boldsymbol{A}$ is the state Jacobian matrix computed at the linearization point:

$$\boldsymbol{A}(t) = \left.\frac{\partial \mathcal{G}}{\partial \boldsymbol{X}}\right|_{\boldsymbol{X}_{des}(t)} \tag{15}$$

Introducing the actual error vector as

$$\boldsymbol{X}_{err}(t) = \boldsymbol{X}(t) - \boldsymbol{X}_{des}(t) \tag{16}$$

the complete state space representation of the spacecraft simulator dynamics is then defined by Eq. (17):

$$\dot{\boldsymbol{X}}_{err}(t) = \boldsymbol{A}(\boldsymbol{X}_{des})\boldsymbol{X}_{err}(t) + \boldsymbol{B}\mathcal{F}(t) \tag{17}$$

Thus, given the rotation sequence *yxz*, a symbolic calculation tool was used to derive the parametric expression of the matrix $\boldsymbol{A}$.

In particular the state matrix $\boldsymbol{A}$ is composed of a translational and a rotational term, then gathered generating the global 12×12 state matrix:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_{transl} & \boldsymbol{0}_{6\times6} \\ \boldsymbol{0}_{6\times6} & \boldsymbol{A}_{rot} \end{bmatrix} \tag{18}$$

where

$$\boldsymbol{A}_{transl} = \begin{bmatrix} \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} \end{bmatrix} \tag{19}$$

$$\boldsymbol{A}_{rot} = \begin{bmatrix} \boldsymbol{A}_{rot11} & \boldsymbol{A}_{rot12} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{A}_{rot22} \end{bmatrix} \tag{20}$$

The expression of the state matrix rotational term has been further partitioned due to complexity of its elements, allowing us, to focus on those terms depending on the *yxz* convention, that is on the Euler angles:

$$\boldsymbol{A}_{rot11} = [\boldsymbol{A}_{rot1} \;\; \boldsymbol{A}_{rot2} \;\; \boldsymbol{A}_{rot3}] \tag{21}$$

$$\boldsymbol{A}_{rot1} = \begin{bmatrix} 0 \\ \dfrac{\sin(\theta_x)\{\omega_y[2\ \sin(\theta_z/2)^2 - 1] - \omega_x\ \sin(\theta_z)\}}{\sin(\theta_x)^2 - 1} \\ \dfrac{\omega_y\ \cos(\theta_z) + \omega_x\ \sin(\theta_z)}{\cos(\theta_x)^2} \end{bmatrix} \tag{22}$$

$$\boldsymbol{A}_{rot2} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{23}$$

$$\boldsymbol{A}_{rot3} = \begin{bmatrix} -\omega_y\ \cos(\theta_z) - \omega_x\ \sin(\theta_z) \\ \dfrac{\omega_x\ \cos(\theta_z) - \omega_y\ \sin(\theta_z)}{\cos(\theta_x)} \\ \dfrac{\sin(\theta_x)[\omega_x\ \cos(\theta_z) - \omega_y\ \sin(\theta_z)]}{\cos(\theta_x)} \end{bmatrix} \tag{24}$$

$$\boldsymbol{A}_{rot12} = \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 \\ \dfrac{\sin(\theta_z)}{\cos(\theta_x)} & \dfrac{\cos(\theta_z)}{\cos(\theta_x)} & 0 \\ \sin(\theta_z)\tan(\theta_x) & \cos(\theta_z)\tan(\theta_x) & 1 \end{bmatrix} \tag{25}$$

$$\boldsymbol{A}_{rot22} = \begin{bmatrix} 0 & \dfrac{\omega_z(J_y - J_z)}{J_x} & \dfrac{\omega_y(J_y - J_z)}{J_x} \\ -\dfrac{\omega_z(J_x - J_z)}{J_y} & 0 & -\dfrac{\omega_x(J_x - J_z)}{J_y} \\ \dfrac{\omega_y(J_x - J_y)}{J_z} & \dfrac{\omega_x(J_x - J_y)}{J_z} & 0 \end{bmatrix} \tag{26}$$

The 12×6 input matrix $\boldsymbol{B}$ (where 12 represents the number of the states, while 6 represents the number of the force and torque inputs) remains unvaried, and is given by

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{B}_{transl} & \boldsymbol{0}_{6\times3} \\ \boldsymbol{0}_{6\times3} & \boldsymbol{B}_{rot} \end{bmatrix} \tag{27}$$

where

$$\boldsymbol{B}_{transl} = \begin{bmatrix} \boldsymbol{0}_{3\times3} \\ \boldsymbol{I}_{3\times3}/m \end{bmatrix} \tag{28}$$

$$\boldsymbol{B}_{rot} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \dfrac{1}{J_x} & 0 & 0 \\ 0 & \dfrac{1}{J_y} & 0 \\ 0 & 0 & \dfrac{1}{J_z} \end{bmatrix} \tag{29}$$

Moreover, it was assumed that all the states are observed output. This model provides the inputs of the LQR that given the problem dimensions (12 states and 6 inputs) provides an optimal Kalman gain as a 6×12 matrix to be multiplied by the 12×1 state error $\boldsymbol{X}_{err}$, finally obtaining the optimal solution $\mathcal{F}_{LQR}$. This represents the optimal input generalized force, including both forces and torques, necessary to reach the desired state $\boldsymbol{X}_{des}$, while minimizing the cost functional. However the spacecraft simulator is moved by twelve cold gas thrusters, placed on the upper part of the simulator; hence the generalized optimal force has been converted into a set of twelve unidirectional thruster commands by means of a mapping matrix $\boldsymbol{H}$, as proposed in [24]. According to Fig. 2, thrusters are placed on the edge of the four arms symmetrically connected to the basis of the attitude stage and extended upwards and downwards respectively: specifically eight thrusters are placed with an inclination of 45° with respect to the sides of the imaginary square (side $2d = 0.64$ m) circumscribed to the basis, while the remaining four (thrusters 3, 6, 9, and 12) provide pure vertical motion when the basis is placed horizontally.

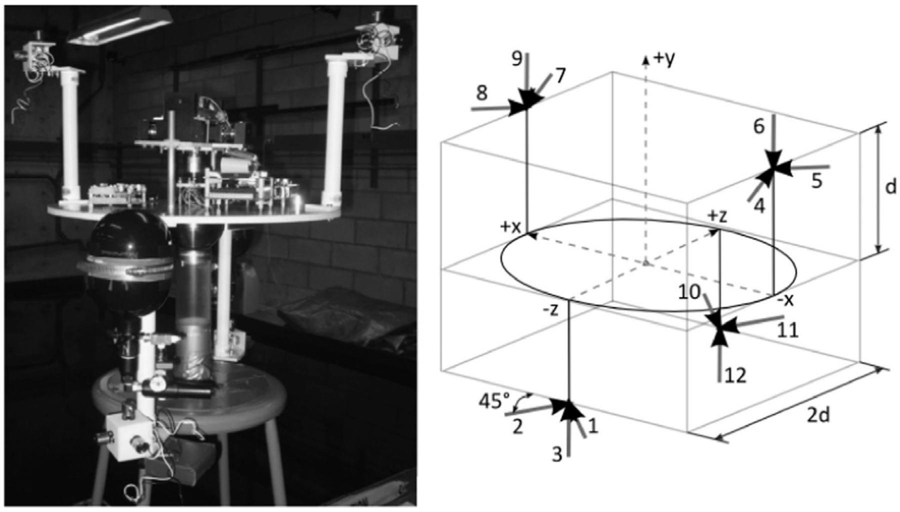


Fig. 2. Axonometric view of the attitude stage (AS) and geometric representation of the thrusters configuration: dotted lines represent the inertial principal axes, black thin lines refer to the AS with the relative upper/lower arms, thick arrows illustrate the thrust direction, light thin lines symbolize the imaginary square circumscribed to the AS base.

The thrust distribution matrix is then given by

$$\boldsymbol{H} = \begin{bmatrix} {}^{E}\boldsymbol{DCM}_B(\boldsymbol{X}_{des})\boldsymbol{H}_f \\ \boldsymbol{H}_m \end{bmatrix} \tag{30}$$

where $\boldsymbol{H}_f$ and $\boldsymbol{H}_m$ represent the force and torque terms respectively.

$$\boldsymbol{H}_f = \begin{bmatrix} p & -p & 0 & p & p & 0 & -p & -p & 0 & -p & p & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 1 \\ p & p & 0 & p & -p & 0 & -p & p & 0 & -p & -p & 0 \end{bmatrix} \tag{31}$$

$$\boldsymbol{H}_m = \begin{bmatrix} -p & -p & 1 & p & -p & 0 & -p & p & 0 & p & p & -1 \\ -p & p & 0 & p & -p & 0 & p & -p & 0 & -p & p & 0 \\ p & -p & 0 & -p & -p & 1 & p & p & -1 & -p & p & 0 \end{bmatrix} d \tag{32}$$

with $p = \cos(45^\circ) = \sin(45^\circ)$ and $d = 0.32$ m. The twelve resulting normalized continuous inputs, $\boldsymbol{U}_{cont}$, have been then computed according to the following expression:

$$\boldsymbol{U}_{cont} = pinv(\boldsymbol{H})\frac{\mathcal{F}_{LQR}}{u_a} \tag{33}$$

$pinv()$ is the pseudoinverse operator, whereas u_a is a dimensionless parameter whose magnitude corresponds to the nominal thrust force $F_t = 0.3$ N, introduced to normalize the values. After the pseuoinverse is computed a pulse-width-modulation strategy is used to finally derive a set of twelve dimensionless binary inputs, $\boldsymbol{u}_{10}$, to be sent to the onboard on/off thrusters. It is worth underlining that the pseudoinverse solution is applied to a constant matrix, posing no computational or solution existence issues.

2.4. Linear quadratic control

The control strategy is sub-optimized since the LQR problem is solved at each time step, using dynamically sized weighting matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$, adapting gains with respect to the actual position reached by the robot. Thus the negative effects deriving from the linearization will be minimized since the whole trajectory will consist of a sequence of linearized sections and will approach the non-linear one as the sample time tends to zero.

The definition of the control law requires the introduction of a state feedback gain whose design is a trade-off between the transient response and the control effort. The optimal control approach [25] to this design trade-off is to define and minimize a performance index; furthermore, since generally the robot has to reach non-zero target position and attitude, a non-zero set point optimal control [26] has been considered, shifting the actual state $\boldsymbol{X}$ by the desired quantity $\boldsymbol{X}_{des}$, obtaining the error, defined in Eq. (34):

$$\boldsymbol{X}_{err}(t) = \boldsymbol{X}(t) - \boldsymbol{X}_{des}(t) \tag{34}$$

Consequently, the shifted optimal regulation problem becomes

$$(LQR)\begin{cases} \text{Minimize} & J = \frac{1}{2}\int_{t_0}^{\infty}\left[\boldsymbol{X}_{err}^T(t)\boldsymbol{Q}\boldsymbol{X}_{err}(t) + \boldsymbol{U}^T(t)\boldsymbol{R}\boldsymbol{U}(t)\right]dt \\ \text{Subject to} & \dot{\boldsymbol{X}}_{err}(t) = \boldsymbol{A}\boldsymbol{X}_{err}(t) + \boldsymbol{B}\boldsymbol{U}(t) \\ & \boldsymbol{Y}(t) = \boldsymbol{X}_{err}(t) \end{cases} \tag{35}$$

with $\boldsymbol{A}$ and $\boldsymbol{B}$ from Eqs. (18) and (27) respectively, and where the state weighting matrix $\boldsymbol{Q}$ is assumed symmetric and positive semi-definite and the input weighting matrix $\boldsymbol{R}$ symmetric and positive definite.

In this case, the LQR generates the optimal solution $\boldsymbol{U}^*(t)$ (all the following functions are listed as functions of time, as they depend on $\boldsymbol{X}_{des}$ which depends on time)

$$\boldsymbol{U}^*(t) = -\boldsymbol{K}(t)\boldsymbol{X}_{err}(t) = -\boldsymbol{R}^{-1}(t)\boldsymbol{B}^T\boldsymbol{P}(t)\boldsymbol{X}_{err}(t) \tag{36}$$

where $\boldsymbol{K}(t)$ is often referred to as Kalman gain and $\boldsymbol{P}(t)$ is the solution of the Riccati differential equation,

$$\boldsymbol{P}(t)\boldsymbol{A}(t) + \boldsymbol{A}(t)^T\boldsymbol{P}(t) - \boldsymbol{P}(t)\boldsymbol{B}\boldsymbol{R}^{-1}(t)\boldsymbol{B}^T\boldsymbol{P}(t) + \boldsymbol{Q}(t) = \dot{\boldsymbol{P}}(t) = 0 \tag{37}$$

obtaining the following linear state feedback control:

$$\dot{\boldsymbol{X}}_{err}(t) = (\boldsymbol{A}(t) - \boldsymbol{B}\boldsymbol{K}(t))\boldsymbol{X}_{err}(t)$$

$$\boldsymbol{U}(t) = -\boldsymbol{K}(t)\boldsymbol{X}_{err}(t) \tag{38}$$

The presented control strategy has been simulated employing a Simulink integrated MATLAB function, which is based on the original code employed in the 'lqr' MATLAB command [27]. It requires the following input matrices: $\boldsymbol{A}$ (dynamic matrix), $\boldsymbol{B}$ (control matrix), $\boldsymbol{C}$ (state-output mapping matrix), $\boldsymbol{D}$ (control-output mapping matrix or feedforward matrix), and $\boldsymbol{Q}$ and $\boldsymbol{R}$ weighting matrices. Lastly, it provides the optimal gain matrix $\boldsymbol{K}$, the solution of the Riccati equation $\boldsymbol{S}$ and the closed-loop eigenvalues of the matrix $(\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K})$, allowing us to verify whether or not the system has been stabilized, obtaining negative real part eigenvalues. The presented function executes the routine, in both linux and win32/64 environment where the correspondent C code is generated for compilation under RTAI linux.

Concerning the weighting matrices, the tuning technique is based on the use of constant elements q_i and r_i for both the matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$, as part of a trial and error iterative process where each weight and the factor ρ are gradually adjusted with the aim of obtaining the desired performances, that is the reaching of the target position.

Moreover, two additional gains K_{pos} and K_{rot} are introduced in the $\boldsymbol{Q}$, and the weighting factor ρ becomes variable throughout the simulations as well as the experimental test, depending on these two new gains, thus increasing the control sensitivity to the system evolution.

The deriving matrices have the following structure:

$$\boldsymbol{R} = \rho\begin{bmatrix} r_{F_x} & 0 & 0 & & & \\ 0 & r_{F_y} & 0 & & \boldsymbol{0}_{3\times3} & \\ 0 & 0 & r_{F_z} & & & \\ & & & r_{M_x} & 0 & 0 \\ & \boldsymbol{0}_{3\times3} & & 0 & r_{M_y} & 0 \\ & & & 0 & 0 & r_{M_z} \end{bmatrix} \tag{39}$$

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{Q}_{transl} & \boldsymbol{0}_{6\times6} \\ \boldsymbol{0}_{6\times6} & \boldsymbol{Q}_{rot} \end{bmatrix} \tag{40}$$

where

$$\mathbf{Q}_{transl} = \begin{bmatrix} K_{pos}q_x & 0 & 0 & & & \\ 0 & K_{pos}q_y & 0 & & \mathbf{0}_{3\times3} & \\ 0 & 0 & K_{pos}q_z & & & \\ & & & q_{v_x} & 0 & 0 \\ & \mathbf{0}_{3\times3} & & 0 & q_{v_y} & 0 \\ & & & 0 & 0 & q_{v_z} \end{bmatrix} \quad (41)$$

$$\mathbf{Q}_{rot} = \begin{bmatrix} K_{rot}q_{\theta x} & 0 & 0 & & & \\ 0 & K_{rot}q_{\theta y} & 0 & & \mathbf{0}_{3\times3} & \\ 0 & 0 & K_{rot}q_{\theta x} & & & \\ & & & q_{\omega x} & 0 & 0 \\ & \mathbf{0}_{3\times3} & & 0 & q_{\omega y} & 0 \\ & & & 0 & 0 & q_{\omega z} \end{bmatrix} \quad (42)$$

Specifically, two controls, over the spacecraft position and attitude respectively, allow these new gains to switch between two calibrated values, on the basis of the actual position and attitude:

- once the i-th component of the position or attitude gets into an expressly chosen band, the boundaries of which have been defined as percentages of the relative target component (i.e. $\pm 20\%$ of the target position), a counter variable starts to run;
- when the counter reaches a predefined value the relative K gain (K_{transl} or K_{rot}) switches to a second value appropriately calibrated for the steady state phase (robot near the target position);
- these new K gains are applied to the position and attitude terms of the $\mathbf{Q}$ (Eqs. (42) and (43)), thus forcing a change of the weights as the state approaches the target;
- when both the position and the attitude gains K_{pos} and K_{rot} have changed, the factor ρ is also allowed to change, hence switching to a new value which is more suitable for the steady state phase.

3. Comparison with GPOPS-II

A validation of the proposed sub-optimal controller, implemented in Matlab and Simulink, is carried out by comparing it with an optimizer to highlight how well the proposed controller approximates the best solution. At the time of writing, several open-source, freeware and commercial optimal control and nonlinear programming interfaces are available. However, results accuracy, advancements in mesh refinement and generality of problem formulation represent some of the reasons behind the choice of GPOPS-II as the basis for comparison with the Simulink model. GPOPS-II available at [28] has been developed at the University of Florida in cooperation with the U.S. Office of Naval Research (ONR) and the U.S. Defense Advanced Research Projects Agency (DARPA). Turning to detail, it employs an adaptive Radau pseudospectral Gaussian quadrature method and a sparse finite-differencing is used to estimate all first and second derivatives required by the nonlinear programming (NLP) solver. Moreover, it has been designed to work with the NLP solvers Sparse Nonlinear OPTimizer (SNOPT) and Interior Point OPTimizer (IPOPT) and to be extremely flexible, allowing a user to formulate a wide variety of applications including engineering, economics, and medicine.

The optimal control problem is then stated as a NLP problem (refer to Eqs. (12) and (44), whose differential algebraic equations are collocated using nodes obtained from a Gaussian quadrature, whereas the state and the control are parameterized using Legendre polynomials and their linear combinations. As stated in [29], the combination of Legendre polynomials with Gaussian quadrature ensures exponential convergence for smooth solution problems. More specifically, in order to find the optimal solution, that is a combination of state and input vector solutions minimizing the objective function in the Bolza form [25], the general purpose optimal control software GPOPS-II uses a set of Legendre Gauss Radau (LGR) collocation points: this set is defined on the domain $[-1,1]$, but contains only one of the endpoints. According to [29,30], the whole iterative procedure, employed by GPOPS-II to determine the optimal solution, could be efficaciously divided into five steps representing partial results of the whole method:

1. identification of the first-order optimality conditions of the continuous Bolza problem, on the basis of the Pontryagin Minimum Principle [25];
2. Radau pseudospectral discretization of the continuous-time first-order optimality conditions of the continuous Bolza problem;
3. Radau pseudospectral discretization of the continuous time optimal control problem, resulting in a discrete NLP;
4. statement of the Karush–Kuhn–Tucker (KKT) conditions related to the NLP;
5. costate estimation obtained from the results of steps 3 and 4.

A new model of the spacecraft simulator has then been developed, introducing the same rotational and translational equation of motion (12), but following, this time, according to the GPOPS-II logic design [31], a different problem formulation. Two phases have been considered, in order to analyze both the transient response and the steady state one, introducing the appropriate linkage constraints in the endpoint function to ensure continuity of the solution and set time and event criteria, defining the transit from the previous phase to the next one. In this way, during the mesh iterations, when an optimal solution is found, the mesh is analyzed in each of these phases, verifying if the mesh error tolerance is satisfied; if not, the solver automatically increases the number of collocation points, updating the mesh refinement and generating a non uniform adaptive grid designed to reduce the error.

Finally, the following aspects have been considered to obtain a correct comparison between the approach proposed herein and GPOPS one.

1. The comparison of the LQR method and GPOPS has been focused on the global cost required by the controller. Specifically the global dimensionless cost J, that is the cost or objective functional to be minimized both by the LQR and the optimal solver, consists of two terms P and F, each related to the generalized position and velocity

Table 1
Costs comparison.

Test		Initial state	Final state	Simulink			GPOPS-II			ΔJ% (–)
				F (–)	P (–)	J (–)	F (–)	P (–)	J (–)	
1	x, y, z (m)	(1,1.4,−1)	(−1,1.85,2)	63.3	173.3	**236.6**	53.3	160.4	**213.7**	**9.7**
	$\theta_x, \theta_y, \theta_z$ (deg)	(50,0,50)	(−30,120,30)							
2	x, y, z (m)	(1,1.4,0)	(2,1.85,1)	15	39.5	**54.5**	11.1	33.9	**45**	**17.4**
	$\theta_x, \theta_y, \theta_z$ (deg)	(0,30,−20)	(10,120,10)							
3	x, y, z (m)	(1,1.4,0)	(1,1.4,0)	3.9	11.3	**15.2**	2.8	9	**11.8**	**22.4**
	$\theta_x, \theta_y, \theta_z$ (deg)	(20,40,10)	(0,150,10)							
4	x, y, z (m)	(1,1.4,0)	(1.5,1.9,1)	9.3	18.7	**28**	5.8	17.4	**23.2**	**17.1**
	$\theta_x, \theta_y, \theta_z$ (deg)	(20,40,−5)	(20,40,−5)							
5	x, y, z (m)	(2,1.4,0)	(0,1.4,1)	30.5	76.8	**107.3**	22.5	68	**90.5**	**15.7**
	$\theta_x, \theta_y, \theta_z$ (deg)	(20,160,0)	(0,70,30)							
6	x, y, z (m)	(2,1.4,0)	(0,1.4,1)	34.1	88.7	**122.8**	25.4	77.7	**103.1**	**16.0**
	$\theta_x, \theta_y, \theta_z$ (deg)	(20,200,0)	(0,0,30)							
7	x, y, z (m)	(2,1.9,1)	(1,1.4,1)	9.6	27	**36.6**	7.7	23.8	**31.5**	**13.9**
	$\theta_x, \theta_y, \theta_z$ (deg)	(10,300,10)	(−20,45,0)							
8	x, y, z (m)	(−1,1.9,2)	(0,1.4,0)	19.2	75.4	**94.6**	21.4	64.7	**86.1**	**9.0**
	$\theta_x, \theta_y, \theta_z$ (deg)	(30,60,−20)	(0,0,0)							

accuracy, and the control effort respectively. Since the Simulink model employs a fixed time step of 0.02 s and a discrete solver, the cost function has been computed using the following discrete expression, to match the same type of cost function generated by GPOPS. This summation approximates the integral cost function.

$$J = \sum_{k=1}^{\infty} [P(k) + F(k)] = \frac{1}{2} \sum_{k=1}^{\infty} \left[\boldsymbol{X}_{err}^T(k) \boldsymbol{Q} \boldsymbol{X}_{err}(k) + \boldsymbol{U}^T(k) \boldsymbol{R} \boldsymbol{U}(k) \right] \tag{43}$$

In this analysis more importance has been attributed to the propellant cost, setting the tuning parameters in order to save as much propellant as possible.

2. Since both the optimal solver GPOPS and the LQR solver cannot handle binary variables, the propellant term has been computed using the continuous control vector $\boldsymbol{U}_{cont}$; hence, during these simulations, the block used in the Simulink model to transform the continuous control into a binary series of on/off commands has been bypassed.
3. For the sake of simplicity the constant tuning technique (refer to Section 2.4) has been applied in both models, using also the same numeric values, thus to observe the results on equal weighting terms.

One last expedient has been devised to make the comparison even more truthful: the phase of post-processing transforms the results obtained with GPOPS, interpolating data and then extracting values on the basis of the equally spaced time vector used in Simulink. This technique allows a point by point comparison, hence a more detailed analysis of the results and their discrepancy.

The most important data affecting the outcome of this comparison, hence the Simulink model performances, were the simulation costs. A set of eight simulations has then been executed both using the Simulink and the GPOPS software, generating Table 1, which summarizes the cost values as well as the initial and desired condition for each of these eight tests. It is worth reminding that the translational dynamics is linear, while the rotational is nonlinear. The initial and final values were randomly chosen to cover a wide range of maneuvers.

From a statistical analysis it can be inferred that on average the sub-optimal controller requires 15% additional cost. However, although the GPOPS controller seems to be the best alternative, some observations have to be taken into account before expressing the final evaluation:

- Both Simulink and GPOPS simulations have been run using a laptop with standard performances, at the time of writing (Processor Intel Core i3, RAM 2 GB). Given a fixed time simulated duration of 200 s in either case, the optimal solver definitely incurred longer computation time; specifically, setting a mesh tolerance between 1e-5 and 1e-3 (suggested values 1e-6, default value 1e-3), each simulation required on average 600–700 s, three times the simulated time, while, using Simulink, simulations required 200–500 s. Even though the LQR appears slightly faster in simulation, this comparison does not show such a substantial gap between the two techniques yet. There is, in fact, an additional fundamental observation that needs to be made: while GPOPS needs to compute an entire trajectory before returning a solution, the proposed approach can solve the LQR problem in real time, i.e., it decides how to actuate for the immediate next time step.
- The LQR solution constitutes a linear state feedback control, generating a closed loop system, while the multi-purpose optimal control software GPOPS-II provides a guidance, that is the determination of an entire desired trajectory, which implies incapability to execute in real time. In fact, a 600 s convergence time for a 200 s simulation clearly shows inability for real-time implementation.

To summarize, the Simulink theoretical controller represents the best alternative: in particular the feedback nature and the higher level of flexibility of the controller allow the use of the sub-optimal solution for real-time implementation; furthermore the previous cost analysis ensures that the same sub-optimal controller maintains a high optimization level, since the costs discrepancy is 15%, compensated by implementability with a real spacecraft.

4. The six-degree-of-freedom spacecraft simulator testbed

Once the reliability and the effective optimization level of the Simulink model were proven, a series of experimental tests were conducted on the ADAMUS test bed [32,33].

This test bed consists of a six-degree-of-freedom spacecraft simulator based on an air bearing technology that allows it to move as it would in space, i.e., torque and force free. The so-called moving platform (MP) is controlled by twelve cold gas thrusters, appropriately placed on the upper part of the simulator. The MP is powered by two Lithium-ions batteries which are connected to an on board power management system. Moreover, a 13×15 ft (3.96×4.57 m) flat epoxy floor (Fig. 3) is being used as the main base upon which to move the robot.

The overall system represented in Fig. 4 consists of two main stages:

- The translational stage (TS) provides near frictionless planar translational motion by means of three linear air bearings, to transform operative conditions (1-g) into near-microgravity ones; the near gravity free vertical translation is instead obtained introducing a system of near-frictionless air bearing pulleys and variable-mass counterbalances.
- The attitude stage (AS) represents the real spacecraft simulator and is connected to the TS through a spherical segment air bearing; it gives rise to the roll, pitch, yaw degree-of-freedom and hosts the onboard computer and position tracking system housing together with the thrusters supports.

This test bed belongs to the most complex category of simulators, the category of combination systems, which integrates the capabilities of translational systems with those of rotational systems.

The ADAMUS robot, unlike all the other existing six-degree-of-freedom systems [34–40], ensures more realistic simulations, hence results, thanks to the dynamical reproduction of motion along the full 6 degrees of freedom and in particular along the vertical translational degree of freedom. Another interesting feature of the ADAMUS platform is the flexibility: by simply substituting the attitude stage, different categories of

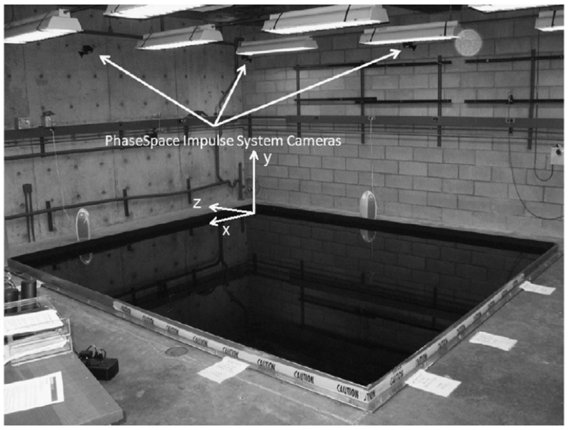


Fig. 3. The reference frame in the ADAMUS laboratory [32].

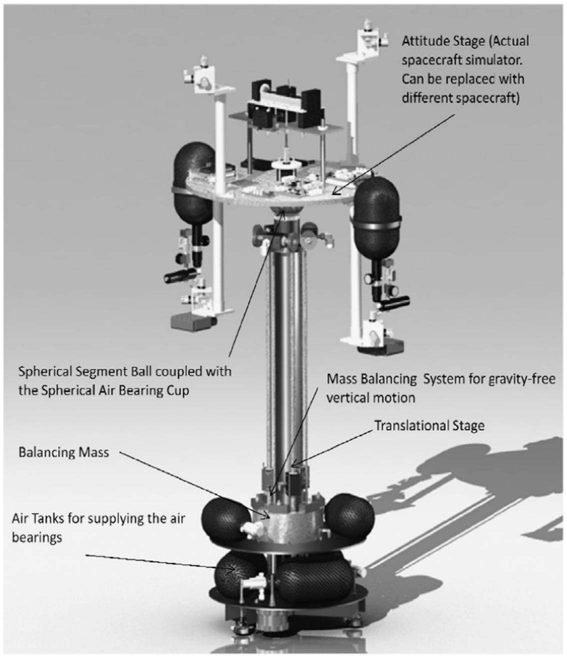

Attitude Stage (Actual spacecraft simulator. Can be replacec with different spacecraft)
Spherical Segment Ball coupled with the Spherical Air Bearing Cup
Mass Balancing System for gravity-free vertical motion
Translational Stage
Balancing Mass
Air Tanks for supplying the air bearings

Fig. 4. The two stages and the main components of the moving platform [32].

spacecraft could be tested, considerably extending its scope of applications. Additionally, special attention has been given to the mass balancing system: a Balancing Platform (BP) allows the robot to modify, by means of three linear motors, the AS center of mass position.

4.1. Translational stage

The translational state (TS), built by Guidance Dynamics Corporation® (GDC), is made up of an horizontal basis, connected to the linear air bearings providing the near-frictionless contact with the epoxy floor, and of a column with a sleeve for the vertical translational motion, at the top of which a spherical air bearing cup and its correspondent spherical segment ball allow the TS to be connected with the AS. Moreover, the TS contains two groups of tanks: the first group, placed on the lower base of the TS, supplies the compressed air to the air bearings, while the second one, placed on the intermediate base, feeds the air pulleys for the near gravity free vertical motion.

4.2. Attitude stage

The attitude stage (AS), designed and built by the ADAMUS lab, provides $\pm 40°$ about the pitch and roll axes and 360° of yaw motion. The main body of the attitude stage consists of a discoid basis of composite material (fiber glass and high density foam) containing the slots of the onboard computer, the power management system, the motor drives and the controller card. Four arms are symmetrically connected to the basis and extend two upwards and two downwards respectively. Three thrusters per arm are located on their edges, along three mutually orthogonal directions.

Furthermore, both tanks, providing the compressed air to the thrusters, and Lithium-ion batteries are connected to the lower arms for the sake of the AS stability. Lastly, 6 LEDs are distributed over the 4 arms and the AS base; these LEDs, together with the puck, are essential components of the PhaseSpace Impulse System, an optical motion tracker, designated to the position tracking.

5. Real-time experimentation

The guidance, navigation and control algorithms were created in Simulink, generating an executable file for Real-Time Application Interface (RTAI) Linux. The Simulink file used to program the experiments is based on the ideal Simulink file employed for the comparison with GPOPS, but has several differences listed in the following.

A set of s-functions was used to interface algorithms and hardware on the robot, providing a way to store, save

Table 2
First experimental test: simulation data.

	x (m)	y (m)	z (m)
Initial conditions	0.25	0.95	−4.21
Desired state	1.5	1.1	−2.5
	θ_x (deg)	θ_y (deg)	θ_z (deg)
Initial conditions	11.5	142.6	−1.4
Desired state	10	125	−10

Table 3
First experimental test: weighting matrices data.

$\boldsymbol{Q}_{12\times12}$	K_{pos}		K_{ang}	
	Trans	Steady	Trans	Steady
$30^{-1}eye(12)$	1	100	0.1	100
$\boldsymbol{R}_{12\times12}$		ρ		
		Trans	Steady	
$eye(6)$		10	10	

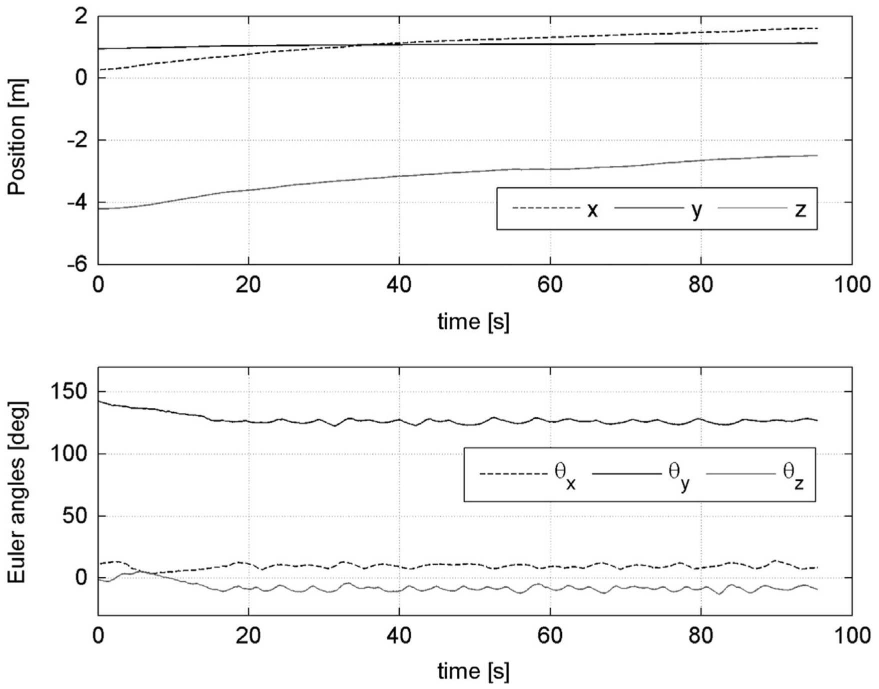


Fig. 5. Experimental results. First test: actual position (top) and actual Euler angles (bottom).

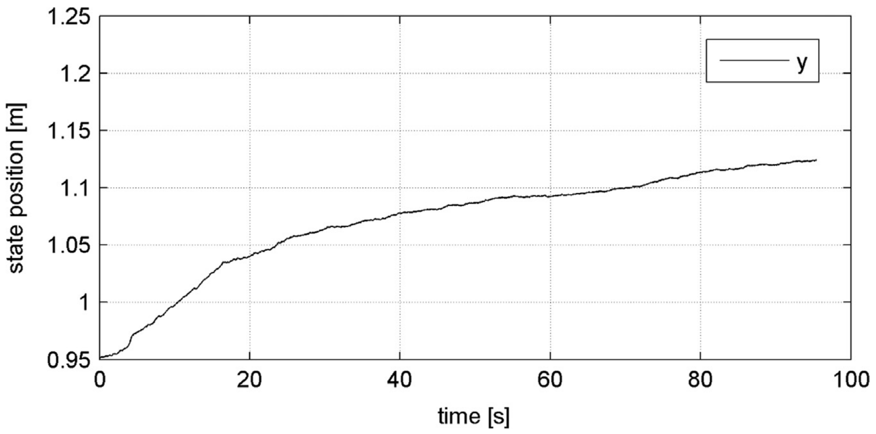


Fig. 6. Experimental results. First test: vertical motion.

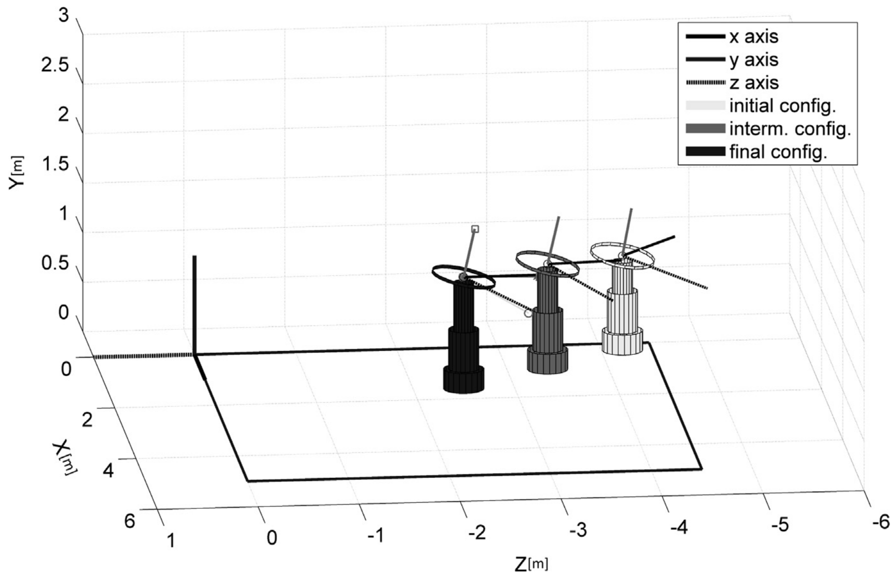


Fig. 7. Experimental results. First test: representation characterized by the data illustrated in Table 2.

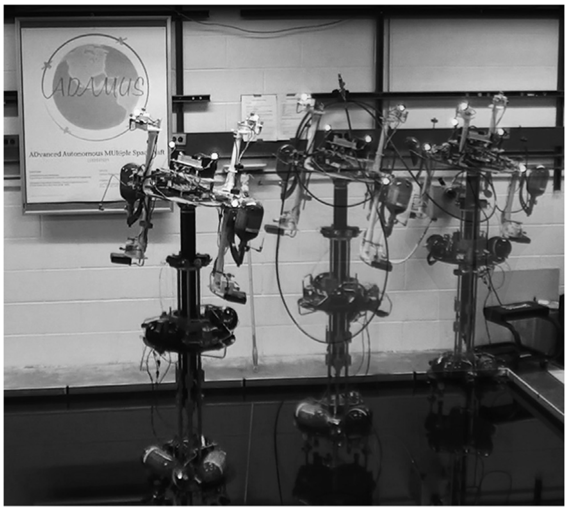


Fig. 8. Experimental results. First test: snapshot of experiment of Fig. 5.

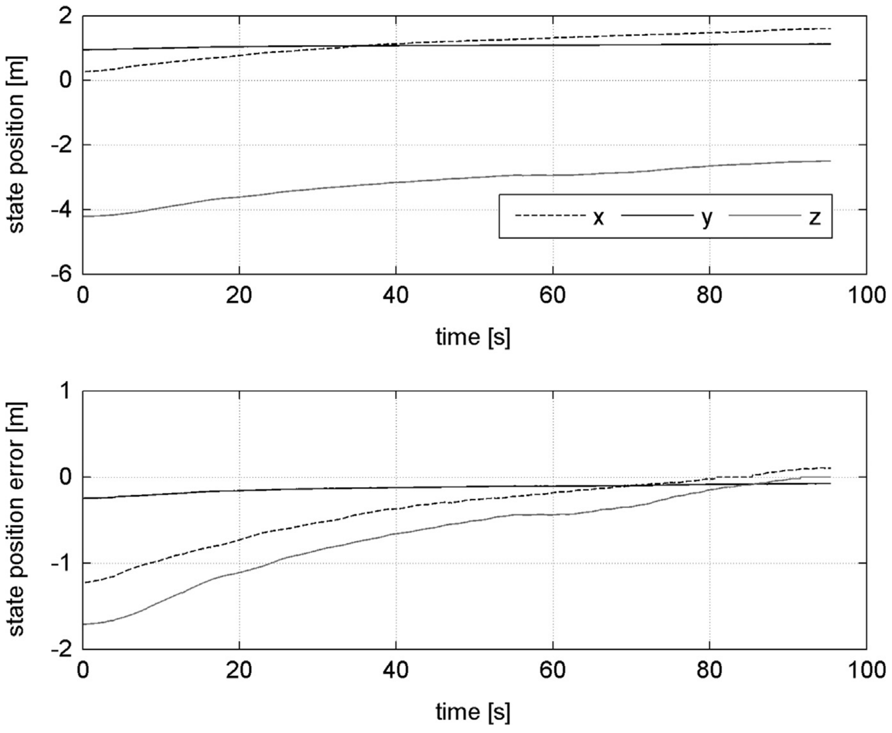


Fig. 9. Experimental results. First test: position (top) and position error (bottom).

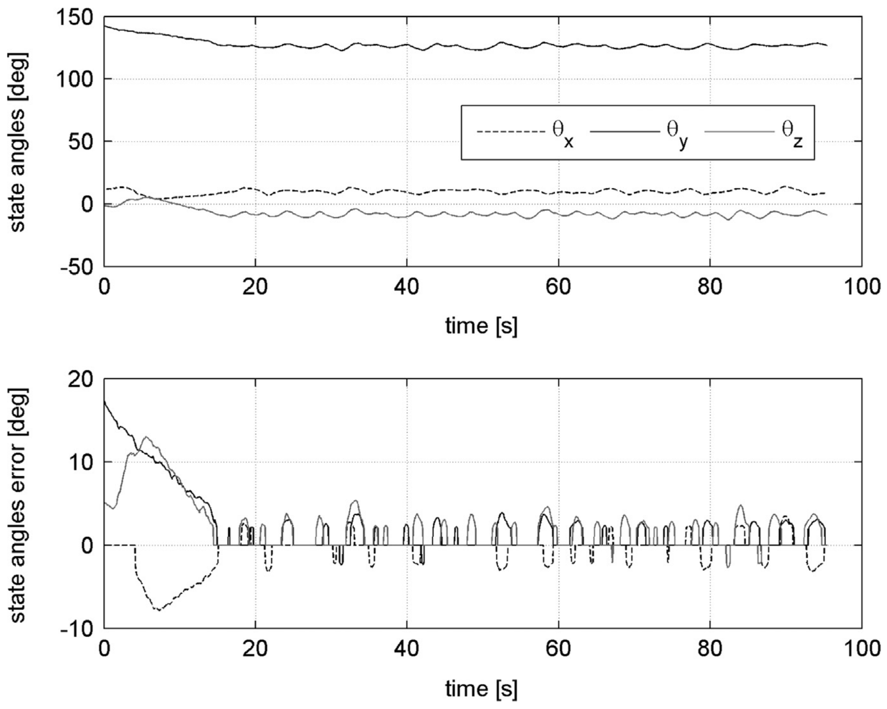
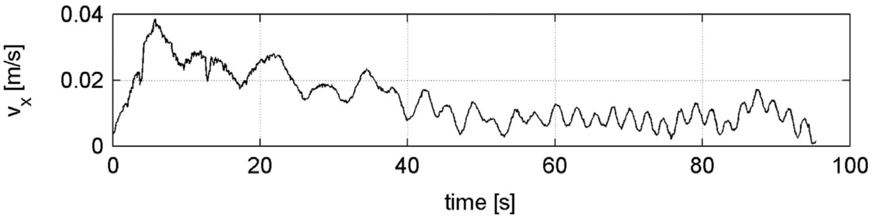
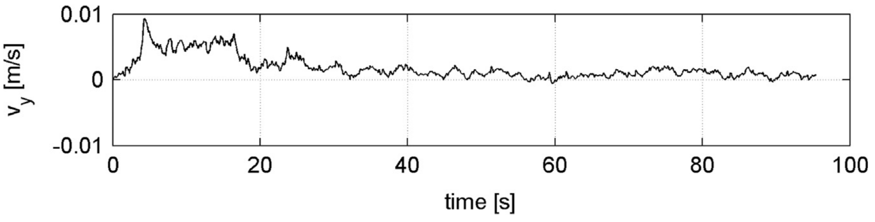


Fig. 10. Experimental results. First test: Euler angles (top) and Euler angles error (bottom).

or modify the desired variables and to send the sub-optimal control vector to the actuators' drivers.

A linear Kalman filter (LKF) and an extended Kalman filter (EKF) [41,42] provided a way to estimate the translational state (position and linear velocities) and the rotational state (Euler angles and the angular velocities) respectively, in such a manner that the error is minimized statistically by means of two-step algorithms (predict/





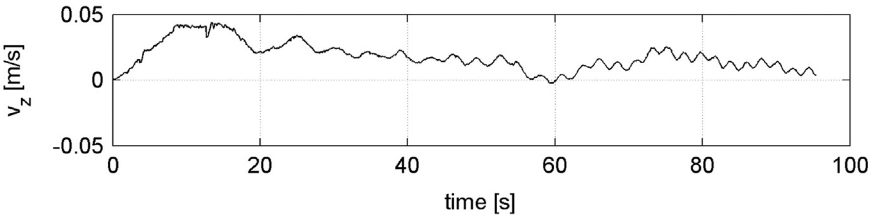
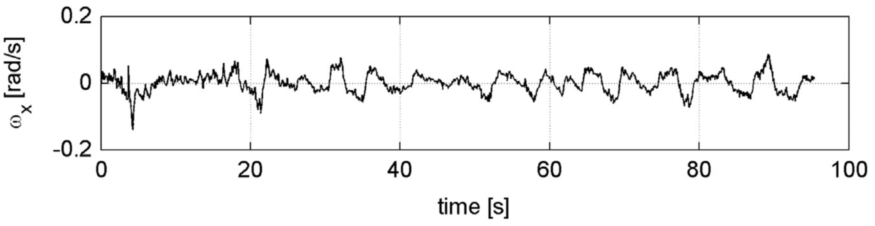
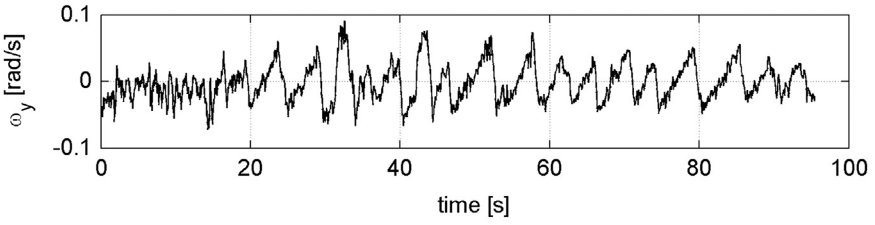


Fig. 11. Experimental results. First test: linear velocity.





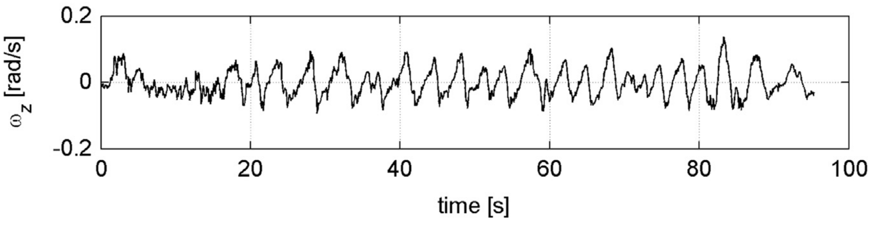


Fig. 12. Experimental results. First test: angular velocity.

update steps). The laboratory motion tracking system provides center of mass position and quaternions, whose processing results in position, attitude, and corresponding velocities from the filters.

The above differences justify the final comparison between the cost obtained from the experiments and the cost obtained running matching simulations via the above described Simulink file, which better represents the hardware.

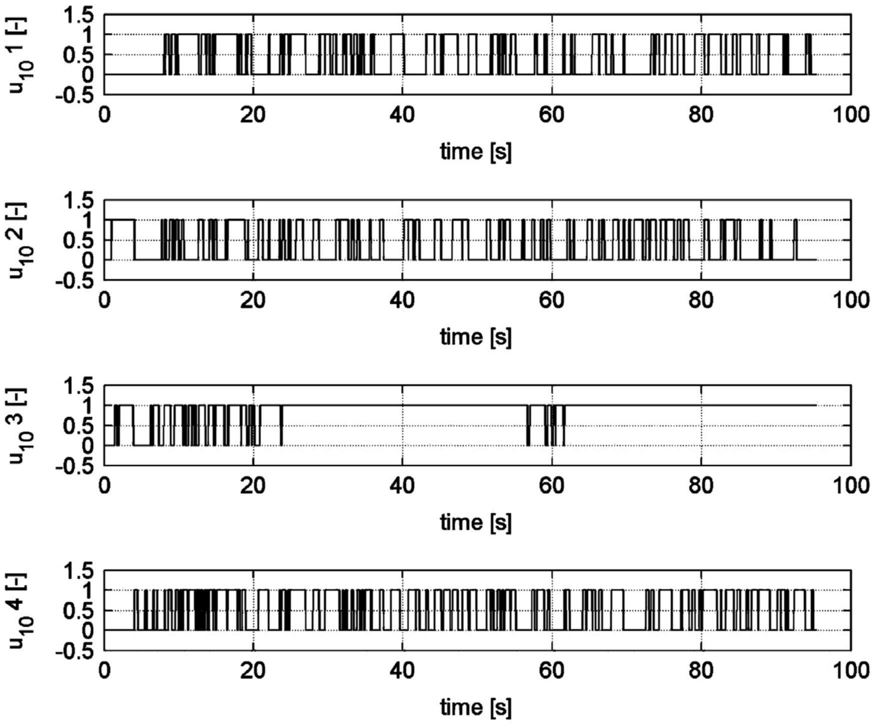


Fig. 13. Experimental results. First test: thrusters configuration 1–4.

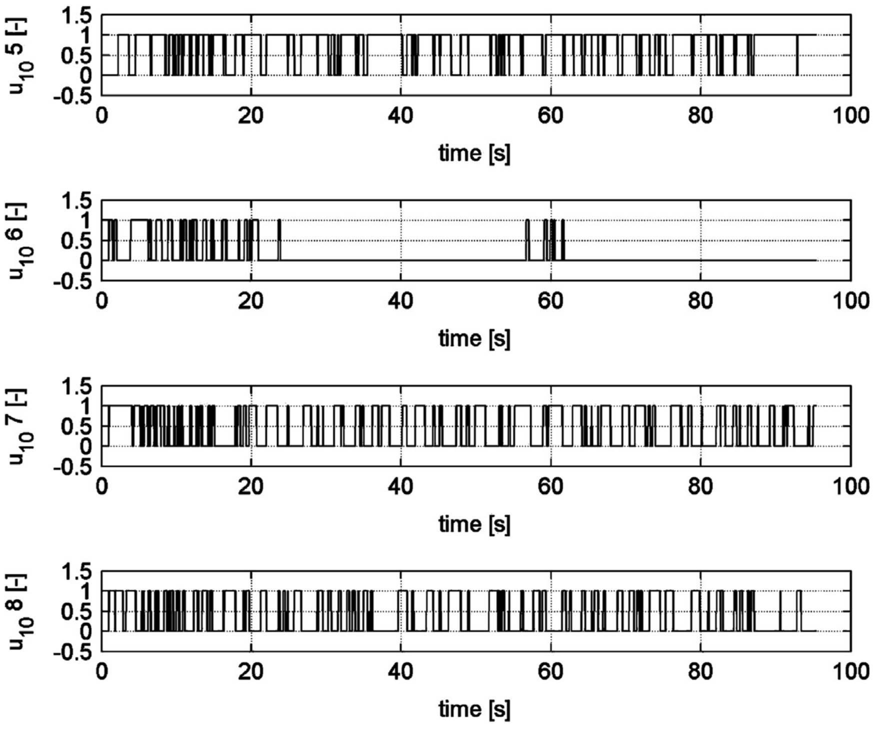


Fig. 14. Experimental results. First test: thrusters configuration 5–8.

Lastly, previous model employed to control the ADAMUS spacecraft simulator used angles error threshold between 8 and 5°, while in the present work the threshold was set to only 2° for the angles and 2 cm for the positions; these assumptions require higher control performances, but, at the same time, allow the controller to increase position accuracy reducing the steady state oscillations.

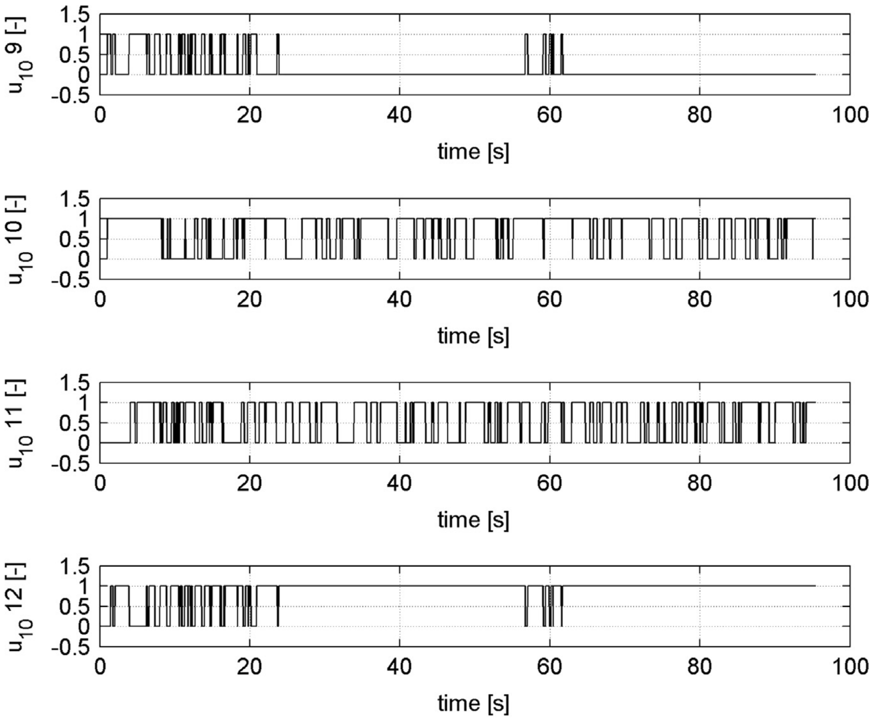


Fig. 15. Experimental results. First test: thrusters configuration 9–12.

Table 4
Second experimental test: simulation data.

	x (m)	y (m)	z (m)
Initial conditions	0.25	1.3	−4.28
Desired state	1.5	1.2	−2.5
	θ_x (deg)	θ_y (deg)	θ_z (deg)
Initial conditions	6.5	31.6	−2.3
Desired state	10	125	−10

Table 5
Second experimental test: weighting matrices data.

$\mathbf{Q}_{12\times12}$	K_{pos}		K_{ang}	
	Trans	Steady	Trans	Steady
$30^{-1}eye(12)$	1	100	0.1	100
$\mathbf{R}_{12\times12}$		ρ		
		Trans	Steady	
$eye(6)$		10	10	

5.1. Experimental results

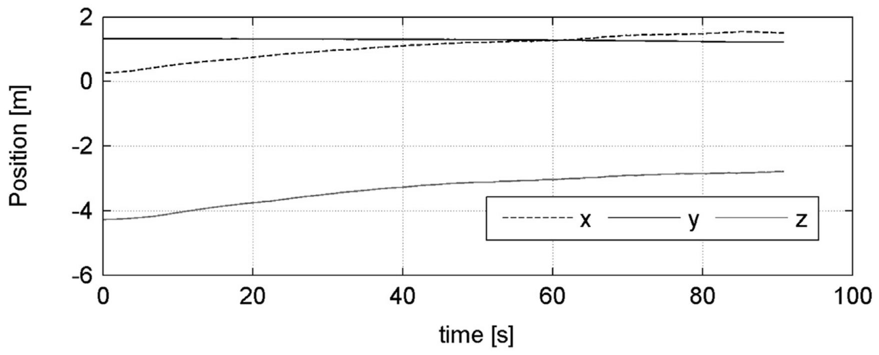
Two examples of full six-degree-of-freedom experiments are shown in the present subsection.

Table 2 summarizes the initial and desired conditions of the first full motion test, Table 3 specifies the weighting matrices values, while Fig. 5 describes its main results.

Since the vertical motion range is small on this first test, an additional representation of the vertical degree of freedom y is provided separately in Fig. 6:

A visualization of the experiment is presented in Figs. 7 and 8.

Figs. 9–15 describe the complete response, including the error which correctly reaches the zero value for each



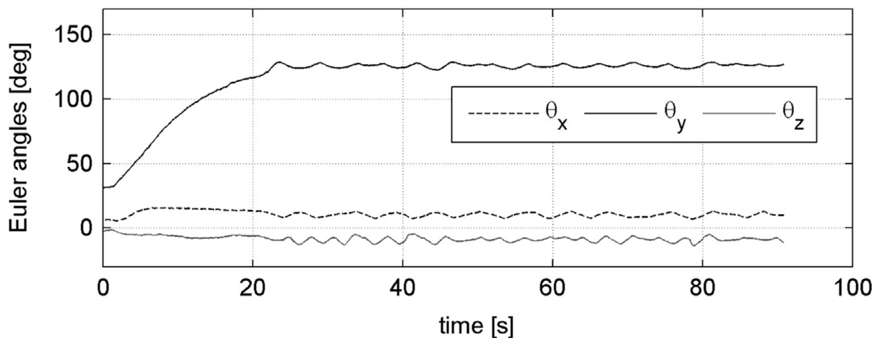


Fig. 16. Experimental results. Second test: position (top) and Euler angles (bottom).

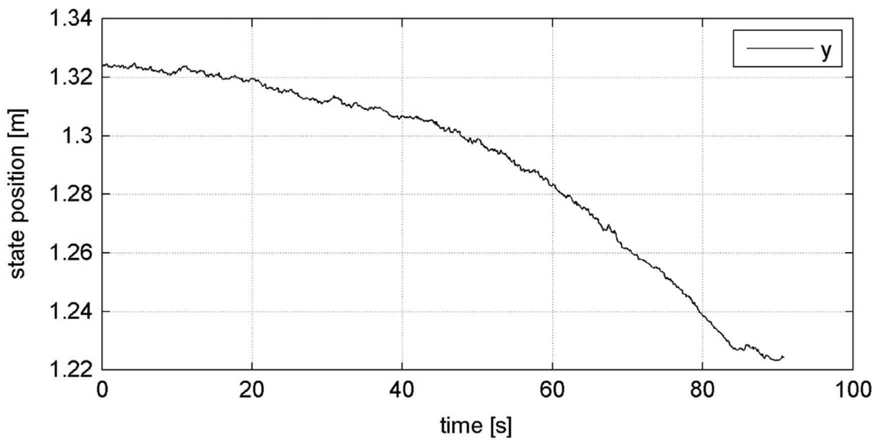


Fig. 17. Experimental result. Second test: vertical motion.

component of the state vector or remains within the relative threshold. This set of full motion tests allowed the authors to study how the real controller could handle the translational and rotational dynamics deriving differences: the translational dynamics is much slower than the rotational one, reaching the target position in about eighty seconds. On the other hand, this behavior influences also the attitude control: much more thrust is in fact required to let the robot translate, hence stressing the attitude control that in this way easily tends to oscillate or possibly overshoot. Consequently the attitude control comes out to be the most critical component; nevertheless, thanks to the last tuning adjustments, overshoot and oscillations have been greatly contained obtaining satisfactory results. In this case, in fact, the three controlled angles reach the desired values in about twenty seconds, exhibiting only minimal steady state oscillation.

Figs. 11 and 12 illustrate the translational velocity and angular velocity. Since the desired values of both velocity terms have been set to zero, the velocity error comes out to be equal to the actual value except for the effect of the error threshold and thus have been omitted for the sake of conciseness.

Lastly since the present test requires an increment in the vertical position of the robot's attitude stage, thrusters 3 and 12, providing positive vertical motion, remain activated for most of the time window, after a transient phase; these results represent the proof of a successful thruster mapping allowing to convert the triaxial forces and torques into a binary configuration of twelve on/off thrusters.

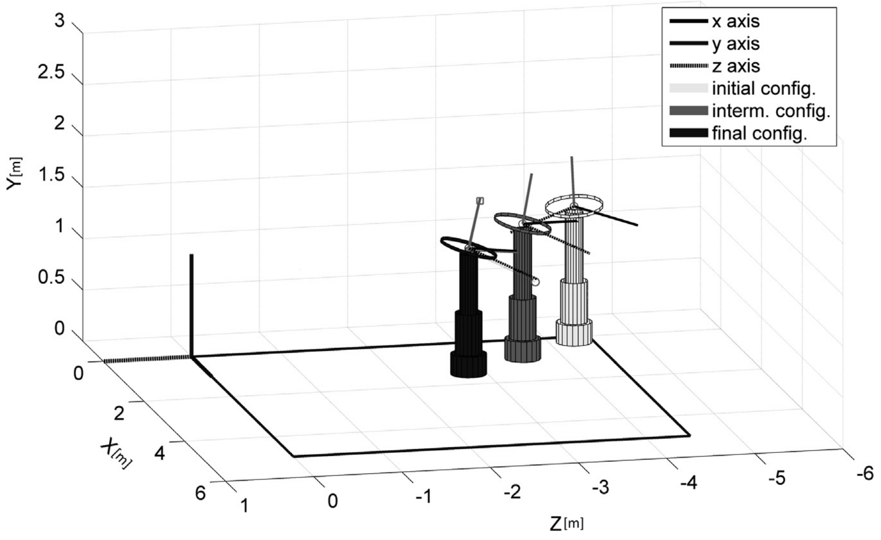


Fig. 18. Experimental results. Second test: representation characterized by the data illustrated in Table 4.

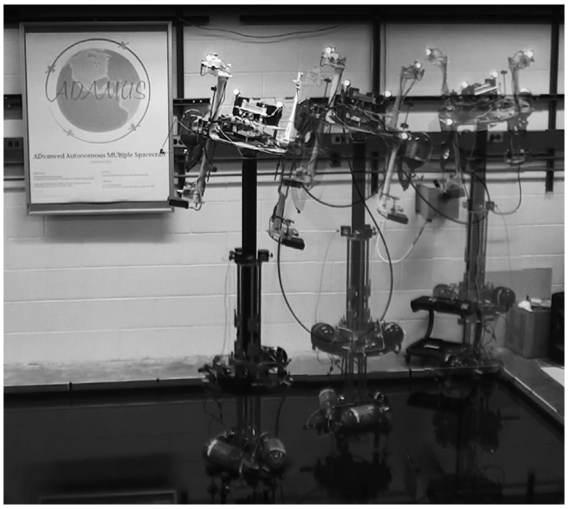


Fig. 19. Experimental results. Second test: snapshot of experiment of Fig. 16.

The second example tests the response to a negative vertical translation, starting from different initial conditions, as shown in Table 4, while Table 5 summarizes the weighting matrices used values.

Following the same order of the previous test, Figs. 16 and 17 summarize the main results of this second test.

The minimal representation and the comparison with the real robot motion are described in Figs. 18 and 19.

Figs. 20–26 provide a more detailed illustration of the complete response, including the actual state components as well as the error components. Similar conclusions can be drawn here as those provided in the previous test.

In this second case the vertical thrusters present an opposite configuration with respect to the previous test: the vertical position, in fact, must decrease, activating, after a transient phase, thrusters 6 and 9 providing negative vertical motion.

Finally, an additional analysis concerning the propellant cost has been conducted, with the aim to demonstrate the discrepancy between the Simulink model and the experiment: the propellant cost term has been computed as a net propellant expenditure as defined in the following equation:

$$U_{cost} = \boldsymbol{u}_{10}^{T}\boldsymbol{u}_{10} \tag{44}$$

On average the propellant cost gap between the simulated solution and the real one corresponds to 3–5% of the real cost. However with equal propellant cost the simulated solution provides a better control, while with equal control the simulated solution requires less propellant.

The consistency between the simulations and the experimental tests could be further increased by incorporating the dual effects of the friction, slowing down the translational motion but at the same time reducing the oscillations, and improving the mass balancing system in order to ensure higher stability in the attitude stage.

The coupling between translation and rotation, by means of the same control inputs, is such that the attitude control is highly affected by the translational one, the latter

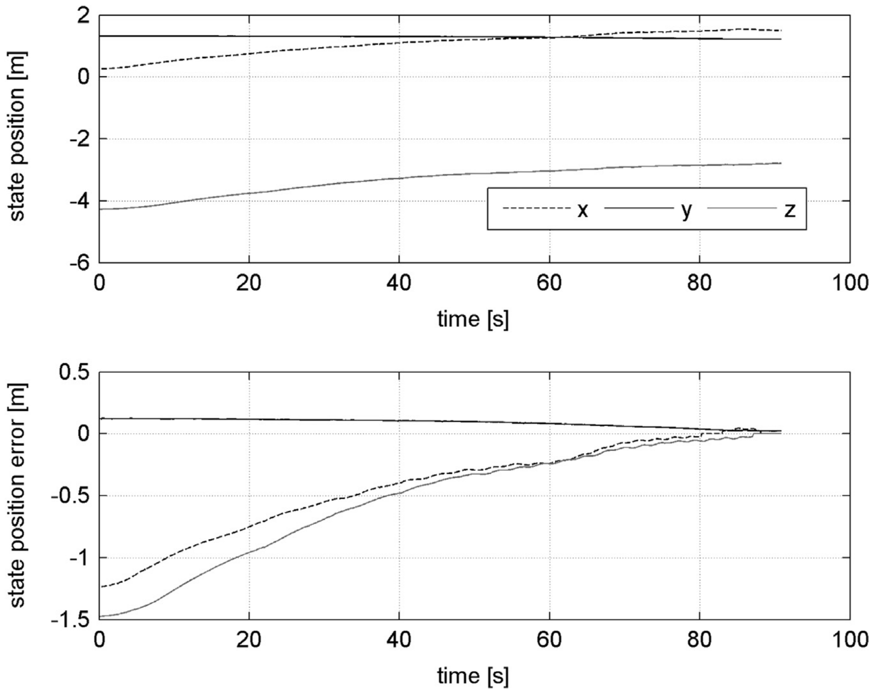


Fig. 20. Experimental results. Second test: position (top) and position error (down).

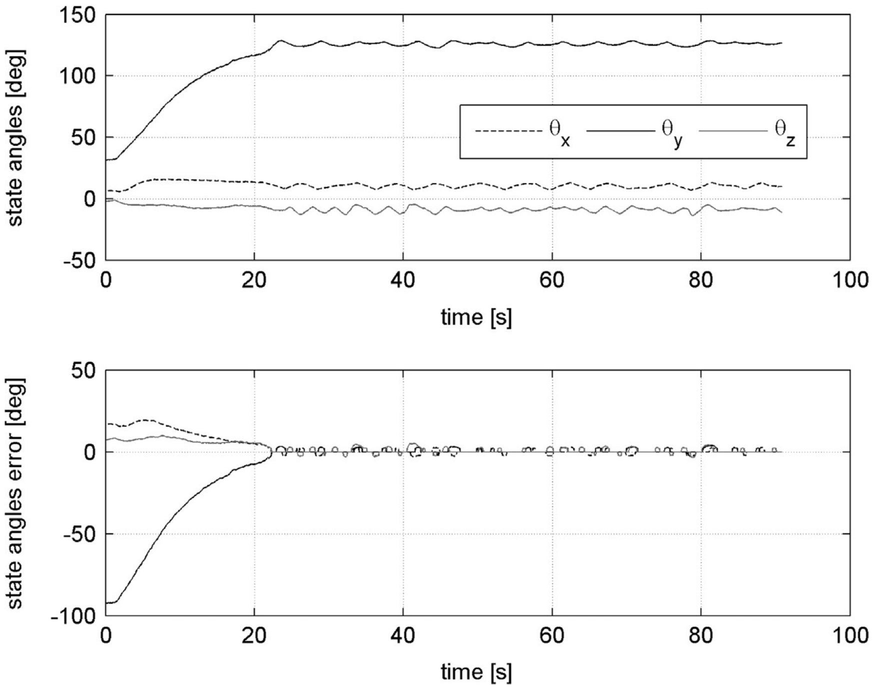


Fig. 21. Experimental results. Second test: Euler angles (top) and Euler angles error (bottom).

requiring much greater thrust values, which tend to generate oscillations or overshoot in the rotational components. It should be emphasized, however, that these irregularities never produced instability and that, as a result of tuning techniques and gain adjustments, these effects have been strongly reduced obtaining satisfactory results.

6. Conclusion

This work presented a real time, LQR-based sub-optimal approach for full six-degree-of-freedom spacecraft control. Previous works proved the effectiveness and reliability of linear quadratic optimal control, while this research has

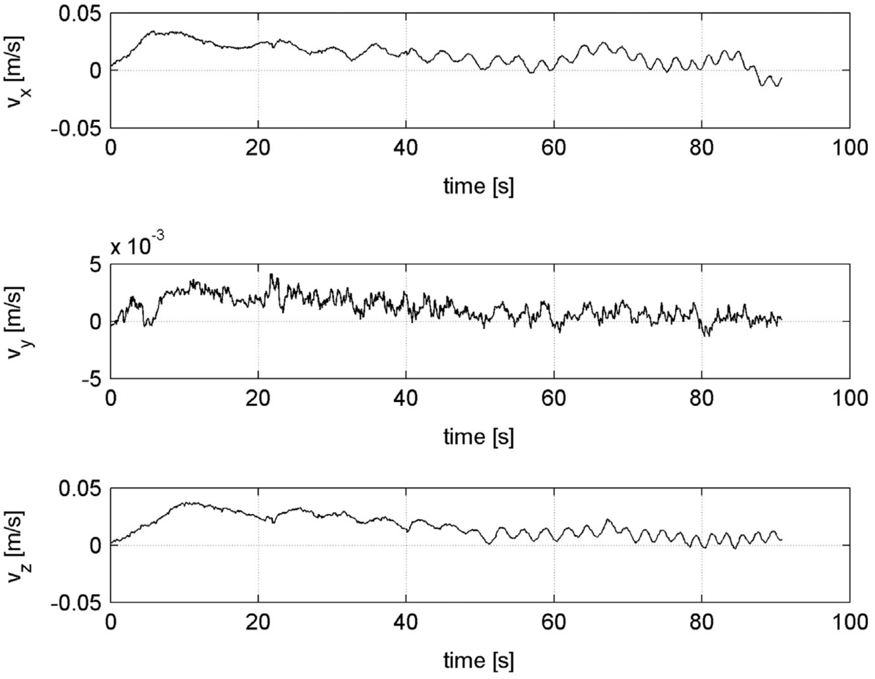


Fig. 22. Experimental results. Second test: linear velocity.

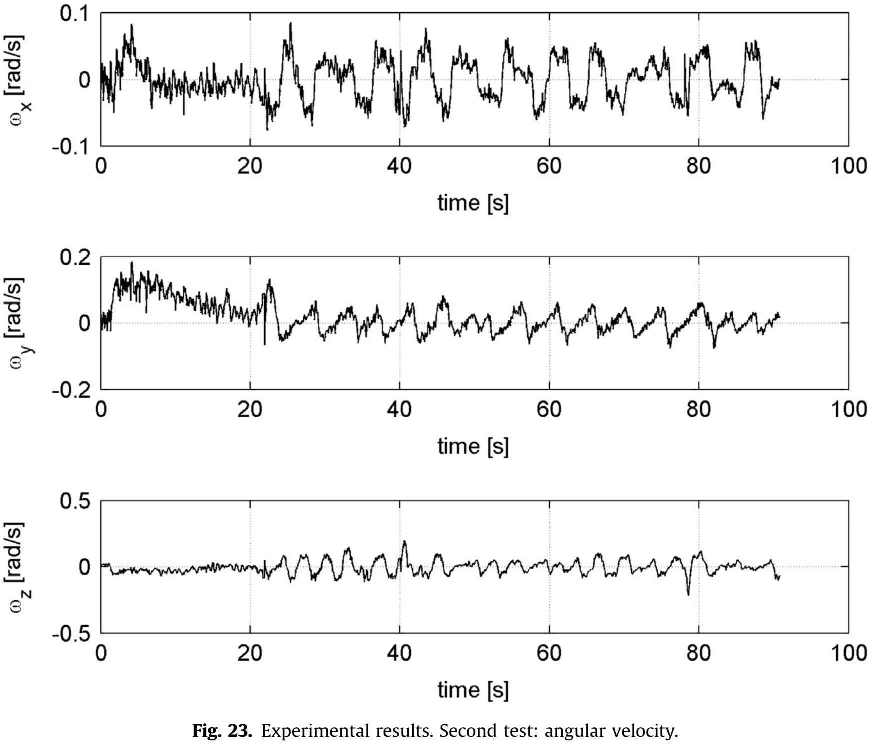


Fig. 23. Experimental results. Second test: angular velocity.

illustrated a way to quantify the trade-off between real-time feasibility and optimality. This consisted of a direct comparison with the general purpose optimal control software GPOPS-II. A deepened cost analysis has shown a discrepancy of 15%, although the sub-optimal solution is characterized by easier real-time implementability, reduced computational burden and hence lower simulation time. The proposed control approach has been tested on the ADAMUS test bed with a series of hardware-in-the-loop experiments that constituted the final validation of the

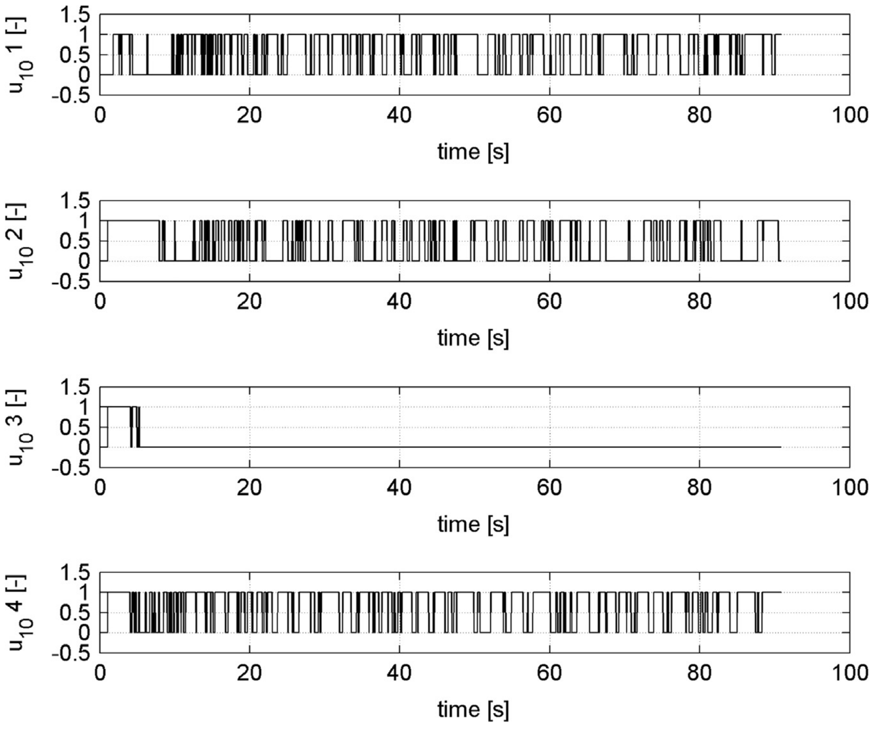


Fig. 24. Experimental results. Second test: thrusters configuration 1–4.

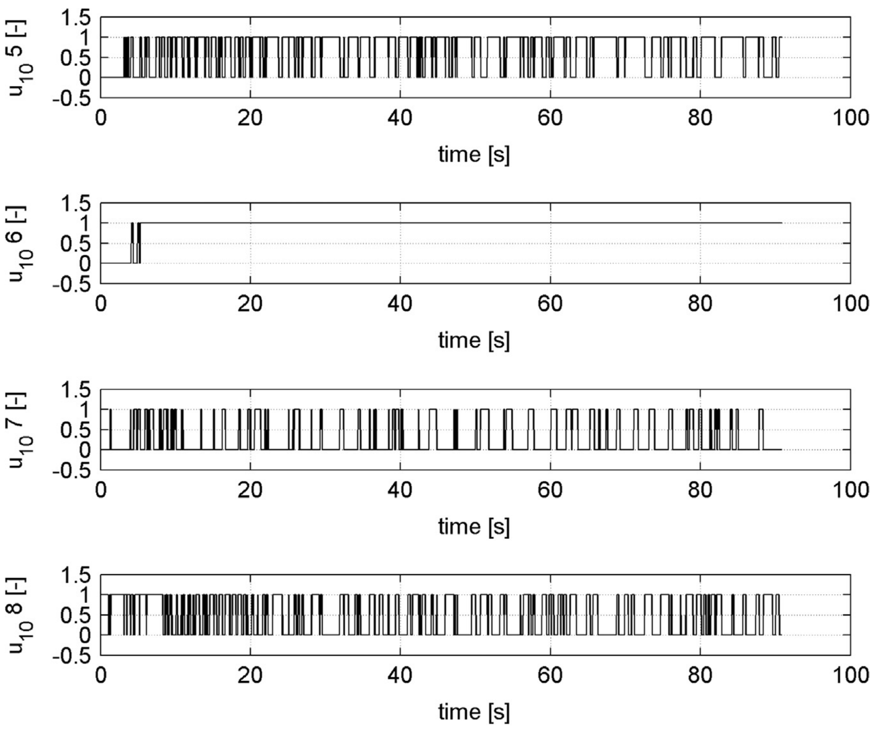


Fig. 25. Experimental results. Second test: thrusters configuration 5–8.

present research: these experiments, in fact, have effectively shown the sub-optimal six-degree-of-freedom control capability.

Both the error and the actual state response have been analyzed, together with the thrusters profile, for a complete study of the controller performances.

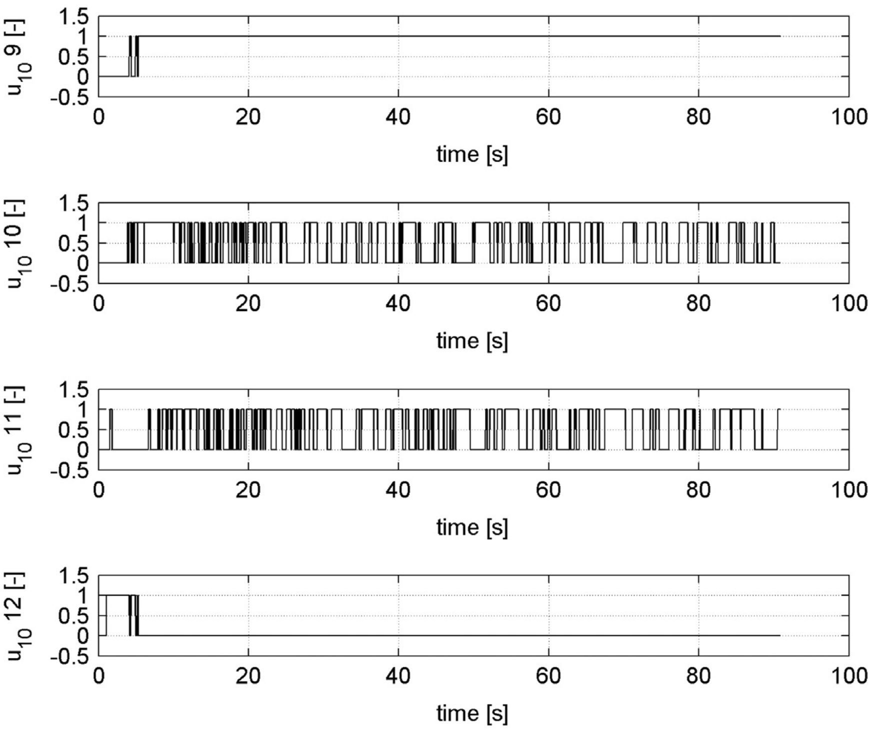


Fig. 26. Experimental results. Second test: thrusters configuration 9–12.

The results showed that both position and attitude reach the desired values, although some differences occur: the Euler angles plots, in fact, lack of the same regularity as the translational components, presenting occasional oscillations or slight overshoots. This imbalance directly derives from the existing differences between the translational dynamics and the rotational one, the former being much slower than the latter, and represents the only limit of the proposed controller, which is the main subject of the present research.

Acknowledgments

The authors wish to thank the U.S. Defense Advanced Research Projects Agency (DARPA) for supporting this research (AFRL contract no. FA9453-12-M-0275), and Dr. Thomas Alan Lovell, the contract manager, for his extensive editorial comments on this paper.

References

[1] David J. Irvin Jr., Richard G. Cobb, T. Alan Lovell, Fuel-optimal maneuvers for constrained relative satellite orbits, J. Guid. Control Dyn. 32 (May–June (3)) (2009) 960–973, http://dx.doi.org/10.2514/1.36618.
[2] R. Bevilacqua, M. Romano, Fuel-optimal spacecraft rendezvous with hybrid on–off continuous and impulsive thrust, J. Guid. Control Dyn. 30 (July–August (4)) (2007) 1175–1178, http://dx.doi.org/10.2514/1.27716.
[3] Zhanhua Ma, Ou Ma, Banavara N. Shashikanth, Optimal control for spacecraft to rendezvous with a tumbling satellite in a close range, in: Proceedings of the 2006 IEEE/RSJ, International Conference on Intelligent Robots and Systems, October 9–15, Beijing China, 2006, pp. 4109–4114, http://dx.doi.org/10.1109/IROS.2006.281877.
[4] Zhanhua Ma, Ou Ma, Banavara N. Shashikanth, Optimal approach to and alignment with a rotating rigid body for capture, J. Astronaut. Sci. 55 (October–December (1)) (2007) 407–419, http://dx.doi.org/10.1007/BF03256532.
[5] George. Boyarko, Oleg. Yakimenko, .Romano Marcello, Optimal rendezvous trajectories of a controlled spacecraft and a tumbling object, J. Guid. Control Dyn. 34 (July–August (4)) (2011) 1239–1252, http://dx.doi.org/10.2514/1.47645.
[6] H.J. Oberle, W. Grimm, BNDSCO-a program for the numerical solution of optimal control problems, Institute for Flight Systems Dynamics, DLR, Oberpfaffenhofen, 1989.
[7] I.M. Ross, F. Fahroo, User's manual for DIDO: a matlab package for dynamic optimization, Naval Postgraduate School Technical Report, Department of Aeronautics and Astronautics, 2002.
[8] P. Rutquist, M.M. Edvall, PROPT – MATLAB Optimal Control Software, 1260 S.E. Bishop Blvd Ste E, Tomlab Optimization Inc, Pullman, WA 99163, USA.
[9] R. Bevilacqua, O. Yakimenko, M. Romano, On-line generation of quasi-optimal spacecraft rendezvous trajectories, Acta Astronaut. (2009) 345–358, http://dx.doi.org/10.1016/j.actaastro.2008.08.001.
[10] Y. Yang, Analytic LQR design for spacecraft control system based on quaternion model, J. Aerosp. Eng. 25 (3 July (3)) (2012) 448–453, http://dx.doi.org/10.1061/(ASCE)AS.1943-5525.0000142.
[11] S. Beatty, Comparison of PD and LQR methods for spacecraft attitude control using star trackers, in: Conference Publication, Automation Congress, 24–26 July 2006, pp. 1–6, http://dx.doi.org/10.1109/WAC.2006.375957.
[12] L. Walker, D. Spencer, Automated proximity operations using image-based relative navigation, in: 26th Annual USU/AIAA Conference on Small Satellites, August 2012.
[13] S.B. McCamish, X. Yun, M. Romano, Distributed autonomous control of multiple spacecraft during close proximity operations (Dissertation), Naval Postgraduate School, Monterey, California, December 2007.
[14] R. Bevilacqua, T. Lehmann, M. Romano, Development and experimentation of LQR/APF guidance and control for autonomous proximity maneuvers of multiple spacecraft, Acta Astronaut. 68 (2011) 1260–1275, http://dx.doi.org/10.1016/j.actaastro.2010.08.012.
[15] R. Bevilacqua, J.S. Hall, J. Horning, M. Romano, Ad hoc wireless networking and shared computation for autonomous multirobot systems, AIAA J. Aerosp. Comput. Inf. Commun. 6 (May (5)) (2009) 328–353, http://dx.doi.org/10.2514/1.40734.
[16] R. Bevilacqua, M. Romano, F. Curti, A.P. Caprari, V. Pellegrini, Guidance navigation and control for autonomous multiple spacecraft

assembly: analysis and experimentation, Int. J. Aerosp. Eng. (2011) 18, Article ID 308245, http://dx.doi.org/10.1155/2011/308245.
[17] Advanced Autonomous Multiple Spacecraft Laboratory, Dr. R. Bevilacqua, ⟨http://www.riccardobevilacqua.com/adamuslab.html⟩, (accessed 19.01.15).
[18] K. Saulnier, D. Pérez, R. Huang, D. Gallardo, G. Tilton, R. Bevilacqua, A six-degree-of-freedom hardware-in-the-loop simulator for small spacecraft, Acta Astronaut. (2014) 444–462, http://dx.doi.org/10.1016/j.actaastro.2014.10.027.
[19] Z. Qu, J.R. Cloutier, C.P. Mracek, A new sub-optimal nonlinear control design technique-SDARE, in: Proceedings of the 13th IFAC World Congress, San Francisco, USA, 1996, pp. 365–370.
[20] K.D. Hammett, Control of nonlinear systems via state feedback state-dependent Riccati equation techniques (Ph.D. thesis), Graduate School of Engineering, Department of Aeronautics and Astronautics, U.S. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, June 1997.
[21] K.D. Hammett, C.D. Hall, D.B. Ridgely, Controllability issues in non-linear state-dependent Riccati equation control, J. Guid. Control Dyn. 21 (September–October (5)) .
[22] T. Çimen, Survey of state-dependent Riccati equation in nonlinear optimal feedback control synthesis, AIAA J. Guid. Control Dyn. 35 (4) (2012) 1025–1047.
[23] Bong Wie, Space Vehicle Dynamics and Control, second ed. American Institute of Aeronautics and Astronautics, Reston, VA, 2008 Chapters 4.6 (pp. 299–303), 5.1–5.3 (pp. 323–334), 5.5.2 (pp. 341–343), 6.1–6.4 (pp. 349–362).
[24] F. Curti, M. Romano, R. Bevilacqua, Lyapunov-based thrusters' selection for spacecraft control: analysis and experimentation, J. Guid. Control Dyn. 33 (July–August (4)) (2010) 1143–1160, http://dx.doi.org/10.2514/1.47296.
[25] I.M. Ross, Control and Optimization: An Introduction to Principles and Applications, Electronic Edition, Naval Postgraduate School, Monterey, CA, December 2005.
[26] H. Kwakernaak, R. Sivan, Linear Optimal Control Systems, Wiley-Interscience, US, 1972.
[27] J.N. Little, Clay M. Thompson, Linear Quadratic Regulator Design, Source code, CopyrightÂ 1986–1993 by the MathWorks, Inc.
[28] GPOPS-II: Next-Generation Optimal Control Software, ⟨http://www.gpops.org/⟩, (accessed 19.01.15).
[29] D. Garg, M. Patterson, W.W. Hager, A.V. Rao, D.A. Benson, G.T. Huntington, A unified framework for the numerical solution of optimal control problems using pseudospectral methods, Automatica 46 (November (11)) (2010) 1843–1851, http://dx.doi.org/10.1016/j.automatica.2010.06.048.
[30] D. Garg, M.A. Patterson, C.L. Darby, C. Francolin, G.T. Huntington, W. W. Hager, A.V. Rao, Direct trajectory optimization and costate estimation of general optimal control problems using a Radau pseudospectral method, Comput. Optim. Appl. 49 (June (2)) (2011) 335–358, http://dx.doi.org/10.1007/s10589-009-9291-0.
[31] M.A. Patterson, A.V. Rao, GPOPS-II Version 1.0: a general-purpose MATLAB toolbox for solving optimal control problems using the Radau pseudospectral method, User's manual, University of Florida, Gainesville, USA, January 2013.
[32] D. Gallardo, R. Bevilacqua, R.E. Rasmussen, Advances on a 6 degrees of freedom testbed for autonomous satellites operations, in: AIAA Guidance, Navigation, and Control Conference, Portland, Oregon, 08–11 August 2011, http://dx.doi.org/10.2514/6.2011-6591.
[33] K. Saulnier, D. Perez, G. Tilton, D. Gallardo, C. Shake, R. Huang, R. Bevilacqua, Operational capabilities of a six degrees of freedom spacecraft simulator, in: AIAA GNC Conference 2013, Boston, MA, (AIAA 2013–5253), http://dx.doi.org/10.2514/6.2013-5253.
[34] J. Jung, S. Park, S. Kim, Y.H. Eun, Y. Chang, Hardware-in-the-loop simulations of spacecraft attitude synchronization using state-dependent Riccati equation technique, Adv. Space Res. 51 (February (3)) (2013) 434–449, http://dx.doi.org/10.1016/j.asr.2012.09.004.
[35] W.R. Wilson, M.A. Peck, An air-levitated testbed for flux pinning interactions at the nanosatellite scale, in: AIAA Guidance, Navigation, and Control Conference, Ontario Canada, August 2010, pp. 1088–1097.
[36] F.D. Roe, D.W. Mitchell, B.M. Linner, D.L. Kelley, Simulation techniques for avionics systems-an introduction to a world class facility, in: Proceedings of the AIAA Flight Simulation Technologies Conference, AIAA, Washington, DC, 1996, pp. 535–543, http://dx.doi.org/10.2514/6.1996-3535.
[37] D. Cho, D. Jung, P. Tsiotras, A 5-DoF experimental platform for spacecraft rendezvous and docking, in: Infotech at Aerospace Conference, AIAA, Washington, DC, 2009, pp. 730–749.
[38] X. Jian, B. Gang, Y. QinJun, L. Jun, Design and development of a 5-DoF air-bearing spacecraft simulator, in: Proceedings of the International Asia Conference on Informatics and Control, Automation, and Robotics, February 2009, pp. 126–130, http://dx.doi.org/10.1109/CAR.2009.7.
[39] A. Ledebuhr, L. Ng, M. Jones, B. Wilson, R. Gaughan, E. Breitfeller, W. Taylor, J. Robinson, D.R. Antelman, D. Neilsen, Micro-satellite ground test vehicle for proximity and docking operations development, in: Proceedings of the IEEE Aerospace Conference, vol. 5, IEEE, Piscataway, NJ, 2001, pp. 2493–2504, http://dx.doi.org/10.1109/AERO.2001.931210.
[40] S.P. Viswanathan, A. Sanyal, L. Holguin, Dynamics and control o f a six degrees of freedom ground simulator for autonomous rendezvous and proximity operation of spacecraft, in: Proceedings of AIAA Guidance, Navigation, and Control Conference, Minneapolis, Minnesota, 2012, pp. 4608–4626, http://dx.doi.org/10.2514/6.2012-4926.
[41] P. Zarchan, H. Musoff, F.K. Lu, Fundamental of Kalman Filtering: A Practical Approach, AIAA, 2005, 129–291 Manufacturer Number: 1563476940, Reston, VA, Chapters 4–7.
[42] R.E. Kalman, A new approach to linear filtering and prediction problems, ASME J. Basic Eng. (1960) 35–45, http://dx.doi.org/10.1115/1.3662552.

Leone Guarnaccia received the B.S. in mechanical engineering (2011) from the University of Catania, Italy and the M.S. in mechanical engineering (2013) from the Polytechnic University of Turin, Italy.

He was a visiting scholar at the ADAMUS laboratory from March to September 2013: he worked on Guidance, Navigation, and Control of a spacecraft simulator.

His research interests include robotics and automation, machine design, thermodynamics and fluid mechanics.

Riccardo Bevilacqua holds a M.S. in aerospace Engineering (2002), and a Ph.D. in applied mathematics (2007), both earned at the University of Rome, "Sapienza", Italy. He worked as a project engineer in mission analysis at Grupo Mecanica del Vuelo, in Madrid, Spain, during 2003. He was a US National Research Council Post-Doctoral Fellow from 2007 to 2010, and an Assistant Professor at Rensselaer Polytechnic Institute from 2010 to 2014. He is presently an Associate Professor of the Mechanical and Aerospace Engineering Department, at the University of Florida.

His research interests focus on Guidance, Navigation, and Control of multiple spacecraft systems and multiple robot systems.

Stefano P. Pastorelli received the M.S. degree in mechanical engineering and the Ph.D. degree in applied mechanics from Politecnico di Torino, Turin, Italy, in 1992 and 1996, respectively.

From 1995 to 1997 he was a Consultant Engineer for a high-tech aeronautical company. He is currently an Associate Professor of Applied Mechanics and Robotics with the Department of Mechanical and Aerospace Engineering at the Politecnico di Torino, Italy.

His research interests include functional design of mechanical systems, mechanical power transmission and actuation servosystems, kinematics and dynamics of robots, 3-D motion capture techniques and identification, active human modelling for biomechanics.