JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 39, No. 2, February 2016

Particle Swarm Optimization for Time-Optimal Spacecraft Reorientation with Keep-Out Cones

Dario Spiller*
University of Rome "La Sapienza", 00184 Rome, Italy
Luigi Ansalone†
Agenzia Spaziale Italiana, 00133 Rome, Italy
and
Fabio Curti‡
University of Rome "La Sapienza", 00138 Rome, Italy

DOI: 10.2514/1.G001228

This paper deals with the problem of spacecraft time-optimal reorientation maneuvers under boundaries and path constraints. The minimum time solution with keep-out constraints is proposed using the particle swarm optimization technique. A novel method based on the evolution of the kinematics and the successive obtainment of the control law is presented and named as inverse dynamics particle swarm optimization. It is established that the computation of the minimum time maneuver with the proposed technique leads to near-optimal solutions, which fully satisfy all the boundaries and path constraints.

Nomenclature

C_x	=	keep-out constraint of the x light source
c, k, l	=	user-defined time constants
c_p, c_l, c_g	=	user-defined multiplying constant in the velocity equation
G_j	=	jth penalty function for the inequality constraint
g^k_{best}	=	global best "position" of the entire swarm at the evolution instant k
g_j	=	jth inequality constraint
H_i	=	ith penalty function for the equality constraint
h_i	=	ith equality constraint
I	=	inertia tensor, kg · m^2
i^*	=	cycle index after which the keep-out constraint is completely considered
J	=	performance index
$l^k_{\text{best},i}$	=	local best position of the ith particle at the evolution instant k
$\boldsymbol{M}$	=	external torque, N · m
$M_{\max}$	=	maximum external torque provided by the actuators, N · m
N_{eq}	=	number of equality constraints
N_{ineq}	=	number of inequality constraints
N_{int}	=	number of iterations of the internal loop
N_{viol}	=	number of violated constraints
$\boldsymbol{p}$	=	vector of the modified Rodriguez parameters
$p^k_{\text{best},i}$	=	personal best position of the ith particle at the evolution instant k
R	=	rotation matrix
r	=	user-defined scale factor
t	=	generic time instant, s
$t_f^{(i)}$	=	final maneuver time of the ith particle
$U_j^{(i)}(k)$	=	kth control interpolation point along the jth axis of the ith particle
u	=	uniformly distributed random number between 0 and 1
v_i^{k+1}	=	"velocity" of the ith particle leading x_i^k to x_i^{k+1}
$v_{\min}$, $v_{\max}$	=	velocity boundaries for the particles
x_i^k	=	position of the ith particle at the evolution instant k
$x_{\min}$, $x_{\max}$	=	position boundaries for the particles
α_s	=	minimum angular distance from the generic light source, rad
Δ_{eq}	=	tolerance on the equality constraints
Δ_{ineq}	=	tolerance on the inequality constraints
$\zeta_j^{(i)}(k)$	=	kth interpolation point for the jth attitude parameter of the ith particle
$\boldsymbol{\eta}$	=	vector of quaternions
Θ_f	=	imposed maneuver angle, rad
θ	=	angle of rotation, rad
Ξ	=	kinematic matrix for the quaternions
Ψ	=	kinematic matrix for the modified Rodriguez parameters
$\boldsymbol{\sigma}$	=	sensor axis vector
$\boldsymbol{\sigma}_s$	=	generic light source axis vector
ϕ	=	fitness function
$\boldsymbol{\omega}$	=	angular velocity in the body fixed frame, rad/s

I. Introduction

TIME-OPTIMAL spacecraft reorientation maneuvers under boundaries and path constraints are difficult to compute because the solutions are related to complex nonlinear problems. The minimum time reorientation maneuver of a rigid spacecraft is a well-known problem; the first related work regarding a numerical approach dates back to the 1990s [1–4]. These papers have been used as a reference providing some test cases for successive works as in [5–7]; the research still focuses on this problem with several approaches, for example, through homotopic approach algorithms [8], pseudospectral optimization analysis [9,10], or with hybrid numerical techniques [11].

The problem of reorientation maneuvers with path constraints has been initially studied by McInnes in [12]. The path constraints are determined by bright sources such as the sun or the moon that need to be avoided by payloads such as telescopes or sensors as with star trackers [13]. The path planning has to take into account exclusion cones with an angle greater than the effective angular diameter of the bright source, due to its very low apparent magnitude. Other path

Received 22 December 2014; revision received 21 April 2015; accepted for publication 13 August 2015; published online 22 September 2015.

*Ph.D. Student, DIMA, Via Eudossiana 18; dario.spiller@uniroma1.it.
†Postdoctoral Fellow, Technology Division, Via del Politecnico snc; luigi.ansalone@est.asi.it.
‡Associate Professor, School of Aerospace Engineering, Laboratory of Automation, Robotics and Control for Aerospace, Via Salaria 851; fabio.curti@uniroma1.it.

constraints could include pointing boundaries for antennas to maintain communication [14]. The angle of exclusion is determined by the sensor baffle and the source (considering its angular dimension and its intensity), and it is typically between 15 and 45 deg, depending on the sensitive optical sensor [15–17]. Several numerical methods, such as the randomized motion planning [18], the logarithmic barrier potentials [19], or the Lie group variational integrator [20], have been used to obtain the optimal solution.

Heuristic and metaheuristic algorithms have been recently proposed for the planning of slew maneuvers, as in [21,22]. Notice that, in this work, we adopt the terminology introduced in [23], making a distinction between heuristic and metaheuristic algorithms. Heuristics refers to methods based on finding the solution by trial and error. Metaheuristics refers to higher-level heuristics, where a tradeoff between randomization and local search is introduced, with the latter usually based on nature-inspired models. Metaheuristic algorithms may be used to find suboptimal solutions; for instance, Melton proposed a hybrid technique where a metaheuristic solution was used as the best available initial guess for a pseudospectral optimizer [24]. The hybrid technique has been newly proposed in [25], where the initial guess is carried out with the covariance matrix adaptation-evolutionary strategy. The metaheuristic algorithms are being studied extensively, and the high interest generated from their results is shown in the research performed by NASA [26]. With regard to the particle swarm optimization (PSO), which is the primary interest of this paper, it has been used not only for the planning of attitude maneuvers as in [27,28] but also for trajectory planning [29–31] or for attitude determination [32,33].

From the previous works, the optimization of attitude maneuvers uses the PSO applied to the control. This approach here is referred to as the direct method. The paper shows how such an approach fails in satisfying the boundary constraints (i.e., final position and final velocity). In this work, a new approach is then proposed, where the PSO technique is applied to the kinematics rather than to the control. As a result, the final boundary constraints are straight satisfied. This approach is referred to as the inverse method.

The paper is organized as follows. Section II reviews the PSO method. Section III shows the scenario in which the slew maneuver has to be accomplished and the optimization approach with the direct method. Section IV introduces the inverse method. Section V shows the numerical results and the comparison between the direct method and the inverse method. Section VI concludes the paper.

II. Particle Swarm Optimization Method

The PSO is a population-based algorithm introduced by Kennedy and Eberhart in 1995 [34]. This method is based on cooperation between solutions, partially random and without selection (i.e., with fixed-size population [35]).

The most important features of the original algorithm may be found in [36]; this method presumes the evolution of a group of candidate solutions called particles that move through the feasible research space (FRS), which indicates the set of all acceptable and meaningful solutions. During the evolution, each particle is evaluated according to a numerical value determined through a fitness function ϕ, which takes into account the goal of the optimization and the constraints. The aim is to find the minimum value of the fitness function.

The generic particle represents a possible optimal solution inside the FRS. The position of the generic particle $x_i^{(k)}$ is defined as the solution associated with the ith particle at the kth step of its evolution. The particle is perturbed by a term called velocity in each step. With reference to Fig. 1, the displacement can be simply evaluated as

$$x_i^{(k+1)} = x_i^{(k)} + v_i^{(k+1)} \tag{1}$$

The actual position $x_i^{(k)}$ is updated through the term $v_i^{(k+1)}$ reaching the new position $x_i^{(k+1)}$. Note that the term v_i is not a velocity in the traditional physical sense (i.e., $v_i \neq \mathrm{d}x_i/\mathrm{d}t$), but the velocity is the rate at which the position per generation changes.

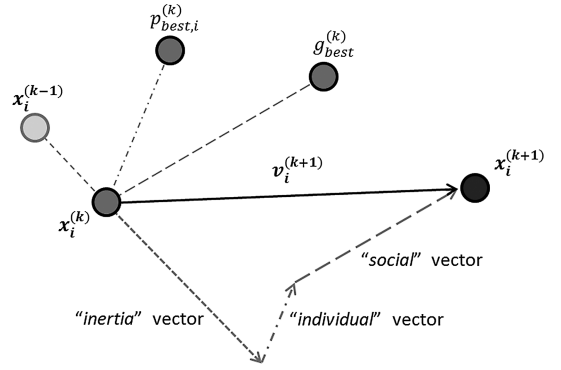

Fig. 1 Representation of the displacement of a particle with respect to the different inputs.

In each step k, every ith particle is assigned a performance index $J_i^{(k)}$ evaluated through the fitness function $\phi_i^{(k)}$. $J_i^{(k)}$ is related to the single best position covered by the ith particle until the step k; this position is referred to as personal best of the ith particle $p_{\mathrm{best},i}^{(k)}$. In particular,

$$\text{if } \phi_i^{(k)} < J_i^{(k)} \Rightarrow J_i^{(k)} = \phi_i^{(k)} \Rightarrow p_{\mathrm{best},i}^{(k)} = x_i^{(k)} \tag{2}$$

The global best particle $g_{\mathrm{best}}^{(k)}$ is the particle that has had the best value of the performance index $J_g^{(k)}$ until step k. The global best is updated only when the personal best of a particle is considered to be better than the previous best position of the swarm:

$$\text{if } J_i^{(k)} < J_g^{(k)} \Rightarrow J_g^{(k)} = J_i^{(k)} \Rightarrow g_{\mathrm{best}}^{(k)} = p_{\mathrm{best},i}^{(k)} \tag{3}$$

These data are used to dictate the direction of the evolution for the entire swarm through the velocity term. With respect to what is reported in Fig. 1, the displacement of each particle is the sum of three different velocity vectors. Every time the position of the group is updated, the generic particle moves partly toward its previous best position $p_{\mathrm{best},i}^{(k)}$ (through the "individual" vector, also known as "cognitive" vector) and partly toward the particle with the best fitness value within the group $g_{\mathrm{best},i}^{(k)}$ (through the "social" vector). Although the direction of these vectors is determined by the history of the evolution, their magnitude is multiplied by a stochastic value uniformly distributed between 0 and 1 to give some randomness to the search of the optimization goal. A further contribution to the displacement is given by the inertial velocity ("inertia" vector), which makes the ith particle move in the direction in which the actual position has been reached. This means that the inertia vector is along the direction that goes from the positions $x^{(k-1)}$ to the position $x^{(k)}$ of the particle. Through the contributions of the inertia, individual, and the social vectors, the ith particle moves from the k position (meaning the position at time k) to the $k+1$ position through the velocity vector $v_i^{(k+1)}$.

To avoid the risk of stopping at the local minimum, a unified particle swarm optimization (UPSO) [37] has been used. When the global best particle is used to lead the search, as previously explained, the method is denoted as a global version. However, a local version [38] may also be used where the ith particle compares its fitness value only in a small neighborhood. In this neighborhood, a local best particle $l_{\mathrm{best},i}^{(k)}$ with performance index $J_{l,i}^{(k)}$ is defined as

$$\text{if } J_i^{(k)} < J_{l,i}^{(k)} \Rightarrow J_{l,i}^{(k)} = J_i^{(k)} \Rightarrow l_{\mathrm{best},i}^{(k)} = p_{\mathrm{best},i}^{(k)} \tag{4}$$

In this manner, the whole search group tends to separate into subgroups, thus enhancing the probability of finding the optimal solution when the problem has more than one suboptimal solution. The UPSO strategy combines the advantages of the global and local

versions [37]. A simple way to deal with UPSO is that, in the first part of the loop, the local best is considered to be more important than the global best. Hence, the swarm may compare different local minima (if various local minima are found) in the first part of the evolution. In the final part of the evolution, the global version is now privileged, consequently allowing the swarm to converge quickly to the global minimum.

As far as the velocity of the particles is concerned, it is described by the following formulation:

$$v_i^{(k+1)} = r(w \cdot v_i^{(k)} + u_1 \cdot c_p(p_{\text{best},i}^{(k)} - x_i^{(k)}) + \\ + u_2 \cdot c_l(l_{\text{best},i}^{(k)} - x_i^{(k)}) + u_3 \cdot c_g(g_{\text{best}}^{(k)} - x_i^{(k)})) \tag{5}$$

where u_1, u_2, and u_3 are different random numbers with uniform distribution between 0 and 1. The velocity considers four inputs.

1) $v_i^{(k)}$ is the inertia term multiplied by a constant w, the inertia weight [39]. This term makes the particle move in the direction through which it reached the actual position. In the proposed algorithm, the inertia weight is set to be linearly decreasing during the loop. In this manner, the search will be more random at the beginning of the evolution, whereas it will converge around the best particle at the end.

2) $p_{\text{best},i}^{(k)}$ is the personal best; during the evolution, each particle remembers its previous personal best position $p_{\text{best},i}^{(k)}$ and always tends to return to that position.

3) The other two terms are the social terms; $l_{\text{best},i}^{(k)}$ is called the local best particle and represents the particle with the best fitness value in a small neighborhood of the ith particle. There is no need to introduce a sorting of the population because the aim of the social term is to divide the whole population into different searching subgroups. As a consequence, the neighborhood of the ith in this work includes the particles $i \pm j$: $j = 1, 2, 3$. The dimension of the neighborhood is a parameter of the algorithm that can influence the results. On the other hand, $g_{\text{best}}^{(k)}$ is the global best position (i.e., the position of the swarm with the best fitness value evaluated until the current phase of the evolution).

The magnitude of the last three quantities depends on the distance of $p_{\text{best},i}^{(k)}$, $l_{\text{best}}^{(k)}$, and $g_{\text{best}}^{(k)}$ from the actual position $x_i^{(k)}$. However, these terms are multiplied by user-defined constants (c_p, c_l, and c_g respectively) and by the random number u_i, which produces an amount of randomness to the displacement. The term r may be introduced as a scale factor: it may be constant or decreasing; however, its use is recommended only when the user has a good insight of the problem.

Another important feature related to Eqs. (1) and (5) is that the magnitude of velocity and displacement must be limited to make the particle move only within the FRS. In this sense, the following boundaries are introduced:

$$v_{\min} < v < v_{\max}, \qquad x_{\min} < x < x_{\max} \tag{6}$$

where the subscript i has been removed for sake of simplicity. The maximum and minimum values of the velocity and the displacement are defined by the user; usually the maximum value of the velocity is set at about 10–20% of the dynamic range of the variable [40]. High velocities lead to a quick convergence with little exploration of the search space, whereas small velocities lead to a slow convergence, usually with high exploration of the search space. The evolution of particles continues until a convergence criterion based on a user-defined tolerance is satisfied.

Convergence of the swarm toward a stable position with velocity tending to zero is demonstrated [35]. The time required for convergence is a function of the parameters in Eq. (5), and nothing guarantees that the point of convergence is the sought optimum. In our case, however, we will illustrate that the swarm meets in the neighborhood of the global optimum.

III. Problem Statement

This study deals with the problem of a slew maneuver. A satellite must perform a reorientation maneuver where some optical instruments (for example a star tracker) cannot be exposed to sources of bright light such as the Earth, the sun, and the moon. The maneuver angle Θ_f and the initial and final attitudes are known. The slewing motion must be constrained to prevent the sensor axis from entering into established "keep-out" zones known as cones. Such areas have central axes pointing to the sun, the Earth, and the moon, and specified half-angles depending on the light magnitude, the distance from the satellite, and the angular diameter of the source. Moreover, the maneuver must be rest-to-rest (i.e., the angular velocity must be equal to zero for $t = t_0$ and $t = t_f$). In particular, t_0 is a constant parameter (that may be set equal to zero).

The objective is to minimize the maneuver overall time $t_f - t_0$, and so the performance index is

$$J = t_f - t_0 \tag{7}$$

A definition for the used reference frames is needed to describe the maneuver.

A definition for the used reference frames is needed to describe the maneuver. One important hypothesis is that the time for completing the maneuver is negligible with respect to the time for completing an orbit. In this case, with regard to an Earth-centered inertial reference frame, it is possible to approximate the velocity of the satellite center of mass to zero. This approximation allows the definition of an inertial reference frame fixed in the original position of the body-fixed frame at time $t = t_0$ that will be referred to as BRF_0. As a consequence, the positions of the keep-out cones defined in this inertial frame do not change during the maneuver. The angular velocity is defined in the body-fixed frame, and it is denoted as $\boldsymbol{\omega} = [\omega_x \quad \omega_y \quad \omega_z]^T$.

The maneuver is a rotation around the initial body-fixed x axis. Obviously, if no keep-out zones are present in the space needed for the maneuver, the rotation takes place around the x-body axis. If, on the other hand, one or more keep-out cones are present, all the components of $\boldsymbol{\omega}$ are different from zero. The rigid-body motion is described by Euler's equation expressed in the most general form as follows:

$$I\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times I\boldsymbol{\omega} = \boldsymbol{M} \tag{8}$$

where I is the inertia tensor, and $\boldsymbol{M}$ is the total torque vector. In particular, we will consider three independent torques aligned with the axes of the body-fixed frame.

In the evolution between the initial and the final positions, the sensor axis must be kept at least at the minimum angular distance α_s from each light source. Using the notation from [21], we can define the so-called keep-out constraints as

$$C_s(t) = \boldsymbol{\sigma}(t) \cdot \boldsymbol{\sigma}_s - \cos(\alpha_s) \le 0 \quad \forall\, t \in [t_0, t_f] \tag{9}$$

where $\boldsymbol{\sigma}(t)$ is the direction pointed to by the optical sensor, and $\boldsymbol{\sigma}_s$ is the vector placed in the center of mass of the satellite and pointing to the generic source of light, here represented by s.

The constraint is shown in Fig. 2. On the left, a feasible configuration is reported; the sensor axis σ is outside the keep-out cone. On the right, an unfeasible configuration is reported; in this case, the sensor axis σ is inside the keep-out cone. Introducing the new variable β defined as $\beta(t) = \cos^{-1}(\boldsymbol{\sigma}(t) \cdot \boldsymbol{\sigma}_s)$, the constraint in Eq. (9) may be rewritten more easily as

$$\beta(t) \ge \alpha_s \quad \forall t \in [t_0, t_f] \tag{10}$$

The cone defined for each source of light will be referred to as keep-out cone. The first analysis has been made by directly integrating the control law [21]. For clarification, Eq. (8) is used in the following form:

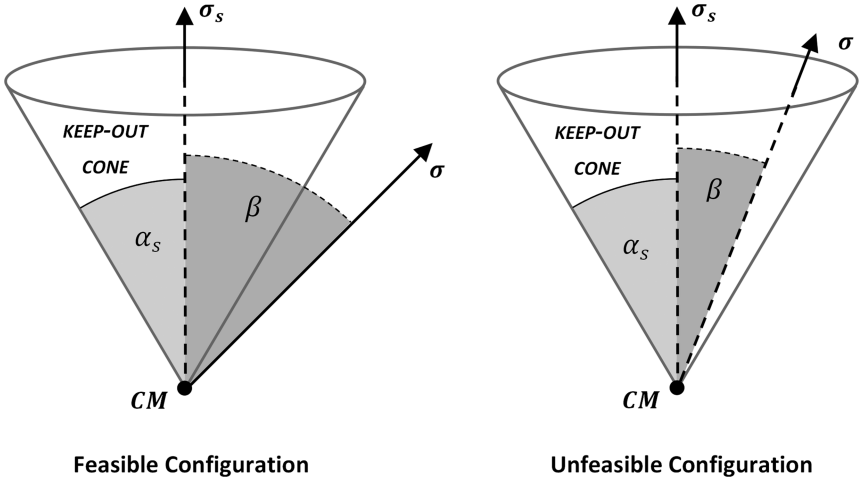

Fig. 2 Graphical representation of the keep-out cone constraint.

$$\dot{\boldsymbol{\omega}} = I^{-1}(\boldsymbol{M} - \boldsymbol{\omega} \times I\boldsymbol{\omega}) \tag{11}$$

In the following pages, the acronym DPSO (for "direct particle swarm optimization") will be used to refer to this technique, where "*d*" stands for direct integration of the dynamics.

Each particle of the swarm contains information about the control law in the body-axes and the final time for the maneuver. Each particle is an array containing 1) the maneuver time $t_f^{(i)}$, and 2) the values for the control law $U_j^{(i)}(k)$, where i identifies the particle, $j = 1, 2, 3$ is the maneuver axis, and $k = 1, 2, \ldots, m$, with being m the number of points that will be used for the interpolation of the control. The control $\boldsymbol{M}(t)$ is obtained in $n + 1$ integration points (so that $t = t_0, t_1, \ldots, t_n$ and $t_n = t_f^{(i)}$) interpolating the m points $U_j^{(i)}(k)$ with cubic splines. The m points are associated to time instants equally spaced between t_0 and $t_f^{(i)}$. If $\|\boldsymbol{M}(t_i)\| > M_{\max}$ for some $0 < i < n$, then $\|\boldsymbol{M}(t_i)\| > M_{\max}$.

The so-called swarm is an array of structure whose length is determined by the number of particles $n_{\text{particles}}$. $\Delta U_j^{(i)}(k)$ and $\Delta t_f^{(i)}$ are defined as the velocities associated with the control and the maneuver time of each particle.

With regard to the displacements constraints of the swarm, Eq. (6) takes the following form:

$$\begin{gathered} |\Delta U_j^{(i)}(k)| \le 0.2 \cdot M_{\max}, \qquad |U_j^{(i)}(k)| \le M_{\max} \\ |\Delta t_f^{(i)}| < 0.1 \cdot (t_{\max} - t_{\min}), \qquad t_{\min} < t_f^{(i)} < t_{\max} \\ i = 1, \ldots, n_{\text{particles}}, \qquad j = 1, \ldots, m \end{gathered} \tag{12}$$

$M_{\max}$ is the known maximum value of the actuators, whereas $t_{\max}$ and $t_{\min}$ may be defined by knowing the unconstrained solution. The values 0.1 and 0.2 are chosen to make the maximum velocities equal to the 10% of the dynamic range of the particles, as already explained in relation to Eq. (6).

The initialization of the time and the control of each particle are based on a uniform random distribution of the particles within the constraints of Eq. (12).

The attitude is described by quaternions of rotation $\bar{\boldsymbol{\eta}} = [\boldsymbol{\eta} \quad \eta_4]$. The description of the quaternions along with the description of the associated rotation matrix $R(\bar{\boldsymbol{\eta}})$ and the kinematic matrix $\Xi(\bar{\boldsymbol{\eta}})$ [41]. For completeness, the kinematic equation for the quaternions is

$$\dot{\bar{\boldsymbol{\eta}}} = \frac{1}{2}\Xi(\bar{\boldsymbol{\eta}})\boldsymbol{\omega} \tag{13}$$

The position of the rotated sensor axis $\boldsymbol{\sigma}(t)$ at some time instant t in the initial inertial reference frame is obtained rotating the initial position of sensor axis $\boldsymbol{\sigma}(t_0)$:

$$\boldsymbol{\sigma}(t) = R(\bar{\boldsymbol{\eta}})^T \boldsymbol{\sigma}(t_0) \tag{14}$$

Once the time records for the angular displacement are produced, the angular velocity and the position of the sensor axis, the boundary, and path constraints must be evaluated. Summarizing the details of the problem thus far outlined, the optimization of the constrained slew maneuver with the direct approach has the following mathematical form:

$$\begin{aligned} &\text{Find: } \min t_f \\ &\text{subjected to} \\ &\text{dynamic constraints: } \dot{\boldsymbol{\omega}} = I^{-1}(\boldsymbol{M} - \boldsymbol{\omega} \times I\boldsymbol{\omega}) \\ &\qquad\qquad \dot{\bar{\boldsymbol{\eta}}} = \frac{1}{2}\Xi(\bar{\boldsymbol{\eta}})\boldsymbol{\omega} \\ &\text{boundary constraints: } \boldsymbol{\omega}(t_f) = 0 \\ &\qquad\qquad \bar{\boldsymbol{\eta}}(t_f) - \bar{\boldsymbol{\eta}}_f = 0 \\ &\text{path constraint: } \boldsymbol{\sigma}(t) \cdot \boldsymbol{\sigma}_x - \cos(\alpha_x) \le 0 \quad \forall\, t \in [t_0, t_f] \\ &\text{control constraint: } \|\boldsymbol{M}_\nu(t)\| - M_{\max} \le 0, \quad \nu = 1,2,3 \quad \forall t \in [t_0, t_f] \\ &\text{initial conditions: } \boldsymbol{\omega}(t_0) = 0 \\ &\qquad\qquad \bar{\boldsymbol{\eta}}(t_0) - \bar{\boldsymbol{\eta}}_0 = 0 \end{aligned} \tag{15}$$

In Eq. (15), $\bar{\boldsymbol{\eta}}_0$ and $\bar{\boldsymbol{\eta}}_f$ denote the exact initial and final angular positions; the angular velocity must be zero in $t = t_0$ and in $t = t_f$. The boundary constraints and the initial conditions are equality constraints, whereas the path and the control are in the form of inequality constraints. The solutions that satisfy all the boundary and path constraints, and the initial conditions lie inside the admissible research space (ARS), which is clearly a subspace of the FRS.

Each particle evaluates all the constraints and the conditions of the problem during the evolution from the FRS to the ARS. The swarm progressively moves toward the individuals (if the local best search is chosen) or individual (if instead the global best search is preferred), which provides the minimum time of maneuver, ensuring the fulfillment of boundary and path constraints. A very critical point is the choice of the fitness function. The fitness function is selected in the form of an exterior penalty function (explained for example in [42–44] with special regard to genetic algorithms, or more generally in [23]). Such a function is problem-dependent and must be built according to the characteristics of the constraints. In this algorithm, the following expression has been adopted:

$$\phi = t_f + \sum_{i}^{N_{\text{eq}}} H_i + \sum_{j}^{N_{\text{ineq}}} G_j + cN_{\text{viol}} \tag{16}$$

where H_i is the penalty function for the ith equality constraint, G_j is the penalty function for the jth inequality constraint, N_{viol} is the number of violated constraints, and c is a user-defined constant. In fact, as far as the equality constraints are concerned, i.e., the boundary conditions in Eq. (15), they have been divided into position and velocity.

Defining the state with $\bar{\boldsymbol{\eta}}$ and $\dot{\bar{\boldsymbol{\eta}}}$, for the generic equality constraint:

$$h_i(\bar{\boldsymbol{\eta}}, \dot{\bar{\boldsymbol{\eta}}}) = 0 \tag{17}$$

the associated penalty function H_i is

$$H_i = k_i \max\{0, |h_j(\bar{\boldsymbol{\eta}}, \dot{\bar{\boldsymbol{\eta}}})| - \Delta_{\text{eq}_i}\} \tag{18}$$

where k_i is a user-defined constant, and Δ_{eq_i} is a tolerance value imposed by the user.

In the present case, the $N_{\text{eq}} = 2$ penalty functions associated to the equality constraints are

$$H_1 = k_1 \sum_{j=1}^{3} \max\{0, |\omega_j(t_f)| - \Delta_{\text{eq}_1}\}$$

$$H_2 = k_2 \sum_{j=1}^{4} \max\{0, |\eta_j(t_f) - \eta_{j,f}| - \Delta_{\text{eq}_2}\} \tag{19}$$

The most important feature of the present formulation is how the constraints are considered. In fact, different from other works in the literature [21], here the constraint tolerances Δ_{eq_i} decrease during the swarm evolution. In this work, we have set a piecewise linear decreasing law: the more the solution converges, the more the tolerance Δ_{eq_i} decreases. For the computational code used in the proposed simulation, the intervals of the piecewise linear function are defined as a function of the tolerance itself. A constraint is violated when the magnitude of $h_i(\bar{\boldsymbol{\eta}}, \dot{\bar{\boldsymbol{\eta}}})$ is greater than the tolerance. Therefore, PSO may easily find its own way toward the pseudo-optimal solution bringing the swarm toward the particles, which satisfy the constraints. With regard to the inequality constraints, we only need to satisfy the path constraint in Eq. (15) for the N_{ineq} light sources:

$$g_j(\bar{\boldsymbol{\eta}}, \dot{\bar{\boldsymbol{\eta}}}) = \boldsymbol{\sigma}(t) \cdot \boldsymbol{\sigma}_x - \cos(\alpha_x) < 0 \tag{20}$$

The associated penalty function is

$$G_j = l_j \sum_{i=0}^{n} \nu(t_i) \tag{21}$$

where l_j is a user-defined constant, and

$$\nu(t_i) = \begin{cases} 0 & \text{if } \boldsymbol{\sigma}(t_i) \cdot \boldsymbol{\sigma}_x - \cos(\alpha_x) < \Delta_{\text{ineq}_j} \\ 1 & \text{otherwise} \end{cases} \tag{22}$$

The inequality constraints are fully satisfied only when $\Delta_{\text{ineq}_j} = 0$. Similar to the case of the equality constraints, Δ_{ineq_j} is a decreasing tolerance. Notice that, from Eq. (22), it follows that

$$g_j(\bar{\boldsymbol{\eta}}, \dot{\bar{\boldsymbol{\eta}}}) < \Delta_{\text{ineq}_j} \quad \forall\, t \in [t_0, t_f] \Leftrightarrow G_j = 0 \tag{23}$$

The last term in Eq. (16), N_{viol}, is the counter that considers the violated constraints; every time the global best particle reaches the value of $N_{\text{viol}} = 0$, the precision is improved, and the tolerances decrease. The control constraint in Eq. (15) is accounted for in Eq. (12).

The tolerance Δ_{ineq} for the keep-out constraint is considered in a peculiar way during the swarm evolution. Denoting with j the index of the external loop:

1) For $j = 1$, the keep-out cones constraint is not considered. The swarm moves toward the maneuver that goes from the initial to the final points, thus minimizing the time. If the optimal maneuver for this unconstrained case is known, it can be used as a guess; in this case, only one particle of the swarm is set to take the form of this optimal maneuver. In this manner, all the other particles move toward this position, ensuring that the swarm initiates the search inside the FRS.

2) From $j = 2$ to $j = j^*$ (where j^* may be defined by the user), the keep-out cones constraint is gradually included accordingly. Consequently, and if all constraints are satisfied, as j increases from $j = 2$ to $j = j^*$, the tolerances decreases. In particular, in this first part of the loop, the local search is set to be more significant than the global search. As a result, the swarm is divided into different search groups that increase the probability to find the global minimum of the problem.

3) From $j = j^* + 1$ until the end of the external loop, the keep-out cones constraint is entirely included (i.e., $\Delta_{\text{ineq}} = 0$), and the swarm will continue to minimize the maneuver time, improving the accuracy of the solution.

In the DPSO, the tolerances for the final position and the final velocity decrease as well; however, they are never set to zero.

The DPSO algorithm is built with the following steps. First, the swarm and the tolerances are initialized. Then, the external loop starts; when the number of violated constraints is equal to zero, the tolerances are updated and the best positions are reset. Inside the external loop, another internal loop of N_{int} iterations is executed: here, we interpolate the control $\boldsymbol{M}$; we integrate Euler's equation and the kinematics with Eqs. (11) and (13); and we obtain the sensor axis position using Eq. (14) and compute the fitness function ϕ with Eq. (16). Next, the best positions of the particles are updated through Eqs. (2–4). At the end of each iteration of the internal loop, the velocity and the position of the particles are updated according to Eqs. (5) and (12). Designating with J_g^1, J_g^2, J_g^3, and J_g^4 the last four updated values of the global best performance index, the external loop stops when the following user-defined tolerance is satisfied:

$$\frac{(J_g^3 - J_g^4) + (J_g^2 - J_g^3) + (J_g^1 - J_g^2)}{3} < \varepsilon \tag{24}$$

where J_g is evaluated as in Eq. (3), i is the index of the external loop, and ε is a user-defined tolerance.

IV. Inverse Method: From the Kinematics to the Dynamics

On the one hand, if the direct integration of the control law is a simple and traditional way of dealing with these types of problems, on the other hand, it shows some issues due to the low speed of the integration process and because it is subject to numerical errors. In fact, in the numerical code, the use of a function for the integration of the Euler equations is necessary when high tolerances values are needed. Moreover, boundary constraints cannot be completely satisfied.

We propose a different approach in this work. Instead of applying the PSO to the control, we use the PSO approach to the kinematics. This technique may be referred to as inverse dynamics, and the acronym IPSO (for "inverse particle swarm optimization") will be used consequently hereinafter. The novelty of the IPSO method lies in the fact that the achieved maneuver may not only be used as initial guess, and as the DPSO maneuver, but it may also be used as a suboptimal maneuver without requiring a pseudospectral optimal control software (POCS) refinement. In particular situations, that is, IPSO may be used as a suboptimal planner.

As in previous cases, the particles are arrays containing 1) the value of the maneuver time $t_f^{(i)}$, and 2) the angular displacement $\zeta_j^{(i)}(k)$, where i identifies the particle, $j = 1, 2, \ldots, n$ and $k = 1, 2, \ldots, m$. In particular, n is the number of scalars required by the chosen angular representation. m is equal to the number of points used for the interpolation of the angular displacement. The m points are associated to time instants equally spaced between t_0 and $t_f^{(i)}$. The

kinematics is obtained in $n+1$ points (so that $t = t_0, t_1, \ldots, t_n$ and $t_n = t_f^{(i)}$) interpolating the m points $\zeta_j^{(i)}(k)$ with basis splines (B-splines).

As in the previous approach, $\Delta\zeta_j^{(i)}(k)$ and $\Delta t_f^{(i)}$ are the velocities associated to the kinematics and the maneuver time of each particle of the swarm. In this case, Eq. (6) takes the following form:

$$\begin{aligned} &|\Delta\zeta_j^{(i)}(k)| \le 0.2 \cdot \tan(\theta^*/4), \qquad |\zeta_j^{(i)}(k)| \le \tan(\theta^*/4) \\ &|\Delta t_f^{(i)}| < 0.1 \cdot (t_{\max} - t_{\min}), \qquad t_{\min} < t_f^{(i)} < t_{\max} \\ &\quad i = 1, \ldots, n_{\text{particles}}, \qquad j = 1, \ldots, m \end{aligned} \tag{25}$$

The expression $\tan(\theta^*/4)$ is explained in Eq. (27); θ^* is an angle which satisfies $\theta^* \ge \Theta_f$, and Θ_f is the imposed angle of maneuver. The time constraints $t_{\max}$ and $t_{\min}$ may be defined by knowing the unconstrained solution. The values 0.1 and 0.2 are selected to make the maximum velocities equal to the 10% of the dynamic range of the particles, as already explained with regard to Eq. (6).

The initialization of the kinematics and the maneuver time of each particle are based on a uniform random distribution of the particles within the constraints of Eq. (25).

In this approach, given the angular displacement of each particle, the control is directly evaluated by Euler's equation [Eq. (8)]. As a consequence, $\boldsymbol{\omega}$ and $\dot{\boldsymbol{\omega}}$ are needed to obtain $\boldsymbol{M}$. There are fundamentally two problems associated with this method.

1) Computation of the body-fixed angular velocities from the inertial description of the kinematics. This calculation requires an inversion of a matrix, whose dimensions depend on the selection of the attitude representation. In this case, a description of the attitude through the quaternions is quite difficult and approximated, because the 3 by 4 kinematic matrix $\Xi(\tilde{\boldsymbol{\eta}})$ should be inverted.

2) Derivation of the curves that interpolate the angular displacement; it is highly recommended to obtain the analytical solution for $\dot{\boldsymbol{\omega}}$ to reduce numerical errors and computational time. Usually, to reach this purpose, we need the first and the second analytical derivatives of the angular displacement.

The first issue is related to the choice of the attitude representation. An attitude description with only three parameters is essential to avoid the invertibility of the kinematics matrix. The modified Rodrigues parameters (MRPs) [41] are suitable for this issue. Moreover, MRPs show no singularity during the maneuver because the imposed Θ_f is below 2π.

The mathematical formulation that describes the kinematics through the MRPs is reflected below a vector $\boldsymbol{p}$, defined as follows:

$$\boldsymbol{p} = \frac{\boldsymbol{\eta}}{1+\eta_4} \tag{26}$$

where $\boldsymbol{\eta}$ and η_4 are the vectorial and scalar components of the quaternion. Moreover, $\boldsymbol{p}$ may be rewritten in terms of axis and angle of rotation as

$$\boldsymbol{p}(\hat{\boldsymbol{n}}, \theta) = \tan(\theta/4)\hat{\boldsymbol{n}} \tag{27}$$

The rotation matrix using the modified Rodrigues parameters appears as

$$R(\boldsymbol{p}) = I + \frac{4(1-|\boldsymbol{p}|^2)}{(1+|\boldsymbol{p}|^2)^2}[\tilde{\boldsymbol{p}}] + \frac{8}{(1+|\boldsymbol{p}|^2)^2}[\tilde{\boldsymbol{p}}]^2 \tag{28}$$

where $[\tilde{\boldsymbol{p}}]$ is defined as

$$[\tilde{\boldsymbol{p}}] = \begin{bmatrix} 0 & p_3 & -p_2 \\ -p_3 & 0 & p_1 \\ p_2 & -p_1 & 0 \end{bmatrix} \tag{29}$$

In particular,

$$\boldsymbol{\sigma}(t) = R(\boldsymbol{p})^T \boldsymbol{\sigma}(t_0) \tag{30}$$

The derivative of the modified Rodrigues parameters are related to the angular velocity by the following equation:

$$\dot{\boldsymbol{p}} = \frac{1}{4}\Psi(\boldsymbol{p})\boldsymbol{\omega} \tag{31}$$

where the matrix $\Psi(\boldsymbol{p})$ is defined as

$$\Psi(\boldsymbol{p}) = [(1 - \boldsymbol{p}^T\boldsymbol{p})I + 2[\tilde{\boldsymbol{p}}] + 2\boldsymbol{p}\boldsymbol{p}^T] \tag{32}$$

For the following development of the inverse dynamics with PSO algorithm, it is necessary to find $\boldsymbol{\omega}(\boldsymbol{p}, \dot{\boldsymbol{p}})$ and $\dot{\boldsymbol{\omega}}(\boldsymbol{p}, \dot{\boldsymbol{p}}, \ddot{\boldsymbol{p}})$. As far as the former vector is concerned, it is quite simple to obtain it from Eq. (31):

$$\boldsymbol{\omega} = 4\Psi^{-1}(\boldsymbol{p})\dot{\boldsymbol{p}} \tag{33}$$

Ψ^{-1} is defined as a near-orthogonal matrix because its inverse matrix is proportional to its transpose:

$$\Psi^{-1}(\boldsymbol{p}) = \frac{\Psi^T(\boldsymbol{p})}{(1+\boldsymbol{p}^T\boldsymbol{p})^2} \tag{34}$$

From Eq. (33), an important consequence may be drawn:

$$\boldsymbol{\omega} = 0 \Leftarrow \dot{\boldsymbol{p}} = 0 \tag{35}$$

From Eq. (33), $\dot{\boldsymbol{\omega}}(\boldsymbol{p}, \dot{\boldsymbol{p}}, \ddot{\boldsymbol{p}})$ may be derived as

$$\dot{\boldsymbol{\omega}} = 4(\dot{\Psi}^{-1}(\boldsymbol{p})\dot{\boldsymbol{p}} + \Psi^{-1}(\boldsymbol{p})\ddot{\boldsymbol{p}}) \tag{36}$$

where $\dot{\Psi}$ and $\dot{\Psi}^{-1}$ are evaluated as

$$\begin{aligned} \dot{\Psi} &= [-(\dot{\boldsymbol{p}}^T\boldsymbol{p} + \boldsymbol{p}^T\dot{\boldsymbol{p}})I + 2[\dot{\tilde{\boldsymbol{p}}}] + 2(\dot{\boldsymbol{p}}\boldsymbol{p}^T + \boldsymbol{p}\dot{\boldsymbol{p}}^T)] \\ \dot{\Psi}^{-1} &= \frac{\dot{\Psi}^T}{(1+\boldsymbol{p}^T\boldsymbol{p})^2} - \frac{2\Psi^T}{(1+\boldsymbol{p}^T\boldsymbol{p})^3}(\dot{\boldsymbol{p}}^T\boldsymbol{p} + \boldsymbol{p}^T\dot{\boldsymbol{p}}) \end{aligned} \tag{37}$$

These equations fully describe the attitude kinematics through the modified Rodrigues parameters. The main feature of the previous equations is that an analytical closed-form solution is found to compute $\boldsymbol{\omega}$ and $\dot{\boldsymbol{\omega}}$. Although the mathematical form of these equations is more complex than the mathematical form described in the attitude kinematics with the quaternions, the advantage is in dealing with square matrices. To summarize these results, placing Eqs. (33) and (36) in Eq. (8), the following relation is obtained:

$$\boldsymbol{M} = f(\boldsymbol{p}, \dot{\boldsymbol{p}}, \ddot{\boldsymbol{p}}) \tag{38}$$

The second issue is related to the formulas reported in Eqs. (33–36). In fact, the representation used for the description of the kinematics must easily allow the computation of the needed derivatives. The problem may be solved using the B-spline interpolation with the Cox–de Boor recursion formula [45–48]. B-splines are based on the fact that each segment of the interpolating curve is built upon several rather than on only one polynomial. The clamped version of the B-spline is used here, where clamped means that the curve passes through the initial and final control points $\boldsymbol{U}_0$ and $\boldsymbol{U}_f$. The other control points $\boldsymbol{U}_1, \boldsymbol{U}_2 \ldots, \boldsymbol{U}_{f-1}$ define the shape of the whole curve.

For positive results, it is recommended to interpolate the angular displacement at least with a curve based on functions with differentiability class C^4. In fact, in this case, the second derivative (i.e., the angular acceleration) is a smooth function of class C^2. Moreover, the first derivative of the angular displacement at $t = t_0$ and $t = t_f$ may be user-defined because the clamped B-spline curve is tangent to the first and last leg of the control polyline, as shown in Fig. 3. The points U_i are the curve control points. The curve is tangent to the first leg $\overline{\boldsymbol{U}_0\boldsymbol{U}_1}$ in $\boldsymbol{U}_0$ and tangent to the last leg $\overline{\boldsymbol{U}_5\boldsymbol{U}_6}$ in $\boldsymbol{U}_6$.

As a consequence, to obtain a rest-to-rest (i.e., with velocity equal to zero in $t = t_0$ and $t = t_f$) maneuver, it is only necessary to set the

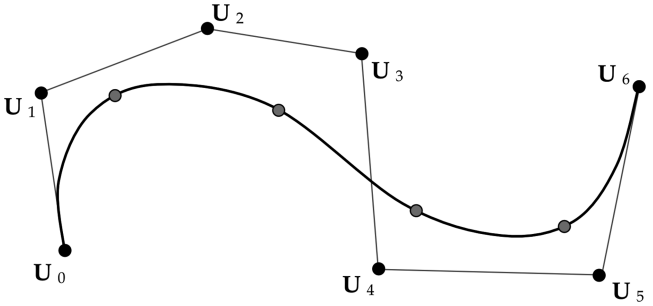

Fig. 3 Example of interpolation curve obtained with a clamped B-spline.

first two points $\boldsymbol{U}_0$ and $\boldsymbol{U}_1$, the last two points $\boldsymbol{U}_{f-1}$, and $\boldsymbol{U}_f$ of each angular displacement curve at the same ordinate value. In fact, as stated in Eq. (35), when $\omega_x = \omega_y = \omega_z = 0$, also the modified Rodrigues parameters are equal to zero. Summarizing, four points for each of the three curves interpolating the MRP are defined. As a result, we can set $\dot{\boldsymbol{p}}(t_0) = 0$ and $\dot{\boldsymbol{p}}(t_f) = 0$ by imposing

$$\boldsymbol{U}_0 = \boldsymbol{U}_1 = \boldsymbol{p}_0 \quad \boldsymbol{U}_{f-1} = \boldsymbol{U}_f = \boldsymbol{p}_f \tag{39}$$

In this formulation, the problem may be summarized as follows:

Find: $\min t_f$

subjected to

dynamic constraints: $\boldsymbol{M} = f(\boldsymbol{p}, \dot{\boldsymbol{p}}, \ddot{\boldsymbol{p}})$

the boundary constraints: $\dot{\boldsymbol{p}}(t_f) = 0$

$\boldsymbol{p}(t_f) - \boldsymbol{p}_f = 0$

path constraint: $(R(\boldsymbol{p})^T \boldsymbol{\sigma}(t_0)) \cdot \boldsymbol{\sigma}_x - \cos(\alpha_x) \le 0 \quad \forall\, t \in [t_0, t_f]$

control constraint: $\|\boldsymbol{M}_\nu(t)\| - \boldsymbol{M}_{\max} \le 0, \qquad \nu = 1,2,3 \quad \forall\, t \in [t_0, t_f]$

initial conditions: $\dot{\boldsymbol{p}}(t_0) = 0$

$\boldsymbol{p}(t_0) - \boldsymbol{p}_0 = 0$ (40)

The extreme simplification of this approach lies in the fact that both the boundary constraints and the initial conditions are imposed a priori for each particle.

Consequently, the fitness function is chosen as follows:

$$\phi = t_f + \sum_{i=1}^{N_{\text{ineq}}} G_i + c_1 \sum_{j=1}^{3} \sum_{l=1}^{n} \max\{0, |M_j(l)| - M_{\max}\} + c_2 N_{\text{viol}} \tag{41}$$

where n is the length of the control vector, and $M_j(l)$ is the jth element of the control along the i axis. The term with the double summation penalizes the particles whose values of the control exceed $M_{\max}$; in fact, if $|M_j(l)| \le M_{\max}$, the maximum function value returns 0, otherwise the values of $|M_j(l)|$ so that $|M_j(l)| > M_{\max}$ are penalized. The parameter c_1 is another user-defined constant useful to set the exact order of magnitude for the double summation compared to the other terms. The meanings of N_{viol} and G_i are the same as in the direct method. As it can be seen, the fitness function does not need to consider the equality constraints. In this case, in fact, each particle is built to fully satisfy the boundary constraints.

As previously explained, the keep-out constraint is taken into account. Note that, in this approach, the final attitude and angular velocity do not require the introduction of the tolerances because the final state is imposed by the problem formulation.

The IPSO algorithm is built with the following steps. First the swarm, the constants and the tolerances are initialized. Then, the external loop starts; when the number of violated constraints is equal to zero, the tolerances are updated and the best positions are reset. Inside the external loop, another internal loop of N_{int} iterations is executed: here, we interpolate the kinematics $\boldsymbol{p}$, $\dot{\boldsymbol{p}}$, and $\ddot{\boldsymbol{p}}$; we obtain $\boldsymbol{\omega}$ and $\dot{\boldsymbol{\omega}}$ using Eqs. (33–36); we obtain the sensor axis position with Eq. (14); and we compute $\boldsymbol{M}$ from Eq. (8) as well as the fitness function ϕ with Eq. (16). Then, best positions of the particles are updated through Eqs. (2–4). At the end of each iteration of the internal loop, the velocity and the position of the particles are updated according to Eqs. (5) and (25). The external loop stops as already described for the DPSO through Eq. (24).

The IPSO approach may be used as 1) initial guess for a POCS, as the DPSO approach, and 2) planner for near minimum-time maneuvers; in fact, the algorithm guarantees that all constraints are satisfied. Moreover, the numerical results will prove that the maneuver time is very close to that obtained with the POCS approach.

V. Numerical Results

A satellite for Earth observation in low Earth orbit (LEO) is taken as test case. The nominal attitude is defined as follows.

1) The $\boldsymbol{Z}_b$ axis points in the nadir direction toward the Earth.

2) The $\boldsymbol{X}_b$ axis is in the direction of the spacecraft velocity vector for circular orbits.

3) The $\boldsymbol{Y}_b$ axis completes the right-handed coordinate system, and it is perpendicular to the orbital plane in the negative orbit normal direction.

The inertial reference frame BRF_0 is associated to the coordinate system $\{\boldsymbol{X}_b, \boldsymbol{Y}_b, \boldsymbol{Z}_b\}$ at time $t = t_0$. A typical operative maneuver is defined as a rotation around the $\boldsymbol{X}_b$ axis (roll rotation): the satellite switches between the right-pointing, with a positive roll angle and $\boldsymbol{Z}_b$ and pointing to the right with respect to the ground track, and the symmetric left-pointing configuration with a negative roll angle; see Fig. 4. In the following simulation, roll angles set to $\Theta_f = 60$ deg and $\Theta_f = 135$ deg will be used to switch from the right-pointing position to the left-pointing position.

Assuming that the inertia tensor is diagonal in the BRF and has the following values:

$$I = \begin{bmatrix} I_1 & 0 & 0 \\ 0 & I_2 & 0 \\ 0 & 0 & I_3 \end{bmatrix} = \begin{bmatrix} 3000 & 0 & 0 \\ 0 & 4500 & 0 \\ 0 & 0 & 6000 \end{bmatrix} \text{kg} \cdot \text{m}^2 \tag{42}$$

Let us consider a star tracker sensor mounted on the $\boldsymbol{Y}_b\boldsymbol{Z}_b$ plane with the unit vector $\boldsymbol{ST}$ expressed in the in BRF as

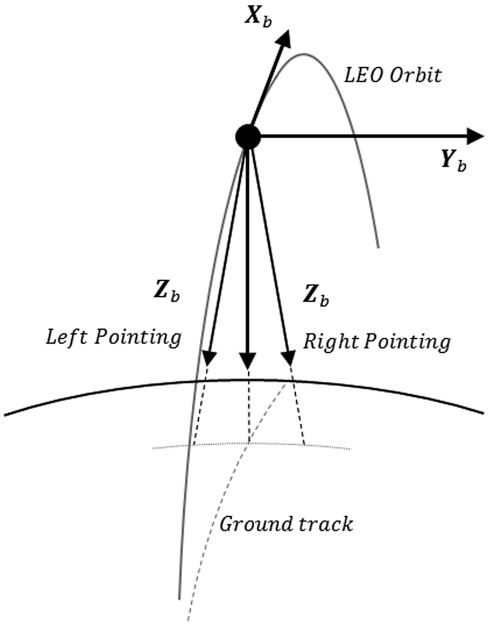

Fig. 4 Right- and left-looking configurations of the body reference frame.

Table 1 Direction of sun and moon in BRF_0 for case 1

Case 1: roll angle $\Theta_f = 60$ deg	Sun	Moon
x	−0.58	0.40
y	−0.08	−0.13
z	−0.81	−0.90

Table 2 Direction of sun and moon in BRF_0 for case 2

Case 2: roll angle $\Theta_f = 60$ deg	Sun	Moon
x	−0.65	0.17
y	0.28	−0.26
z	−0.707	−0.95

$$\boldsymbol{ST} = [0 - 0.62 - 0.79]^T \tag{43}$$

We assume that the attitude maneuvers are obtained through three independent torques aligned with the BRF with same maximum value:

$$M_{\max} = 0.25 \text{ N} \cdot \text{m} \tag{44}$$

Two keep-out cones are considered, centered respectively on the sun and on the moon. The corresponding half-angles are set to 40 and 17 deg. Two different case studies are proposed, whose characteristics are shown in the Tables 1 and 2. In both cases, the roll angle is set to $\Theta_f = 60$ deg; a third case with a roll angle $\Theta_f = 135$ deg will be further studied using the hybrid method. The directions of the moon and the sun are referred in BRF_0. The mean computational times required by the POCS (without best guess) for cases 1 and 3 are 351.25 and 1522.57 s, respectively. These values will be used for further comparisons with the proposed methods.

As a further explanation, the geometry of case 1 is represented in Fig. 5, where the two keep-out cones, the optimal maneuver, and the inertial frame are reported.

To compare the results, both the DPSO and the IPSO have been tested with 30 particles. With regard to Eq. (5), the inertia weight w, the local best constant c_l, and the global best constant c_g are given by the following linear laws:

$$w = \begin{cases} w_0 - \frac{w_0 - w_f}{N^*}\left(j - 1 - \frac{l-1}{N_{\text{int}}}\right) & \text{if } w > w_f \\ w_f & \text{otherwise} \end{cases}$$

$$c_l = \begin{cases} c_{l_0} - (j-1)\frac{c_{l_0} - c_{l_f}}{N^*} & \text{if } c_l > c_{l_f} \\ c_{l_f} & \text{otherwise} \end{cases}$$

$$c_g = \begin{cases} c_{g_0} - (j-1)\frac{c_{g_0} - c_{g_f}}{N^*} & \text{if } c_g < c_{g_f} \\ c_{g_f} & \text{otherwise} \end{cases}$$

$$\text{with} \quad j = 1, 2, \ldots, N^*, \qquad l = 1, 2, \ldots, N_{\text{int}} \tag{45}$$

where we set $N_{\text{int}} = 20$, $N^* = 30$, $w_0 = 1.2$, $w_f = 0.6$, $c_{l_0} = c_{g_f} = 2$, and $c_{l_f} = c_{g_0} = 0$. In particular, N_{int} is the number of iteration for the internal loop, and l and j are the index of the internal and external loop, respectively. N^* is a parameter that determines the decreasing rate of w, c_l, and c_g. The scale factor r has been set to 0.9, and the cognitive constant c_p is equal to 1.5.

The constraint tolerances Δ_{eq} and Δ_{ineq} have the same piecewise linear law expressed as

$$\Delta_{j+1} = \Delta_j - 10^{-\xi} \tag{46}$$

where $\Delta_1 = 0.14$, and ξ is selected to satisfy $10^{-\xi} < \Delta_j \leq 10^{-\xi+1}$ with $\xi \in \mathbb{N}$. Note that Eq. (46) is taken into account only when all tolerances are satisfied.

The iteration index j^* related to the setting $\Delta_{\text{ineq}} = 0$ is the external iteration related to $\Delta = 0.08$. The total number of external iterations is a function of the tolerance criteria; both the DPSO and the IPSO stop according to the criterion in Eq. (24) with $\varepsilon = 1\text{e} - 8$.

For numerical reasons, normalized units are considered: the control is divided by $M_{\max}$, the inertia matrix by I_1, and the maneuver time by $\sqrt{I_1/M_{\max}}$. As a consequence, Eqs. (16) and (41) are nondimensional.

The constants k_1, k_2, l_j, and c in Eqs. (16), (19), and (21), as well as c_1 and c_2 in Eq. (41), have been set to 10.

For case 1, the PSO solutions will be shown and compared with the optimal solution obtained with a pseudospectral optimal control software (POCS) [49–53]. Case 2 is presented to underline that the IPSO approach performs better than the DPSO approach. However, the graphs of the solutions for case 2 do not differ much from those of case 1 and consequently will be not reported.

All results are obtained considering a PC with an Intel Core i7-2670QM CPU at 2.20 GHz and 6.00 GB of RAM.

A. Results Obtained with Direct Particle Swarm Optimization

The DPSO algorithm produces a solution with a suboptimal final time; moreover, the boundary constraints are not completely satisfied. However, this kind of solution may be used as optimal initial guess to be given to a POCS, which may eventually find the optimal solution.

The only disadvantage is that the DPSO algorithm is strongly affected by the chosen geometry. The same algorithm may deliver excellent results with regard to the fulfillment of the boundary conditions for one geometry and poor results for another one.

The results obtained in case 1 are shown in Figs. 6–8, where they are compared with the optimal solution. In this case, both the angular displacement and the angular velocity reach the desired final value with acceptable tolerance values. The errors on the boundary constraints are reported in Table 3, where the results of a generic simulation are chosen. From the comparison with the POCS optimal solution, it is clear that the result is in agreement with the optimal solution.

In Table 4, 10 representative results obtained with the DPSO are reported. T refers to the computational time, t_f is the obtained maneuver time, and ϵ represents the percentage relative error with respect to the POCS optimal time $t_f^* = 225.33$ s. The value $\Delta\theta$ is the difference

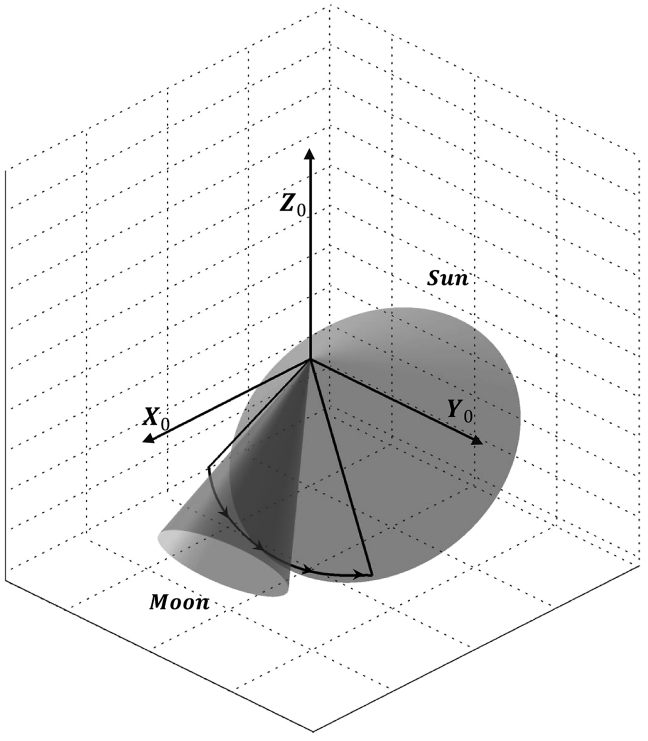

Fig. 5 Geometry of the case 1: the optimal maneuver of the sensor axis is reported.

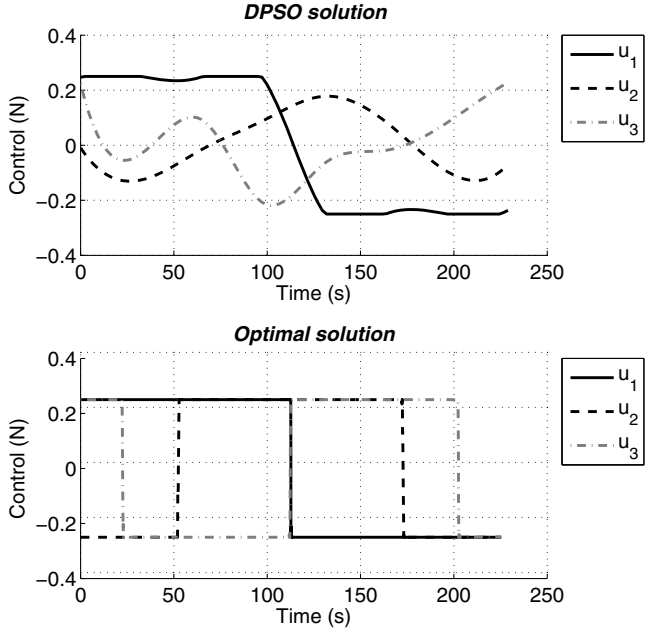

Fig. 6 Control law solution from DPSO compared with the optimal solution, case 1.

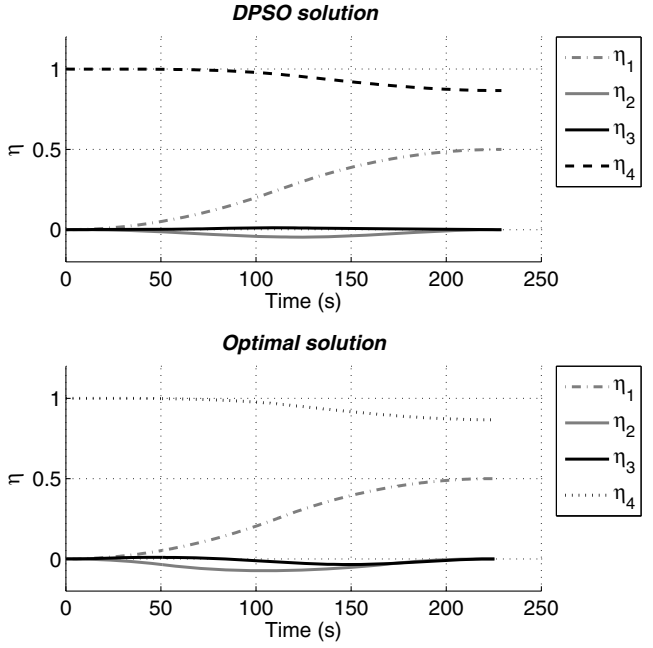

Fig. 7 Quaternions solution from DPSO compared with the optimal solution, case 1.

between the obtained angle of rotation θ and the imposed Θ_f, whereas $\Delta\|\boldsymbol{\omega}\|$ is the final angular velocity. With the DPSO approach, the maneuver time is affected by highly variable relative errors. Moreover, the tests 4 and 7 show that, when t_f is in proximity with t_f^*, the rotation angle is affected by a relatively high error. Furthermore, the required computational time varies from one test to another. This unstable trend of results makes the generic DPSO solution unreliable.

When case 2 is analyzed with DPSO, the results are poorer than those obtained in the first case. The errors on the boundary constraints are reported in Table 5; it can be seen that the errors are greater than those obtained in case 1. The maneuver time t_f, however, is as positive as in those solutions of the first case.

The solutions in cases 1 and 2 using the DPSO approach cannot be used as standalone solutions unless a further control system improves the final time of the maneuver. However, the DPSO solution may always be used as the initial guess for a POCS.

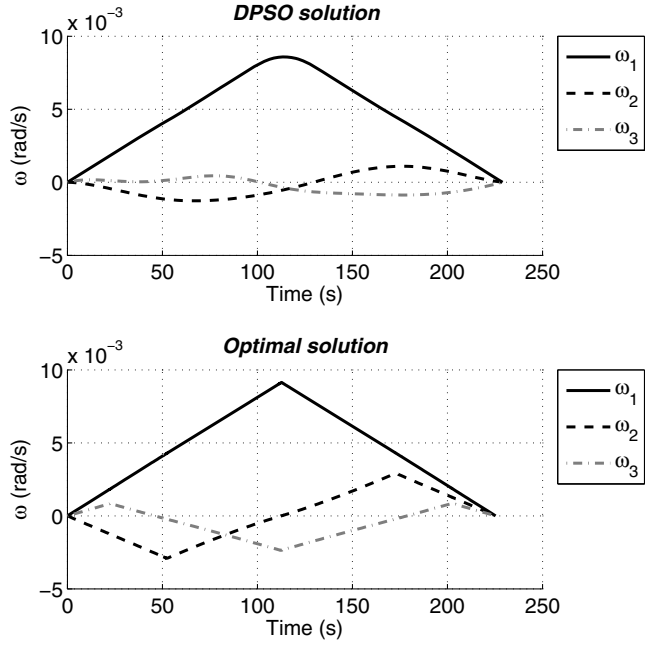

Fig. 8 Angular velocity solution from DPSO compared with the optimal solution, case 1.

B. Results Obtained with Inverse Particle Swarm Optimization

When the maneuver planning with a time close to the optimal one and when the boundary and path constraints are satisfied, the IPSO algorithm may be used as a planner. The advantage of the IPSO maneuver may be evaluated comparing its solution to a POCS solution. As it can be seen in Figs. 9–11, where case 1 is studied, the solution obtained with the proposed algorithm captures all the essential characteristics of the optimal maneuver. Although, as explained in Sec. IV, the IPSO algorithm uses the modified Rodrigues parameter. The angular displacement is described using the quaternions, to easily compare the solutions offered by the different methods. Therefore, for example, the IPSO control (Fig. 9) on each axis is positive when the optimal maneuver requires a positive bang and negative when the optimal maneuver requires a negative bang. The trend of the quaternions (Fig. 10) and that of the angular velocity (Fig. 11) are still closer to the optimal one. Both the angular displacement (i.e., the values of the quaternion) and the angular velocity reach the imposed final value. In fact, the most important difference between the IPSO and the DPSO algorithm is that the IPSO satisfies completely all the boundary conditions because they are imposed at the beginning of the algorithm.

The evolution of the swarm is reported through Figs. 12–14, where the maneuver is shown in the inertial reference frame in terms of latitude and longitude. The reference frame is rotated to make the figures easier to read, i.e., the axes of the keep-out cones are brought near the equator axis (where the latitude is equal to zero) to minimize the deformation of the cones.

As it can be seen in Fig. 12, in the first step, the sensor axis goes from A to B without considering the two keep-out cones and passing through them. The idea used in the simulation is that one of the

Table 3 Final state values for a representative experiment with DPSO, case 1

Parameter	Final value	Expected final value	Absolute error
ω_1, rad/s	$-2.5061 \cdot 10^{-6}$	0	$-2.5061 \cdot 10^{-6}$
ω_2, rad/s	$-8.3160 \cdot 10^{-7}$	0	$-8.3160 \cdot 10^{-7}$
ω_3, rad/s	$-1.9921 \cdot 10^{-7}$	0	$-1.9921 \cdot 10^{-7}$
η_1	0.4998	0.5	0.0002
η_2	$-2.8491 \cdot 10^{-4}$	0	$-2.8491 \cdot 10^{-4}$
η_3	$2.1350 \cdot 10^{-4}$	0	$2.1350 \cdot 10^{-4}$
η_4	0.8661	0.8660	0.0001
t_f, s	228.6434	225.3324	3.3110

Table 4 List of 10 different results obtained with DPSO for case 1

T_{DPSO}, s	t_f, s	ϵ, %	$\Delta\theta$, deg	$\Delta\|\boldsymbol{\omega}\|$, deg/s
217.51	228.19	1.27	0.12	0.05
263.98	234.83	4.22	0.29	0.09
284.46	228.11	1.23	0.08	0.03
242.52	225.22	−0.05	2.10	0.87
218.06	249.17	10.58	0.06	0.02
190.74	240.22	6.61	0.75	0.27
216.35	225.10	−0.10	2.34	0.71
216.87	228.51	1.41	0.05	0.03
248.90	228.09	1.22	0.08	0.06
221.10	235.95	4.71	0.05	0.03

Table 5 Final state values for a representative experiment with DPSO, case 2

	Final value	Expected final value	Absolute error
ω_1, rad/s	$1.7812 \cdot 10^{-4}$	0	$1.7812 \cdot 10^{-4}$
ω_2, rad/s	$1.7981 \cdot 10^{-4}$	0	$1.7981 \cdot 10^{-4}$
ω_3, rad/s	$-1.3373 \cdot 10^{-4}$	0	$1.3373 \cdot 10^{-4}$
η_1	0.4804	0.5	0.0196
η_2	−0.0200	0	0.0200
η_3	−0.0070	0	0.0070
η_4	0.8768	0.8660	0.0108
t_f, s	222.1578	225.9021	3.7443

particles of the swarm is initialized with a linear interpolation between the initial and the final values of the modified Rodriguez parameters, whereas for the value of the final time, it is not required to select particular value. This expedient guarantees that all particles enter the feasible search space in the first cycle of the evolution, when the keep-out cones are not contemplated.

In the following cycles (Fig. 13), the swarm moves in such a way that the keep-out cones constraint is progressively satisfied; the global best particle moves from A to B to minimize the maneuver time and the inclusion of the keep-out cones simultaneously. The arrows and the different colors of the maneuvers show the direction of the evolution. The progressive removal of the keep-out cones is implemented to make the evolution faster. In fact, only a little alteration of the global best is needed to completely satisfy the renewed tolerance value.

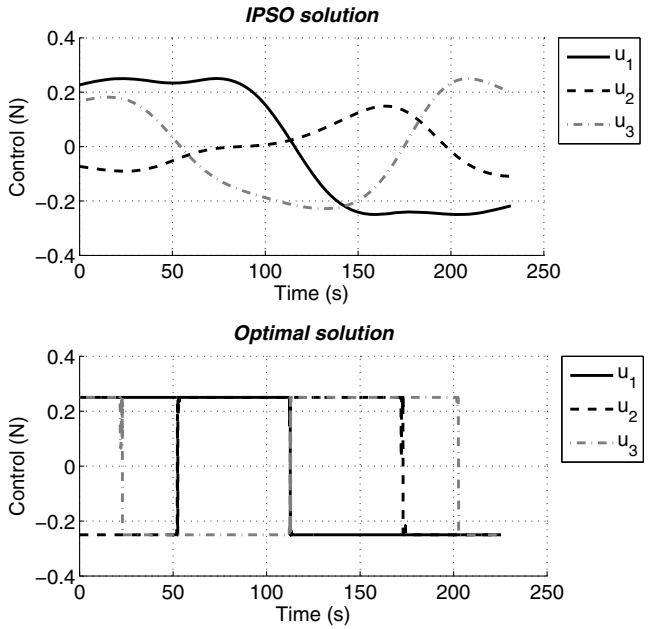

Fig. 9 Control law solution from IPSO compared with the optimal solution, case 1.

Finally, as it can be seen in Fig. 14, the swarm satisfies completely the keep-out cones constraint and evolves to minimize the maneuver time. With regard to the kinematics, no relevant differences are to be seen between the IPSO and the POCS solutions. The most important criterion, which is to prevent the axis from entering the keep-out cones, is a common characteristic of all the geometries studied with the IPSO approach.

We observe an important feature of the IPSO through the different numerical simulations: even when the evaluation of the direction of the trajectory is incorrect in the first cycles (i.e., the maneuver does not pass between the cones), the swarm is able to change direction and set the new global best particle thanks to the division of the swarm in several search groups.

With regard to the computational time, a list of 10 different experiments carried out for case 1 are reported in Table 6. As before,

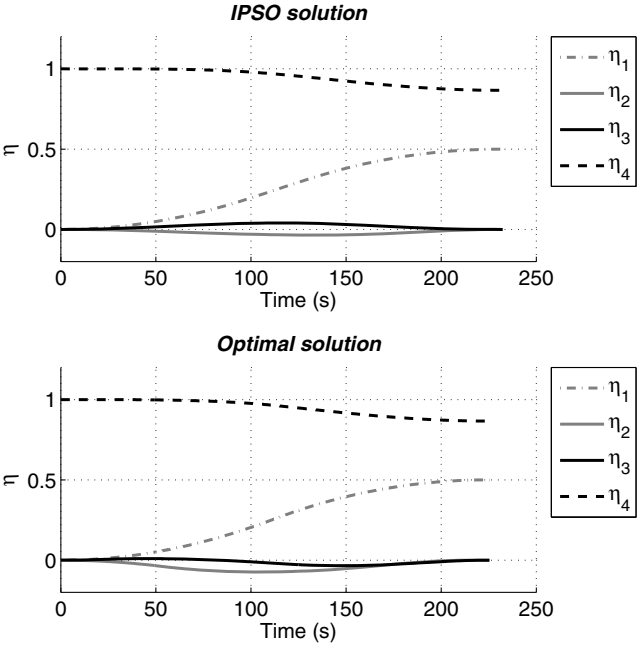

Fig. 10 Quaternions solution from IPSO compared with the optimal solution, case 1.

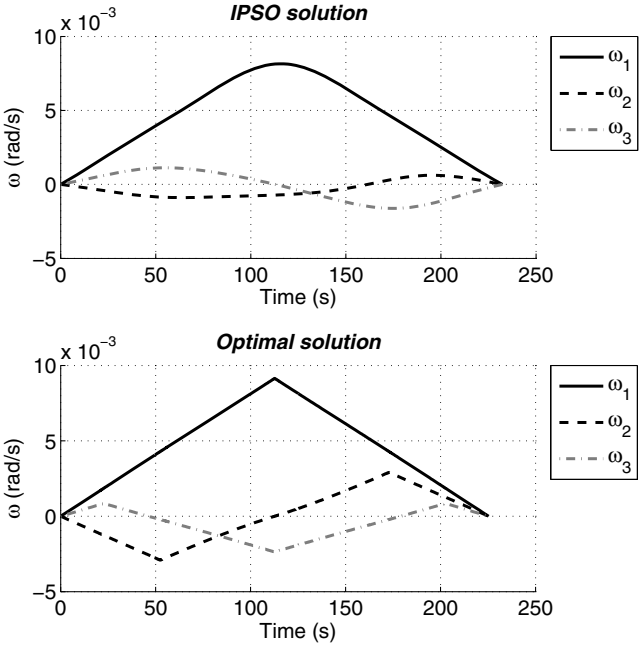

Fig. 11 Angular velocity solution from IPSO compared with the optimal solution, case 1.

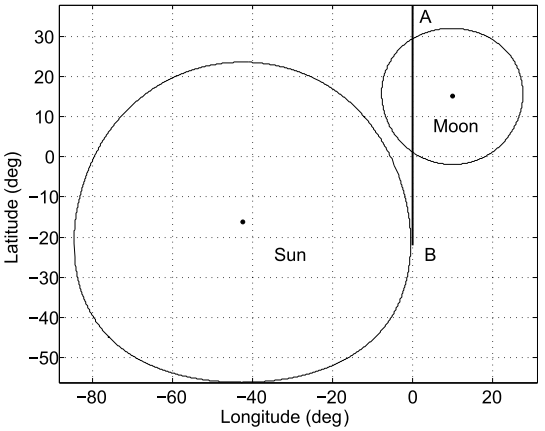

Fig. 12 Sensor path at the initialization of the IPSO algorithm.

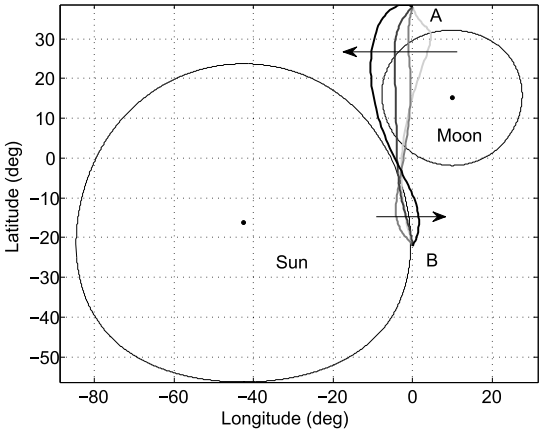

Fig. 13 Evolution of the global best particle in the swarm in the first cycles.

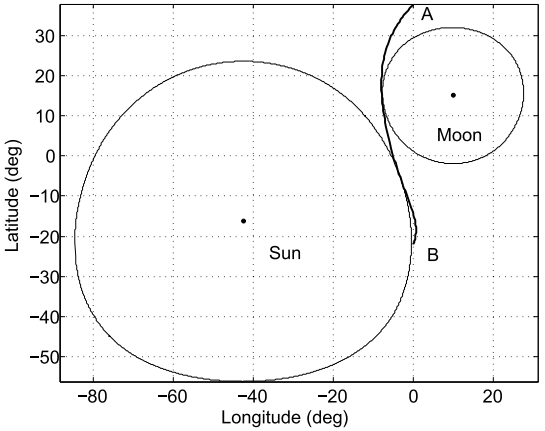

Fig. 14 Final maneuver obtained at the end of the evolution.

T refers to the computational time, t_f refers to the maneuver time, and ϵ represents the relative error with respect to the POCS optimal time $t_f^* = 225.33$ s. The following important remarks may be pointed out.

1) The computational time does not substantially change from one experiment to another.

2) The computational time has been halved with respect to the computational time required by the DPSO algorithm.

3) The error between the obtained final time and the optimal one (225.33 s) is always about 3%, which is a very reasonable result. The unstable trend of the DPSO approach is removed, making the IPSO a more reliable solution.

With regard to the solution in case 2, IPSO provides the same results as in case 1. The computational time is equal to the computational time in case 1. The boundary constraints are completely satisfied, as well as the constraint defined by the keep-out cones, and the error of the final time is always about 3%. The performances are the same reported in Table 6, with errors sharing the same order of magnitude about the maneuver time t_f.

The strength of the IPSO lies in the fact that a near-optimal solution can be found for all geometries without requiring different computational times. As a final remark, it is important to underline that the IPSO solution is able to fully satisfy all the boundary and path constraints offering a maneuver time slightly greater than the optimal one. If the constraint about the optimal minimum time is relaxed and a near-optimal solution consistent with the constraints is accepted, then the IPSO solution is favorable considering the computational time required. Moreover, although no differences have been reported when analyzing cases 1 and 2 applying the IPSO approach, using a POCS, case 2 is more difficult to solve than case 1 because it requires greater computational times to obtain a positive solution.

C. Inverse Particle Swarm Optimization Solution Used as Best Guess

If the computation of the exact optimal solution is required, the IPSO algorithm alone is not sufficient because it only gives a near-optimal solution. As mentioned in the previous section, the exact solution may be obtained using a POCS, which often requires high computational times. Different from the proposed PSO algorithms, which require always the same computational time regardless of the geometry, the POCS may significantly change in behavior depending on the studied geometry. To prevent such a problem, a hybrid method is proposed (such a strategy has already been proposed in [21]); a suboptimal solution produced by IPSO is given as best guess to the POCS, which may now obtain the optimal solution in very low computational times. In this case, in fact, the POCS only needs to improve the IPSO solution, which is closer to the optimum. Furthermore, the DPSO may be used as the initial guess obtaining the same improvement in the computational time required by the POCS. However, the IPSO minimizes the total computational time, requiring less time to evaluate the suboptimal solution than the DPSO. When no initial guess is given, the POCS creates the initial solution as a linear interpolation between the initial and the final values both for the state and the control.

It is important to underline that the behavior of the pseudospectral optimization software is strongly related to the required tolerance at the mesh points. When a bang–bang solution is the optimal solution, a high value of the tolerance is needed, and the computational time is quite high. For the reported experiments, the mesh tolerance is set to 10^{-11}. Obviously, when higher tolerances are required, the hybrid method is even more useful.

With regard to case 1, the optimal solution is shown in Figs. 9–11, where the behavior of angular displacements, angular velocities, and control are reported. As far as the computational time is concerned, Table 7 reports the computational time required by the hybrid method. The total required time is, on average, 225.12 s; when using the POCS without the best guess, the computational time is on average 351.25 s. As it can be seen, the hybrid method leads to a time savings of 35.91%. The advantage may be also seen in the mesh refinements and the total variables needed to obtain the optimal solution; their value is always higher when a best guess is not given.

Table 6 List of 10 different results obtained with IPSO for case 1

T_{IPSO}, s	t_f, s	ϵ, %
107.60	232.23	3.06
107.15	232.03	2.97
107.85	232.09	3.00
108.53	232.12	3.01
108.09	231.93	2.93
108.48	232.45	3.16
107.62	231.92	2.92
108.15	233.11	3.45
108.11	232.33	3.11
108.10	232.85	3.34

Table 7 Results of the hybrid method for 10 different experiments, case 1

	T_{PSO}, s	$t_{f,\mathrm{PSO}}$, s	T_{POCS}, s	$t_{f,\mathrm{POCS}}$, s	T_{TOT}, s
1	107.60	232.23	178.17	225.33	285.77
2	107.15	232.03	61.23	225.33	168.38
3	107.85	232.09	203.68	225.33	311.53
4	108.53	232.12	67.78	225.33	176.31
5	108.09	231.93	87.86	225.33	195.95
6	108.48	232.45	75.76	225.33	184.24
7	107.62	231.92	107.78	225.33	215.40
8	108.15	233.11	88.60	225.33	196.74
9	108.11	232.33	93.31	225.33	201.42
10	108.10	232.85	207.39	225.33	315.49

Table 8 Direction of the exclusion cones in $\mathbf{BRF_0}$ for the case 3

Case 3: roll angle $\Theta_f = 135$ deg	Cone 1	Cone 2
x	−0.18	0.95
y	0.56	0
z	0.81	0.31

Table 9 Results of the hybrid method for 10 different experiments, case 3

T_{PSO}, s	$t_{f,\mathrm{PSO}}$, s	T_{POCS}, s	$t_{f,\mathrm{POCS}}$, s	T_{TOT}, s
99.32	443.82	150.06	404.71	249.38
100.42	452.61	141.63	404.71	242.05
99.61	447.78	175.41	404.71	275.01
99.79	449.83	346.00	404.71	445.79
96.50	443.39	252.11	404.71	348.61
96.49	453.79	234.74	404.71	331.22
100.22	443.63	259.96	404.71	360.18
105.58	446.79	188.97	404.71	294.56
99.75	456.66	151.34	404.71	251.09
100.20	453.49	197.74	404.71	297.94

For the tests with the third geometry (Table 8), the computational time required by the hybrid method is on average 323.77 s, as reported in Table 9. When no best guess is used, the average computational time required by the POCS is 1522.57 s, which is almost five times the computational effort of the hybrid approach. This test has been carried out using the same satellite but with different positions of the sensor axis and different exclusion cones.

It may be important to underline that if, on one hand, the pseudospectral algorithm provides the optimal solution, on the other hand, it may require greater computational times. Sometimes, as for case 2, the optimal solution may be obtained with only very high tolerances.

VI. Conclusions

We have demonstrated that the particle swarm optimization may be used for planning suboptimal constrained maneuvers. When the proposed inverse method is used, the boundary and the path constraints are fully satisfied. The final time obtained is always about 3% greater than the time, computed using a pseudospectral method. Moreover, it was underlined that both the results and the computational time do not change considering the different geometries of the keep-out cones or different characteristics of the maneuver. On the contrary, the calculation of the solution through an optimization solver such as a pseudospectral optimization software may require very high computational times. In our tests, the computational time required by the inverse method is up to $1/15$ of the time required by the pseudospectral optimization software without initial guess. The proposed approach is suitable in the perspective of achieving fully autonomous satellites.

Appendix: Notes on B-Splines

B-spline stands for basis spline. The term spline means, as usual, a piecewise polynomial function. Different from the splines, however, interpolation with B-splines is based on the fact that each segment of the interpolating curve is built upon various polynomials rather than on only one polynomial. Useful explanations may be find in [45–48]. A B-spline curve is defined as

$$\boldsymbol{Y}(\tau) = \sum_{i=0}^{n} N_{i,k}(\tau)\boldsymbol{U}_i \tag{A1}$$

where $\boldsymbol{U}_i$ is a control point, $n+1$ is the number of the control points, and i is the index of the control points. The degree of the interpolant polynomial is $k-1$.

Given a knot vector $\boldsymbol{T}$,

$$\boldsymbol{T} = \{t_0 = 0 \le t_1, \ldots, t_{n+k-1} \le t_{n+k} = 1\} \tag{A2}$$

and a strictly increasing vector $\boldsymbol{\tau} = (0, \ldots, 1)'$, the associated B-spline basis functions $N_{i,\kappa}(t)$ are defined through the Cox–de Boor recursion formula as

$$N_{i,1}(\tau) = \begin{cases} 1, & \text{for } t_i \le \tau < t_{i+1} \\ 0, & \text{otherwise} \end{cases} \tag{A3}$$

for $\kappa = 1$, whereas for $1 < \kappa \le k$,

$$N_{i,\kappa}(\tau) = \frac{\tau - t_i}{t_{i+\kappa-1} - t_i} N_{i,\kappa-1}(\tau) + \frac{t_{i+\kappa} - \tau}{t_{i+\kappa} - t_{i+1}} N_{i+1,\kappa-1}(\tau) \tag{A4}$$

In Eq. (A3), $i = 0, 1, \ldots, n+k-1$. In Eq. (A4), $i = 0, 1, \ldots, n+k-\kappa$.

A clamped B-spline is obtained when the curve passes through the first and the final control points; it can be obtained using nonuniform knot points:

$$\begin{aligned} t_i &= 0 && \text{if } 0 \le i < k \\ t_i &= \frac{i-k+1}{n-k+2} && \text{if } k \le i \le n \\ t_i &= 1 && \text{if } n < i \le n+k \end{aligned}$$

References

[1] Bilimoria, K. D., and Wie, B., "Time-Optimal Three-Axis Reorientation of Rigid Spacecraft," *Journal of Guidance, Control, and Dynamics*, Vol. 16, No. 3, 1993, pp. 446–452. doi:10.2514/3.21030

[2] Li, F., and Bainum, P. M., "Numerical Approach for Solving Rigid Spacecraft Minimum Time Attitude Maneuvers," *Journal of Guidance, Control, and Dynamics*, Vol. 13, No. 1, 1990, pp. 38–45. doi:10.2514/3.20515

[3] Scrivener, S. L., and Thompson, R. C., "Survey of Time-Optimal Attitude Maneuvers," *Journal of Guidance, Control, and Dynamics*, Vol. 17, No. 2, 1994, pp. 225–233. doi:10.2514/3.21187

[4] Byers, R. M., and Vadali, S. R., "Quasi-Closed-Form Solution to the Time-Optimal Rigid Spacecraft Reorientation Problem," *Journal of Guidance, Control, and Dynamics*, Vol. 16, No. 3, 1993, pp. 453–461. doi:10.2514/3.21031

[5] Shen, H., and Tsiotras, P., "Time-Optimal Control of Axisymmetric Rigid Spacecraft Using Two Controls," *Journal of Guidance, Control, and Dynamics*, Vol. 22, No. 5, Sept.–Oct. 1999, pp. 682–694. doi:10.2514/2.4436

[6] Fleming, A., Sekhavat, P., and Ross, I. M., "Minimum-Time Reorientation of a Rigid Body," *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 1, Jan.–Feb. 2010, pp. 160–170. doi:10.2514/1.43549

[7] Liu, S. W., and Singh, T., "Fuel/Time Optimal Control of Spacecraft Maneuvers," *Journal of Guidance, Control, and Dynamics*, Vol. 20, No. 2, 1997, pp. 394–397. doi:10.2514/2.4053

[8] Li, J., and Xi, X., "Fuel-Optimal Low-Thrust Reconfiguration for Formation-Flying Satellites via Homotopic Approach," *Journal of Guidance, Control, and Dynamics*, Vol. 35, No. 6, 2012, pp. 1709–1717. doi:10.2514/1.57354

[9] Li, J., and Xi, X., "Time-Optimal Reorientation of the Rigid Spacecraft Using a Pseudospectral Method Integrated Homotopic Approach," *Optimal Control Applications and Methods*, Oct. 2014. doi:10.1002/oca.2145

[10] Yutko, B. M., and Melton, R. G., "Optimizing Spacecraft Reorientation Maneuvers Using a Pseudospectral Method," *Journal of Aerospace Engineering, Sciences and Applications*, Vol. 2, No. 1, Jan.–April 2010, pp. 1–14. doi:10.7446/jaesa.0201.01

[11] Bai, X., and Junkins, J. L., "New Results for Time-Optimal Three-Axis Reorientation of a Rigid Spacecraft," *Journal of Guidance, Control, and Dynamics*, Vol. 32, No. 4, July–Aug. 2009, pp. 1071–1076. doi:10.2514/1.43097

[12] McInnes, C. R., "Large Angle Slew Maneuvers with Autonomous Sun Vector Avoidance," *Journal of Guidance, Control, and Dynamics*, Vol. 17, No. 4, 1994, pp. 875–877. doi:10.2514/3.21283

[13] Koenig, J. D., "A Novel Attitude Guidance Algorithm for Exclusion Zone Avoidance," *Proceedings of the IEEE Aerospace Conference*, IEEE Publ., Piscataway, NJ, March 2009, pp. 1–10. doi:10.1109/AERO.2009.4839538

[14] Hablani, H. B., "Attitude Commands Avoiding Bright Objects and Maintaining Communication with Ground Station," *Journal of Guidance, Control, and Dynamics*, Vol. 22, No. 6, Nov.–Dec. 1999, pp. 759–767. doi:10.2514/2.4469

[15] Boldrini, F., Procopio, D., Airy, S. P., and Giulicchi, L., "Miniaturised Star Tracker (AA-STR) Ready to Fly," *The 4S Symposium: Small Satellites, Systems and Services (ESA SP-571)* [CD-ROM], edited by Warmbein, B., Noordwijk, The Netherlands, Sept. 2004.

[16] Schmidt, U., Fiksel, T., Kwiatkowski, A., Steinbach, I., Pradarutti, B., Michel, K., and Benzi, E., "Autonomous Star Sensor ASTRO APS: Flight Experience on Alphasat," *CEAS Space Journal*, Vol. 7, No. 2, June 2015, pp. 237–246. doi:10.1007/s12567-014-0071-z

[17] Melton, R. G., "Numerical Analysis of Constrained, Time-Optimal Satellite Reorientation," *Mathematical Problems in Engineering*, Vol. 2012, pp. 1–19. doi:10.1155/2012/769376

[18] Frazzoli, E., Dahleh, M. A., Feron, E., and Kornfel, R. P., "A Randomized Attitude Slew Planning Algorithm for Autonomous Spacecraft," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, AIAA Paper 2001-4155, Aug. 2001.

[19] Lee, U., and Mesbahi, M., "Spacecraft Reorientation in Presence of Attitude Constraints via Logarithmic Barrier Potentials," *Proceedings of the American Control Conference*, IEEE Publ., Piscataway, NJ, June–July 2011, pp. 450–455. doi:10.1109/ACC.2011.5991284

[20] Lee, T., Leok, M., and McClamroch, N. H., "Time Optimal Attitude Control for a Rigid Body," *Proceedings of the American Control Conference*, IEEE Publ., Piscataway, NJ, June 2008, pp. 5210–5215. doi:10.1109/ACC.2008.4587322

[21] Cui, P., Zhong, W., and Cui, H., "Onboard Spacecraft Slew-Planning by Heuristic State-Space Search and Optimization," *Proceedings of the International Conference on Mechatronics and Automation*, IEEE Publ., Piscataway, NJ, Aug. 2007, pp. 2115–2119. doi:10.1109/ICMA.2007.4303878

[22] Lai, L.-C., Yang, C.-C., and Wu, C.-J., "Time-Optimal Maneuvering Control of a Rigid Spacecraft," *Acta Astronautica*, Vol. 60, Nos. 10–11, May–June 2007, pp. 791–800. doi:10.1016/j.actaastro.2006.09.039

[23] Yang, X.-S., "Engineering Optimization—An Introduction with Metaheuristic Applications," Wiley, Hoboken, NJ, 2010, Chaps. 1, 2, 15.

[24] Melton, R. G., "Hybrid Methods for Determining Time-Optimal, Constrained Spacecraft Reorientation Maneuvers," *Acta Astronautica*, Vol. 94, No. 1, Jan.–Feb. 2014, pp. 294–301. doi:10.1016/j.actaastro.2013.05.007

[25] Melton, R. G., "Maximum-Likelihood Estimation Optimizer for Constrained, Time-Optimal Satellite Reorientation," *Acta Astronautica*, Vol. 103, Oct.–Nov. 2014, pp. 185–192. doi:10.1016/j.actaastro.2014.06.032

[26] Komfeld, R. P., "On-Board Autonomous Attitude Maneuver Planning for Planetary Spacecraft Using Genetic Algorithms," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, AIAA Paper 2003-5784, Aug. 2003.

[27] Siliang, Y., and Shijie, X., "Spacecraft Attitude Maneuver Planning Based on Particle Swarm Optimization," *Journal of Beijing University of Aeronautics and Astronautics*, Vol. 2010, No. 1, 2010, pp. 48–51.

[28] Showalter, D. J., and Black, J. T., "Responsive Theater Maneuvers via Particle Swarm Optimization," *Journal of Spacecraft and Rockets*, Vol. 51, No. 6, Nov.–Dec. 2014, pp. 1976–1985. doi:10.2514/1.A32989

[29] Xu, W., Li, C., Wang, X., Liu, Y., Liang, B., and Xu, Y., "Study on Non-Holonomic Cartesian Path Planning of Free-Floating Space Robotic System," *Advanced Robotics*, Vol. 23, Nos. 1–2, 2009, pp. 113–143. doi:10.1163/156855308X392708

[30] Huang, P., Liu, G., Yuan, J., and Xu, Y., "Multi-Objective Optimal Trajectory Planning of Space Robot Using Particle Swarm Optimization," *Advances in Neural Networks — ISNN 2008*, Vol. 5264, Lecture Notes in Computer Science, Springer, Berlin, 2008, pp. 171–179. doi:10.1007/978-3-540-87734-9-20

[31] Rahimi, A., Dev Kumar, K., and Alighanbari, H., "Particle Swarm Optimization Applied to Spacecraft Reentry Trajectory," *Journal of Guidance, Control, and Dynamics*, Vol. 36, No. 1, Jan.–Feb. 2013, pp. 307–310. doi:10.2514/1.56387

[32] Chen, S.-M., and Dong, Y.-F., "Satellite Attitude Tracking Controller Optimization Based on Particle Swarm Optimization," *Procedia Engineering*, Vol. 15, 2011, pp. 526–530. doi:10.1016/j.proeng.2011.08.100

[33] Xia, N., Han, D., Zhang, G. F., Jiang, J. G., and Vu, K., "Study on Attitude Determination Based on Discrete Particle Swarm Optimization," *Science China Technological Sciences*, Vol. 53, No. 12, Dec. 2010, pp. 3397–3403. doi:10.1007/s11431-010-4148-4

[34] Kennedy, J., and Eberhart, R., "Particle Swarm Optimization," *Proceedings of the IEEE International Conference on Neural Networks*, Vol. 4, IEEE Publ., Piscataway, NJ, 1995, pp. 1942–1948. doi:10.1109/ICNN.1995.488968

[35] Clerc, M., *Particle Swarm Optimization*, ISTE, Feb. 2006, pp. 17–50.

[36] Kennedy, J., "The Particle Swarm: Social Adaptation of Knowledge," *Proceedings of the IEEE International Conference on Evolutionary Computation*, IEEE Publ., Piscataway, NJ, April 1997, pp. 303–308. doi:10.1109/ICEC.1997.592326

[37] Parsopoulos, K. E., and Vrahatis, M. N., "Parameter Selection and Adaptation in Unified Particle Swarm Optimization," *Mathematical and Computer Modelling*, Vol. 46, Nos. 1–2, July 2007, pp. 198–213. doi:10.1016/j.mcm.2006.12.019

[38] Kennedy, J., and Eberhart, R., "A New Optimizer Using Particle Swarm Theory," *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, IEEE Publ., Piscataway, NJ, Oct. 1995, pp. 39–43. doi:10.1109/MHS.1995.494215

[39] Shi, Y., and Eberhart, R., "A Modified Particle Swarm Optimizer," *Evolutionary Computation Proceedings, IEEE World Congress on Computational Intelligence*, IEEE Publ., Piscataway, NJ, 1998, pp. 69–73. doi:10.1109/ICEC.1998.699146

[40] Eberhart, R., and Yuhui, S., "Particle Swarm Optimization: Developments, Applications and Resources," *Proceedings of the 2001 Congress on Evolutionary Computation*, Vol. 1, IEEE Publ., Piscataway, NJ, May 2001, pp. 81–86. doi:10.1109/CEC.2001.934374

[41] Shuster, M. D., "A Survey of Attitude Representations," *Journal of the Astronautical Sciences*, Vol. 41, No. 4, Oct.–Dec. 1993, pp. 439–517.

[42] Kuri-Morales, A. F., and Gutiérrez-Garcia, J., "Penalty Function Methods for Constrained Optimization with Genetic Algorithms: A Statistical Analysis," *MICAI 2002: Advances in Artificial Intelligence*, Vol. 2313, 2002, pp. 108–117. doi:10.1007/3-540-46016-0-12

[43] Gen, M., and Cheng, R., "A Survey of Penalty Techniques in Genetic Algorithms," *Proceedings of IEEE International Conference on Evolutionary Computation*, IEEE Publ., Piscataway, NJ, May 1996, pp. 804–809. doi:10.1109/ICEC.1996.542704

[44] Yeniay, Ö., "Penalty Function Methods for Constrained Optimization with Genetic Algorithms," *Mathematical and Computational Applications*, Vol. 10, No. 1, 2005, pp. 45–56.

[45] De Boor, C., *A Practical Guide to Splines (Applied Mathematical Sciences)*, Springer, New York, Dec. 2001, Chaps. 9–11.

[46] De Boor, C., "On Calculating with B-Splines," *Journal of Approximation Theory*, Vol. 6, No. 1, 1972, pp. 50–62. doi:10.1016/0021-9045(72)90080-9

[47] Lorentz, G. G., Chui, C. K., and Schumaker, L. L., "Approximation Theory II," Academic Press, New York, 1976, pp. 1–47.

[48] Cox, M. G., "Practical Spline Approximation," *Topics in Numerical Analysis*, Vol. 965, Springer, Berlin, 1982, pp. 79–112. doi:10.1007/BFb0063201

[49] Rao, A. V., Benson, D. A., Darby, C. L., Patterson, M. A., Francolin, C., Sanders, I., and Huntington, G. T., "Algorithm 902: GPOPS, A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using the Gauss Pseudospectral Method," *ACM Transactions on Mathematical Software*, Vol. 37, No. 2, April–June 2010, p. 39. doi:10.1145/1731022.1731032

[50] Benson, D. A., Huntington, G. T., Thorvaldsen, T. P., and Rao, A. V., "Direct Trajectory Optimization and Costate Estimation via an Orthogonal Collocation Method," *Journal of Guidance, Control, and Dynamics*, Vol. 29, No. 6, Nov.–Dec. 2006, pp. 1435–1440. doi:10.2514/1.20478

[51] Garg, D., Patterson, M. A., Darby, C. L., Francolin, C., Huntington, G. T., Hager, W. W., and Rao, A. V., "Direct Trajectory Optimization and Costate Estimation of Finite-Horizon and Infinite-Horizon Optimal Control Problems Using a Radau Pseudospectral Method," *Computational Optimization and Applications*, Vol. 49, No. 2, June 2011, pp. 335–358. doi:10.1007/s10589-009-9291-0

[52] Garg, D., Patterson, M. A., Hager, W. W., Rao, A. V., Benson, D. A., and Huntington, G. T., "A Unified Framework for the Numerical Solution of Optimal Control Problems Using Pseudospectral Methods," *Automatica*, Vol. 46, No. 11, Nov. 2010, pp. 1843–1851. doi:10.1016/j.automatica.2010.06.048

[53] Garg, D., Hager, W. W., and Rao, A. V., "Pseudospectral Methods for Solving Infinite-Horizon Optimal Control Problems," *Automatica*, Vol. 47, No. 4, April 2011, pp. 829–837. doi:10.1016/j.automatica.2011.01.085