Acta Astronautica 94 (2014) 690–698

Contents lists available at ScienceDirect

Acta Astronautica

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

Time-optimal trajectory planning for underactuated spacecraft using a hybrid particle swarm optimization algorithm

Yufei Zhuang *, Haibin Huang

School of Information and Electrical Engineering, Harbin Institute of Technology at Weihai, Weihai 264200, China

ARTICLE INFO

Article history:
Received 11 June 2012
Received in revised form
15 June 2013
Accepted 18 June 2013
Available online 24 June 2013

Keywords:
Time-optimal
Trajectory planning
Particle swarm optimization
Nonlinear programming
Pseudospectral method
Underactuated spacecraft

ABSTRACT

A hybrid algorithm combining particle swarm optimization (PSO) algorithm with the Legendre pseudospectral method (LPM) is proposed for solving time-optimal trajectory planning problem of underactuated spacecrafts. At the beginning phase of the searching process, an initialization generator is constructed by the PSO algorithm due to its strong global searching ability and robustness to random initial values, however, PSO algorithm has a disadvantage that its convergence rate around the global optimum is slow. Then, when the change in fitness function is smaller than a predefined value, the searching algorithm is switched to the LPM to accelerate the searching process. Thus, with the obtained solutions by the PSO algorithm as a set of proper initial guesses, the hybrid algorithm can find a global optimum more quickly and accurately. 200 Monte Carlo simulations results demonstrate that the proposed hybrid PSO–LPM algorithm has greater advantages in terms of global searching capability and convergence rate than both single PSO algorithm and LPM algorithm. Moreover, the PSO–LPM algorithm is also robust to random initial values.

1. Introduction

The time-optimal trajectory planning problem of rigid spacecrafts has received consistent interest in last decades, since rapid attitude maneuvers are critical to various space missions such as military observation and satellite communication [1]. To solve this problem using classic optimal theory, the Hamilton–Jacobi–Bellman (HJB) partial differential equations must be solved. However, due to the complicated nonlinear dynamics of high dimension and many types of constraints for rigid spacecraft, the HJB equations are very difficult to be solved. Thus, numerical methods base on gradient information have been widely used [2–4]. There are two distinct branches of numerical methods: indirect and direct [5], both of which attempt to minimize the functions and constraint violations using discrete approximations of the parameters of the system. Compared to classic optimal theory, numerical methods have advantages in terms of flexible applicability to practical complex problems and meanwhile guaranteeing a relatively quick convergence rate and very accurate results. However, two key drawbacks inherent to both indirect and direct methods are: (1) the lack of a global search capability; and (2) the requirement of suitable initial guesses [6]. A set of poor initial guesses may cause the optimization program trapped at a local optimum in multimodal problems or even diverge.

To deal with this problem, evolutionary algorithms (EAs) such as genetic algorithms (GA), differential evolution (DE) and particle swarm optimization (PSO) have been employed. These algorithms usually have better ability to converge to a global optimum or a near optimum solution than traditional optimization methods, and moreover, they are not sensitive to initial guesses of solutions. Thus, EAs have been successfully applied in many real world nonlinear optimal problems [6–9]. However, as we know, EAs are characterized by their poor numerical accuracy and difficult constraint handling.

In recent years, in order to overcome these shortages of both numerical methods and EAs, a series of hybrid optimization algorithms have been proposed [10–14]. The fundamental idea of these algorithms is that first an EA with random initial solutions is utilized to enhance the

* Corresponding author. Tel.: +86 6315 687548.
E-mail addresses: yufeizhuang9@gmail.com (Y. Zhuang), hhb833@gmail.com (H. Huang).

0094-5765/$ - see front matter © 2013 IAA. Published by Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.actaastro.2013.06.023

global search capability. When the change in fitness value is smaller than a predefined value, or the candidate solution of GA or the particles in swarm being close to the global optimum [13–15], the evolution process is stopped generating. Then, the searching algorithm is switched to a direct method, which can achieve a faster convergence rate around the global optimum and a higher accuracy than the EA. And, the obtained near-optimal solution by EA will be taken as an initial guess of solutions for the subsequent nonlinear programming (NLP) solver. In this way the hybrid algorithm may find a global optimal solution more quickly and accurately.

Other than the literature on hybrid optimization algorithms with GA as a starter engine [13,14], in this paper, a new hybrid algorithm is proposed which combines the PSO algorithm and the Legendre pseudospectral method (LPM) for the time-optimal trajectory planning problem of an underactuated spacecraft. Recently, PSO algorithm has been found to be a promising technique for a variety of optimization problems due to its superior advantages. PSO is initialized with a population of random solutions and searches for the optimum by updating generations. And compared to GA and DE, PSO has no operators such as crossover, mutation and selection, and it has fewer parameters to adjust. Therefore, PSO is quite easy to implement and it also can be applied in many areas to replace GA. In this present work LPM is used as a discretization scheme, and PSO is used to find an initial starting point being within the domain of convergence of the particular NLP solvers. And further, the differential flatness property of the spacecraft is also discussed, which can reduce the number of variables to be optimized in order to decrease the time consumption of the whole optimization process.

The rest of the paper is organized as follows. Section 2 describes the formulation of the trajectory optimization problem. Section 3 discusses the flatness property of a general nonlinear system. In Section 4 the hybrid PSO-LPM algorithm is introduced. Section 5 applies the proposed algorithm to a minimum-time trajectory planning problem for a rigid spacecraft under actuator failure condition, and presents the simulation results. Finally, conclusion is contained in Section 6.

2. Problem formulation

The purpose of trajectory optimization is to determine the time history of a control vector $\boldsymbol{u}(\tau), \tau\in[\tau_0, \tau_f]$ with considering the constraints, in order to minimize the cost function J. Typically, J can be written in Bolza form consisting a Mayer term Φ and a Lagrange term L [5]:

$$J=\Phi(\boldsymbol{x}(\tau_f),\tau_f)+\int_{\tau_0}^{\tau_f} L(\boldsymbol{x}(\tau),\boldsymbol{u}(\tau),\tau)\mathrm{d}\tau \tag{1}$$

Subject to

(1) State constraints
$$\dot{\boldsymbol{x}}(\tau)=\boldsymbol{f}(\boldsymbol{x}(\tau),\boldsymbol{u}(\tau),\tau) \tag{2}$$

(2) Endpoint constraints with upper and lower bounds on function $\boldsymbol{\psi}[\cdot]$
$$\boldsymbol{\psi}_{0l}\le\boldsymbol{\psi}(\boldsymbol{x}(\tau_0),\boldsymbol{u}(\tau_0),\tau_0)\le\boldsymbol{\psi}_{0u},\quad \boldsymbol{\psi}_{fl}\le\boldsymbol{\psi}(\boldsymbol{x}(\tau_f),\boldsymbol{u}(\tau_f),\tau_f)\le\boldsymbol{\psi}_{fu} \tag{3}$$

(3) Path constraints with upper and lower bounds on function $\boldsymbol{g}[\cdot]$
$$\boldsymbol{g}_l\le\boldsymbol{g}(\boldsymbol{x}(\tau),\ \boldsymbol{u}(\tau),\tau)\le\boldsymbol{g}_u \tag{4}$$

(4) Box constraints on states and controls
$$\boldsymbol{x}_l\le\boldsymbol{x}(\tau)\le\boldsymbol{x}_u,\quad \boldsymbol{u}_l\le\boldsymbol{u}(\tau)\le\boldsymbol{u}_u \tag{5}$$

In Eqs. (1)–(5), $\boldsymbol{x}(\tau)\in\mathbf{R}^n$ is the state vector of the system, and $\boldsymbol{u}(\tau)\in\mathbf{R}^m$ is the control vector.

3. Problem conversion

3.1. Notation of differential flatness

A dynamic system

$$\dot{\boldsymbol{x}}=\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})\quad \boldsymbol{x}\in\mathbf{R}^n,\ \boldsymbol{u}\in\mathbf{R}^m \tag{6}$$

is differentially flat or just flat, if there exist smooth maps $\boldsymbol{Y}$, $\boldsymbol{A}$ and $\boldsymbol{B}$ defining on open neighborhoods of $\mathbf{R}^n\times(\mathbf{R}^m)^{\rho+1}$, $(\mathbf{R}^m)^{r+1}$ and $(\mathbf{R}^m)^{r+2}$, such that

$$\begin{aligned}\boldsymbol{y}&=\boldsymbol{Y}(\boldsymbol{x},\boldsymbol{u},\dot{\boldsymbol{u}},\ddot{\boldsymbol{u}},...,\boldsymbol{u}^{(\rho)})\\ \boldsymbol{x}&=\boldsymbol{A}(\boldsymbol{y},\dot{\boldsymbol{y}},\ddot{\boldsymbol{y}},...,\boldsymbol{y}^{(r)})\\ \boldsymbol{u}&=\boldsymbol{B}(\boldsymbol{y},\dot{\boldsymbol{y}},\ddot{\boldsymbol{y}},...,\boldsymbol{y}^{(r+1)})\end{aligned} \tag{7}$$

here ρ and r are positive integers, $\boldsymbol{y}$ is called a set of flat outputs, and the components of $\boldsymbol{y}$ are not related by a differential relation [16].

3.2. Problem reformulation in flat output space

From the notation of flatness, it appears that if a dynamic system is flat, then its state and input variables can be parameterized in terms of the set of flat outputs and a finite number of their derivatives. Thus, the above original optimal problem can be converted and reformulated in flat output space as

$$\begin{aligned}&\min_{\tilde{\boldsymbol{y}}(\tau)}\tilde{J}(\tilde{\boldsymbol{y}}(\tau),\tau)=\tilde{\Phi}(\boldsymbol{A}(\tilde{\boldsymbol{y}}(\tau_f)),\tau_f)+\int_{\tau_0}^{\tau_f}\tilde{L}(\boldsymbol{A}(\tilde{\boldsymbol{y}}(\tau)),\boldsymbol{B}(\tilde{\boldsymbol{y}}(\tau)),\tau)\mathrm{d}\tau\\ \text{s.t.}\quad&\boldsymbol{\psi}_{0l}\le\tilde{\boldsymbol{\psi}}(\boldsymbol{A}(\tilde{\boldsymbol{y}}(\tau_0)),\boldsymbol{B}(\tilde{\boldsymbol{y}}(\tau_0)),\tau_0)\le\boldsymbol{\psi}_{0u}\\ &\boldsymbol{\psi}_{fl}\le\tilde{\boldsymbol{\psi}}(\boldsymbol{A}(\tilde{\boldsymbol{y}}(\tau_f)),\boldsymbol{B}(\tilde{\boldsymbol{y}}(\tau_f)),\tau_f)\le\boldsymbol{\psi}_{fu}\\ &\boldsymbol{g}_l\le\tilde{\boldsymbol{g}}(\boldsymbol{A}(\tilde{\boldsymbol{y}}(\tau)),\boldsymbol{B}(\tilde{\boldsymbol{y}}(\tau)),\tau)\le\boldsymbol{g}_u\\ &\boldsymbol{x}_l\le\boldsymbol{A}(\tilde{\boldsymbol{y}}(\tau))\le\boldsymbol{x}_u,\quad \boldsymbol{u}_l\le\boldsymbol{B}(\tilde{\boldsymbol{y}}(\tau))\le\boldsymbol{u}_u\quad \tau\in[\tau_0,\tau_f]\end{aligned} \tag{8}$$

where, $\tilde{\boldsymbol{y}}(\tau)=[\boldsymbol{y}(\tau),\dot{\boldsymbol{y}}(\tau),\ddot{\boldsymbol{y}}(\tau),...,\boldsymbol{y}^{(r+1)}(\tau)]^{\mathrm{T}}$. Differential flatness is an excellent property, especially for the hybrid PSO–LSM algorithm proposed in this paper. First, by the flat conversion the state equality constraints which are quite difficult to handle for PSO algorithm can be entirely eliminated as in Eq. (8). Thus, a feasible initial solution within the convergence domain of the NLP solver is probably obtained. Second, usually the number of the variables to be optimized can be dramatically decreased in this reformulated problem. Therefore, the following NLP

process with a direct method can be simplified and the time consumption can also be reduced.

4. Hybrid PSO–LPM algorithm for optimal control

4.1. Particle swarm optimization (PSO)

4.1.1. Standard PSO

Particle swarm optimization (PSO) is an evolutionary computation technique, which was introduced in the mid 1990s [17]. The standard PSO algorithm consists of three steps: generating positions and velocities of particles, velocity update and position update. And the outline of a standard PSO algorithm can be illustrated as follows:

Step 1: Initialize a set of particles positions $\boldsymbol{X}_0^i$ and velocities $\boldsymbol{V}_0^i$ randomly, and set the upper bound $\boldsymbol{X}_{\max}$ and lower bound $\boldsymbol{X}_{\min}$ of the design variables.
Step 2: Evaluate the optimization fitness function; update the best ever position of the ith particle $\boldsymbol{P}_k^i$ at iteration k, and the best global position $\boldsymbol{P}_k^g$ in the swarm up to iteration k; update the velocity vector of each particle in the swarm.
Step 3: Update the position vector of each particle, using its previous position and the updated velocity vector.
Step 4: Go to Step 2 until the stopping criteria is met.
In Step 2, the updating scheme for the velocity vector of each particle is

$$\boldsymbol{V}_{k+1}^i = w\boldsymbol{V}_k^i + c_1 r_1(\boldsymbol{P}_k^i - \boldsymbol{X}_k^i) + c_2 r_2(\boldsymbol{P}_k^g - \boldsymbol{X}_k^i) \tag{9}$$

where, subscript k indicates an unit pseudo-time increment, $\boldsymbol{V}_k^i, \boldsymbol{X}_k^i$ are the velocity vector and position vector of particle i at iteration k, r_1 and r_2 are two random numbers in the range $[0,1]$. c_1, c_2 and w are three problem-dependent parameters, c_1 indicates how much confidence the current particle has in itself and c_2 indicates how much confidence it has in the swarm and w is a inertia weighting factor which controls the exploration abilities of the swarm.

In Step 3, the updating scheme for the position vector of each particle is

$$\boldsymbol{X}_{k+1}^i = \boldsymbol{X}_k^i + \boldsymbol{V}_{k+1}^i \tag{10}$$

4.1.2. Algorithm improvements

In this section, a number of improvements to the standard PSO algorithm are investigated, such that the algorithm would be more general in nature and applicable to a wide range of optimization problems.

Since the inertia weight is important in the global/local exploration behavior of the PSO algorithm, a linearly decreasing w with each iteration is proposed as follows:

$$w_{k+1} = w_{\max} - \frac{w_{\max} - w_{\min}}{k_{\max}} k \tag{11}$$

where, w_{k+1} is the inertia weight at iteration $k+1$, $w_{\min}$ and $w_{\max}$ are the lower and upper bounds of w in the whole optimization, and $k_{\max}$ is the maximum number of iterations of the PSO algorithm. Thus, in initial exploration a larger value of w facilitate a more global behavior, and then a smaller value of w is utilized for a local behavior in the feasible regions of the design space.

Like other EAs, the standard PSO algorithm is defined for unconstrained optimization problems, however, most trajectory optimization problems are constrained. To handle this problem, a straight forward approach is to convert the constrained problem into a non-constrained problem with a linear penalty function approach as

$$\overline{F}(\boldsymbol{X}) = F(\boldsymbol{X}) + M[\sum_{i=1}^{m} \max(0, G_i(\boldsymbol{X})) + \sum_{j=1}^{n} |H_j(\boldsymbol{X})|] \tag{12}$$

where, $F(\boldsymbol{X})$ is the original cost function, M is a constant and positive penalty coefficient, $G_i (i = 1, 2, ..., m)$ denote m inequality constraints (with violated constraints having values larger than zero) and $H_j (j = 1, 2, ..., n)$ denote n equality constraints, and $\overline{F}(\boldsymbol{X})$ is the new penalized cost function.

When dealing with violated constrained particles by penalty functions in PSO, the velocity vector of a particle with violated constraints should be brought back to zero in the velocity update scheme as

$$\boldsymbol{V}_{k+1}^i = c_1 r_1(\boldsymbol{P}_k^i - \boldsymbol{X}_k^i) + c_2 r_2(\boldsymbol{P}_k^g - \boldsymbol{X}_k^i) \tag{13}$$

This is because if a particle is infeasible, there is a big chance that the last search direction was not feasible.

4.2. Legendre pseudospectral method (LPM)

The key step for a direct method to solve the above continuous optimization problem is to discretize and transcribe Eq. (8) into a NLP problem, thus this section adopts LPM as the discretization scheme [18]. In LPM, we approximate the flat output functions $\boldsymbol{y}(t)$ with Nth order Lagrange polynomials $\boldsymbol{y}^N(t)$ on $N+1$ Legendre–Gauss–Lobatto (LGL) points, as

$$\boldsymbol{y}(t) \approx \boldsymbol{y}^N(t) := \sum_{l=0}^{N} \boldsymbol{y}(t_l)\phi_l(t) = \sum_{l=0}^{N} \lambda_l \phi_l(t) \tag{14}$$

where, $t = [2\tau - (\tau_f + \tau_0)]/(\tau_f - \tau_0) \in [-1, 1]$, LGL points $t_l, l = 0, 1, ..., N (t_0 = -1,\ t_N = 1)$, are the roots of the derivative of the Nth order Legendre polynomials $L_N(t)$, and ϕ_l are the Lagrange interpolating polynomials of order N

$$\phi_l(t) = \frac{1}{N(N+1)L_N(t_l)} \cdot \frac{(t^2-1)\dot{L}_N(t)}{t - t_l} \tag{15}$$

The first and the $(r+1)th$ derivative of $\boldsymbol{y}(t)$ at the LGL point t_k can be approximated respectively as follows:

$$\dot{\boldsymbol{y}}(t_k) \approx \dot{\boldsymbol{y}}^N(t_k) := \sum_{l=0}^{N} \boldsymbol{D}_{1,kl}\boldsymbol{y}(t_l) = \sum_{l=0}^{N} \lambda_l \boldsymbol{D}_{1,kl}$$

$$\boldsymbol{y}^{(r+1)}(t_k) \approx \boldsymbol{y}^{(r+1)N}(t_k) := \sum_{l=0}^{N} \boldsymbol{D}_{(r+1),kl}\boldsymbol{y}(t_l) = \sum_{l=0}^{N} \lambda_l \boldsymbol{D}_{(r+1),kl} \tag{16}$$

where, $\boldsymbol{D}_{1,kl}$ are the entries of the $(N+1) \times (N+1)$

matrix $\boldsymbol{D}$

$$\boldsymbol{D}_1 := [\boldsymbol{D}_{1,kl}] := \begin{cases} \frac{L_N(t_k)}{L_N(t_l)} \cdot \frac{1}{t_k - t_l} & k \neq l \\ -\frac{N(N+1)}{4} & k = l = 0 \\ \frac{N(N+1)}{4} & k = l = N \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

Obviously, $\boldsymbol{D}_{(r+1),kl}$ is also an $(N+1)\times(N+1)$ matrix, which can be easily obtained by $\boldsymbol{D}_{(r+1)} := [\boldsymbol{D}_{(r+1),kl}] = \boldsymbol{D}_1^{(r+1)}$.

And the cost function J is approximated by the Gauss–Lobatto integration rule

$$J = \widehat{\Phi}(\widehat{\boldsymbol{A}}(\lambda_N)) + \frac{\tau_f - \tau_0}{2} \sum_{k=0}^{N} \widehat{L}(\widehat{\boldsymbol{A}}(\lambda_k)), \widehat{\boldsymbol{B}}(\lambda_k)), t_k) w_k \tag{18}$$

where $\lambda = [\lambda_0, \lambda_1, ..., \lambda_N]^T$, and the weight function

$$w_k = \frac{2}{N(N+1)} \frac{1}{[L_N(t_k)]^2},\ k = 0,\ 1, ...,\ N$$

The endpoint constraints, path constraints and box constraints in Eq. (8) can also be discretized by evaluating these inequalities at the LGL points. Thus, the continuous trajectory optimization problem in the flat output space can now be further converted into a NLP as follows:

Determine a set of coefficients $\lambda^* = [\lambda_0^*, \lambda_1^*, ..., \lambda_N^*]^T$, which minimizes the cost function Eq. (18), subject to

$$\begin{aligned} &\boldsymbol{\psi}_{0l} \leq \widehat{\boldsymbol{\psi}}(\widehat{\boldsymbol{A}}(\lambda_0^*), \widehat{\boldsymbol{B}}(\lambda_0^*), t_0) \leq \boldsymbol{\psi}_{0u} \\ &\boldsymbol{\psi}_{fl} \leq \widehat{\boldsymbol{\psi}}(\widehat{\boldsymbol{A}}(\lambda_N^*), \widehat{\boldsymbol{B}}(\lambda_N^*), t_N) \leq \boldsymbol{\psi}_{fu} \\ &\boldsymbol{g}_l \leq \widehat{\boldsymbol{g}}(\widehat{\boldsymbol{A}}(\lambda_k^*), \widehat{\boldsymbol{B}}(\lambda_k^*), t_k) \leq \boldsymbol{g}_u \\ &\boldsymbol{x}_l \leq \widehat{\boldsymbol{A}}(\lambda_k^*) \leq \boldsymbol{x}_u,\ \boldsymbol{u}_l \leq \widehat{\boldsymbol{B}}(\lambda_k^*) \leq \boldsymbol{u}_u \end{aligned} \tag{19}$$

where, $k = 0, 1, ..., N$.

One of the main advantages of LPM is that, it offers an exponential convergence rate for the approximation of analytical functions under L^2 norm, while providing Eulerian-like simplicity [18]. Due to its high accuracy and competitive computational efficiency, LPM is widely used in the direct optimization methods. In general, the LPM has a larger radius of convergence than other numerical methods, and it may not require a good initial guess for convergence. However, an educated initial guess dose improve the convergence rate and robustness [11–14]. Therefore, in the following section the PSO–LMP algorithm is proposed.

4.3. Procedure of hybrid PSO–LPM algorithm

The proposed PSO–LPM is a hybrid optimization algorithm combining the PSO algorithm with LPM algorithm. The main idea of the algorithm is that: first, the PSO algorithm serves as a start engine, which can quickly converge to the approximate region of the global minimum with a randomly generated initial solution as described in Section 4.1.1. However, this algorithm does not maintain its computational efficiency when entering the phase where a refined local search is required to pinpoint the minimum exactly. Thus, in this study after evolving the maximum generation or the change for fitness values is smaller than a predefined value, the PSO algorithm is stopped and replaced by a more computational efficient gradient-based direct method. And the best solution of the PSO algorithm is linearly interpolated on the LGL nodes to initialize the NLP. Then, the NLP phase can refine this result until the provided constraints are satisfied to within the convergence criteria of the NLP solver. Finally, the obtained optimal solutions in flat output space

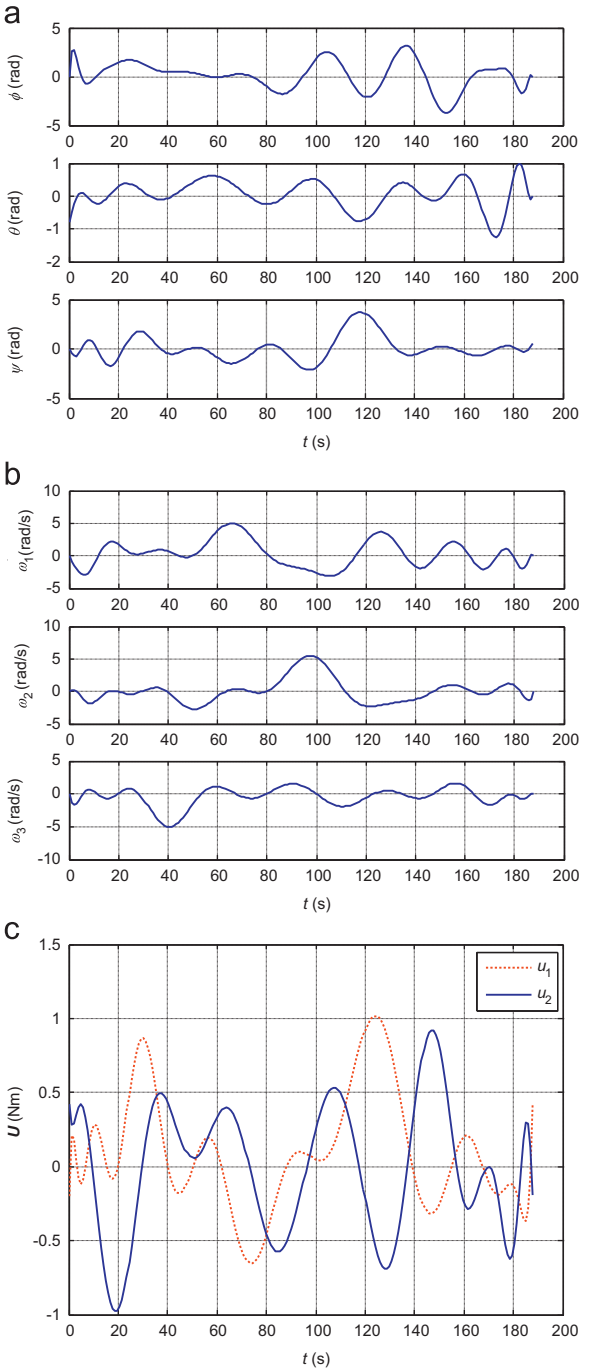

Fig. 1. Initial guess solutions by particle swarm optimization. (a) Initial guess solutions for Euler angles by particle swarm optimization. (b) Initial guess solutions for angular rates by particle swarm optimization.(c) Initial guess solutions for control inputs by particle swarm optimization.

should be mapped back to the state and control input spaces. The procedure of the PSO–LPM algorithm can be summarized as follows:

Step 1: Rewrite the original problem Eqs. (1)–(5) in flat output space as Eq. (14) and approximate the flat output functions by LPM according to Eq. (19).
Step 2: Regard the undetermined NLP variable vector $\lambda^* = [\lambda_0, \lambda_1, ..., \lambda_N]^T$ as a single particle.
Step 3: Given the size of population s, and the maximum number of iterations $K_{\max}$. And, the stopping criteria is the change between the current best particle fitness value and its previous one is smaller than a predefined value κ.
Step 4: Initialize the positions $\boldsymbol{X}_0^i$ and velocities $\boldsymbol{V}_0^i$ of a group of particles randomly.
Step 5: Evaluate each particle's fitness value subject to Eq. (19).
Step 6: If $k > K_{\max}$, the iteration process stops, go to Step 10, else continue.
Step 7: Store the best particle of the current particles. And compare the best fitness value $F(\boldsymbol{P}_k^g)$ with its previous one $F(\boldsymbol{P}_{k-1}^g)$, if the change between the two is smaller than κ, go to Step 10. Else continue.
Step 8: Update the positions and velocities of all the particles according to Eq. (9)–(10).
Step 9: Update the inertia weight according to Eq. (11), and go to Step 5.
Step 10: Switch the search algorithm to a proper NLP algorithm, and consider the best solution obtained by the PSO algorithm as the initial guesses for the NLP algorithm.
Step 11: Obtain the optimal flat output variables $\tilde{\boldsymbol{y}}^*(\tau) = [\boldsymbol{y}^*(\tau), \dot{\boldsymbol{y}}^*(\tau), \ddot{\boldsymbol{y}}^*(\tau), ..., \boldsymbol{y}^{*(r+1)}(\tau)]^T$ by $\lambda^* = [\lambda_0^*, \lambda_1^*, ..., \lambda_N^*]^T$ according to Eq. (14), then map the flat output space to the state and control input spaces by flat transformation.
Step 12: Substitute the optimal control input $\boldsymbol{u}^*$ into the system dynamic equations, and obtain the actual state variable x^N by numerical integral calculations. If the errors between the actual values x^N and the reference values $\boldsymbol{x}^*$ (obtain in Step 11) do not satisfy the precision requirement, then increase the number of LGL points, and go to Step 1, else stops.

5. Simulation results

In this section, the feasibility and effectiveness of the proposed hybrid PSO–LPM algorithm is validated through the applications to an underactuated spacecraft. A rigid spacecraft is underactuated, if the number of independent control inputs is less than its degree of freedom, and this situation always occurs in case of actuators failures. In last decades, the study on underactuated spacecrafts has received much attention both from a practical and theoretic point of view [19–21].

In this paper, we only focus on the two-input underactuated spacecrafts, and assume the rigid body-fixed reference frame along its principle axes of inertia with the origin at the center of mass. Thus, the motion equations of the spacecraft in terms of 3–2–1 sequenced rotation Euler angles, can be described as follows:

$$\begin{aligned}
&I_1\dot{\omega}_1 = (I_2 - I_3)\omega_2\omega_3 + u_1\\
&I_2\dot{\omega}_2 = (I_3 - I_1)\omega_1\omega_3 + u_2\\
&I_3\dot{\omega}_3 = (I_1 - I_2)\omega_1\omega_2\\
&\dot{\phi} = \omega_1 + (\omega_2 \sin\phi + \omega_3 \cos\phi)\tan\theta\\
&\dot{\theta} = \omega_2 \cos\phi - \omega_3 \sin\phi\\
&\dot{\psi} = (\omega_2 \sin\phi + \omega_3 \cos\phi)\sec\theta
\end{aligned} \tag{20}$$

where, $\boldsymbol{I} = \mathrm{diag}(I_1, I_2, I_3)$ is the inertia moment, $\boldsymbol{x} = [\omega_1, \omega_2, \omega_3, \phi, \theta, \psi]^T \in \mathbf{R}^6$ is the state vector and $\boldsymbol{u} = [u_1, u_2]^T \in \mathbf{R}^2$ is the control input vector. Obviously, in Eq. (20) the 3-axis of the spacecraft has no control, thus it is called an underactuated axis, and the spacecraft is an underactuated spacecraft.

A set of flat outputs for system (20) can be easily found as $\boldsymbol{y} = [y_1, y_2, y_3]^T = [\phi, \theta, \psi]^T$, and the corresponding flat transformation can be described as

$$\begin{cases}
\tilde{\boldsymbol{x}} = \boldsymbol{A}(\tilde{\boldsymbol{y}}) = \begin{bmatrix} \dot{y}_1 - \dot{y}_3 \sin y_2 \\ \dot{y}_2 \cos y_1 + \dot{y}_3 \sin y_1 \cos y_2 \\ -\dot{y}_2 \sin y_1 + \dot{y}_3 \cos y_1 \cos y_2 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} \\
\begin{bmatrix} \tilde{\boldsymbol{u}} \\ w \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{B}}(\tilde{\boldsymbol{y}}) \\ \tilde{C}(\tilde{\boldsymbol{y}}) \end{bmatrix} = \begin{bmatrix} \ddot{y}_1 - \ddot{y}_3 \sin y_2 - \dot{y}_2\dot{y}_3 \cos y_2 \\ \ddot{y}_2 \cos y_1 - \dot{y}_1\dot{y}_2 \sin y_1 + \ddot{y}_3 \sin y_1 \cos y_2 + \dot{y}_1\dot{y}_3 \cos y_1 \cos y_2 - \dot{y}_2\dot{y}_3 \sin y_1 \sin y_2 \\ -\ddot{y}_2 s_{y_1} - (\alpha+1)(\dot{y}_1\dot{y}_2 c_{y_1} + \dot{y}_1\dot{y}_3 s_{y_1} c_{y_2}) + \ddot{y}_3 c_{y_1} c_{y_2} + (\alpha - 1)\dot{y}_2\dot{y}_3 c_{y_1} s_{y_2} + \alpha\dot{y}_3^2 s_{y_1} s_{y_2} c_{y_2} \end{bmatrix}
\end{cases} \tag{21}$$

where, $\tilde{\boldsymbol{y}} = [\boldsymbol{y}, \dot{\boldsymbol{y}}, \ddot{\boldsymbol{y}}]^T$ and the short-hand notation $s_x = \sin x$ and $c_x = \cos x$. The variable w is a fictitious input, and the details of the flat transformation can be found in [22,23].

By choosing the values of model parameters for the underactuated spacecraft, the moment of inertia matrix is $\boldsymbol{I} = \mathrm{diag}[55.3,\ 51.5,\ 41.8]\ \mathrm{kg\,m^2}$; The initial and desired states are $\boldsymbol{x}_0 = [0, 0, 0, 0, -\pi/4, 0]^T$ and $\boldsymbol{x}_f = [0, 0, 0,\ 0, 0, \pi/6]^T$, respectively; And the control saturation limit is assumed to be $u_M = 1$. Moreover, the simulations parameters for PSO algorithm are listed as below: the size of population

$s=30$; the maximum number of iterations $K_{\max}=1000$; the maximum and minimum values for the particle velocities $V_{\max}=5, V_{\min}=-5; c_1=c_2=2$; the penalty coefficient $M=10^4$, and the inertia weight w scaling linearly between 0.9 and 0.4. All the methods are coded in Matlab on PC with 2.1 GHz CPU/2GB RAM, and the NLP solver is KNITRO.

In order to find an initial guess solution more quickly, assuming the number of LGL points for the PSO

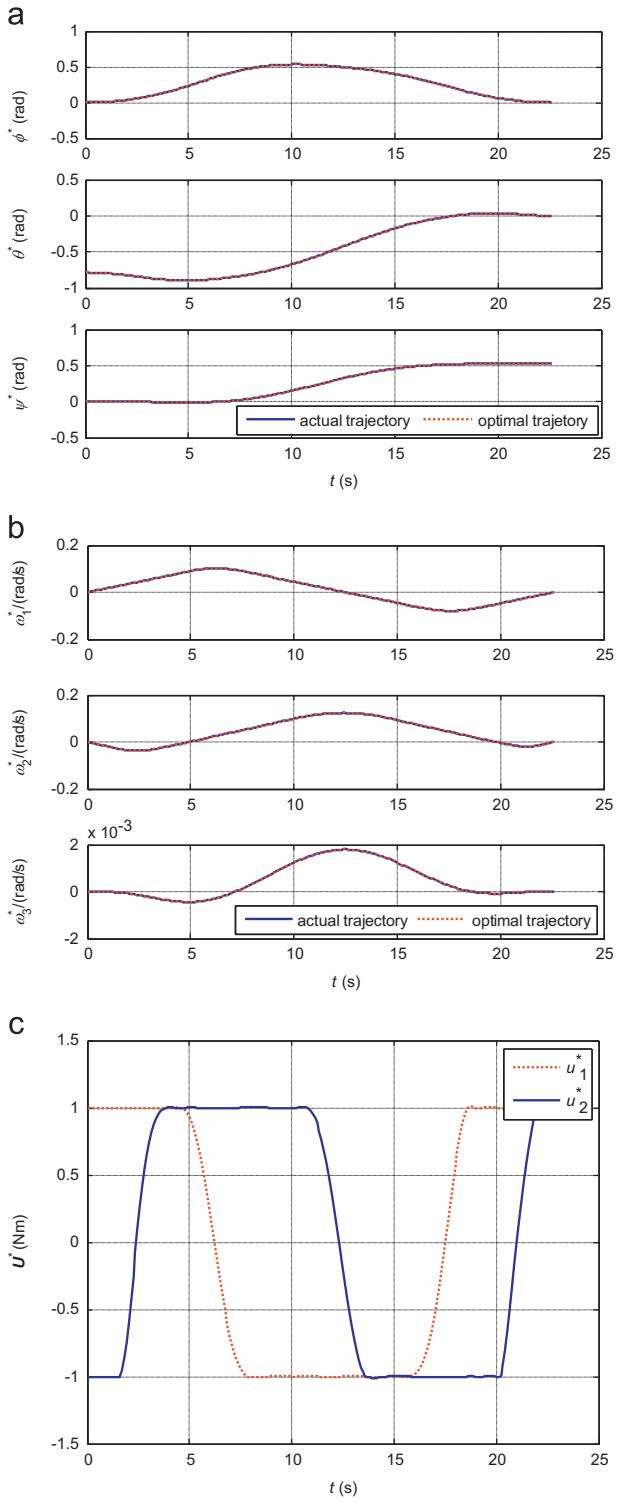

Fig. 2. Trajectories for optimal states and optimal control inputs. (a) Trajectories for Euler angles. (b) Trajectories for angular rates. (c) Optimal control inputs.

initialization phase is 11. Then, Fig. 1 shows the initialization solutions for state variables and the control input variables. And the execution time for the initialization process is 187.953 s. It is seen that the state trajectories in Fig. 1(a) and (b) can meet the end constraints very well, which demonstrates that the solutions obtained in the initialization phase are feasible solutions.

Moreover, with the solutions in Fig. 1 as the initialization guess solutions, the original time-optimal trajectory planning problem is solved by the optimization software KNITRO with the number of LGL points as 21. Fig. 2 shows the time-optimal trajectories of the underactuated spacecraft from initial point $\boldsymbol{x}_0$ to final point $\boldsymbol{x}_f$. The required execution time of the optimization process is $t_c=24.6961$s, and the resulting value of cost function is $J=22.6065$ s. In Fig. 1(a) and (b), there are actually plotted two separate sets of trajectories, the dashed line is planned by directly solving the time-optimal trajectory planning problem using the PSO–LPM algorithm; the other solid line is generated by integrating the dynamic models in Eq. (20) subject to the control inputs histories in Fig. 2(c). The two sets are almost coincident, so there is no visible discrepancy in Fig. 2(a) and (b). And Fig. 2(c) also shows that the corresponding control input histories which satisfy the control input saturation restriction very well. Fig. 3 shows the state error trajectories on 21 LGL points, and obviously the approximation accuracy for the algorithm proposed is less than 1×10^{-4}. Furthermore, it is apparent from Fig. 2(c) that the control input is in the form of Bang–Bang control, thus one can conclude that the solution is a global optimum.

In order to further analyze the performance of the PSO–LPM algorithm, Monte Carlo simulations are carried out with 200 groups of random initial values, and meanwhile other simulation parameters are kept similar with the ones above. The simulation results are plotted in Figs. 4–5, Fig. 4 shows the performance indexes for 200 simulations, and Fig. 5 shows the maxima of state variable errors also for 200 simulations.

Remark. : In the robustness analysis, the values of the random initial guesses $\lambda^*=[\lambda_0^*,\lambda_1^*,\ldots,\lambda_N^*]^T$ for the PSO phase should be bounded, such that in Step 11 of the PSO–LPM algorithm, the attitude angles after flat transformation can still satisfy the constraints as $\phi,\psi\in[-\pi,\pi],\theta\in(-\pi/2,\pi/2)$.

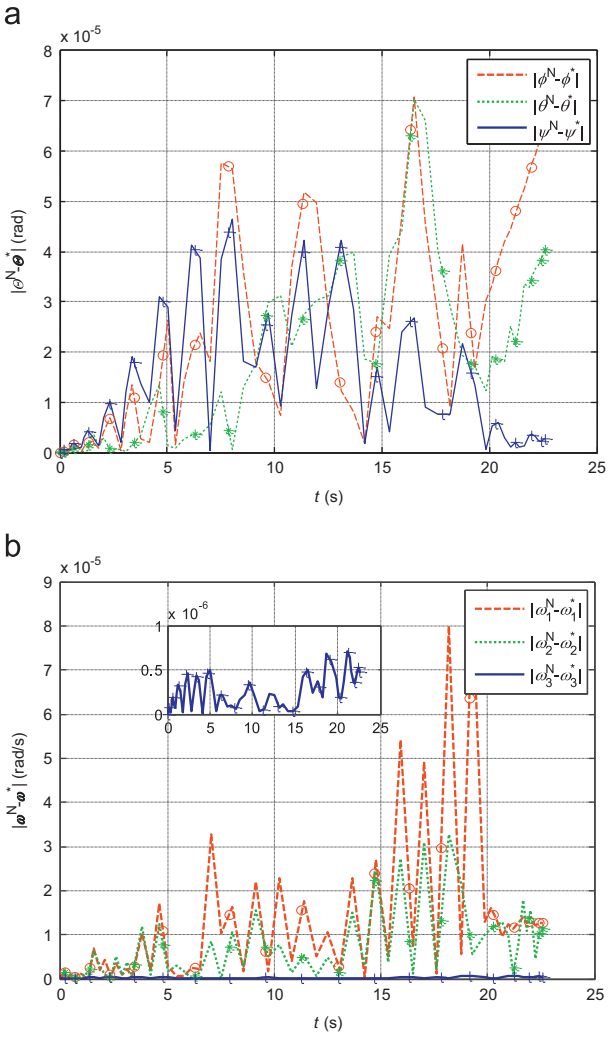

Fig. 3. State errors at LGL nodes. (a) Euler angle errors at LGL nodes. (b) Angular rate errors at LGL nodes.

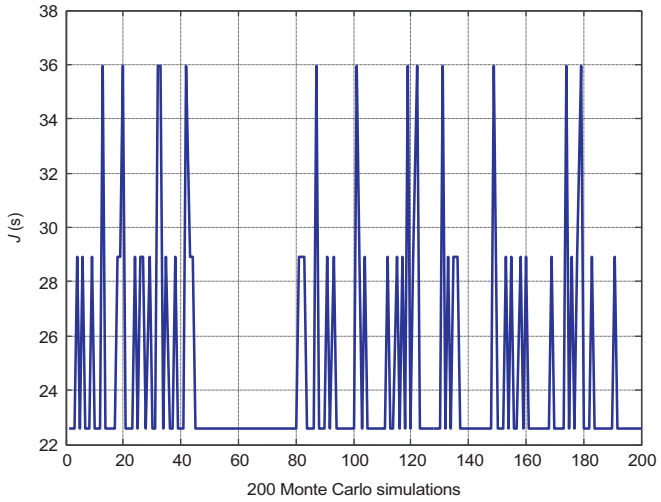

Fig. 4. Performance indexes for 200 Monte Carlo simulations.

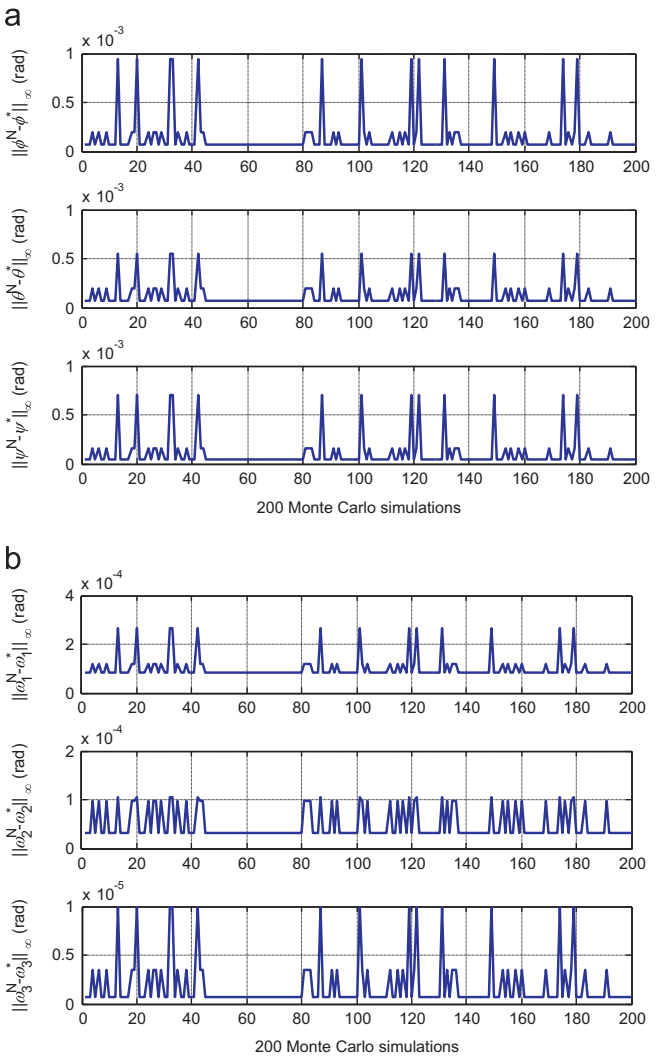

Fig. 5. Maxima of state errors for 200 Monte Carlo simulations. (a) Maxima of Euler angle errors for 200 Monte Carlo simulations. (b) Maxima of angular rate errors for 200 Monte Carlo simulations.

Table 1
Comparison of performances between LPM algorithm with PSO–LPM algorithm.

Method	Global optimum 22.6065 s Times (Percentage)	Local optimum 1 28.9201 s Times (Percentage)	Local optimum 2 35.9457 s Times (Percentage)	Average execution time t_c (s)
A	40 (20)	38 (19)	53 (26.5)	33.4674
B	151 (75.5)	36 (18)	13 (6.5)	21.9083

In Fig. 4, it can be seen that there exist three optimal solutions for the time-optimal trajectory panning problem for underactuated spacecraft, one of which is the global optimum $J=22.6065$ s, and the other two are local sub-optima 28.9201 s and 35.9457 s. Furthermore in Figs. 4–5, it is found that a more optimal solution results in higher approximation accuracy.

Table 1 presents the comparison results of performances between LPM algorithm and PSO–LPM algorithm, where Method A is the LPM algorithm without the initialization phase, and Method B is the PSO–LPM algorithm proposed in this paper. In addition, there are 42 times(21%) in Method A, where the solutions obtained are not optimal but only feasible ones. And there are also 27 times(13.5%) failing to find a solution, which are not recorded in Table 1. The simulation results demonstrate that PSO–LPM algorithm has a greater advantage than single LPM algorithm in term of global searching ability.

6. Conclusions

A hybrid particle swarm optimization (PSO) method for generating initial solutions to gradient-based direct trajectory optimization is proposed in this paper. High-precision is obtained by using Legendre pseudospectral method, which transcribes the optimal problem into a nonlinear programming (NLP). And a feasible and near-optimal initial guess for the NLP phase is provided by PSO method, which constructs a random search in initial condition space. Furthermore, the proposed approach is applied to an underactuated rigid spacecraft trajectory optimization problem considering its differential flatness property. Simulation results demonstrate that the proposed approach is competitive in convergence rate, robustness and global searching capability than single PSO and other classical optimization algorithms with gradient information.

Also, the approach proposed in this paper could be extended to other underactuated systems, and there still remains great potential for future research in the area of trajectory optimization with more complex constraints.

Acknowledgments

This work was supported in part by the Fundamental Research Funds for the Central Universities (HIT. NSRIF. 2013135) and the Natural Scientific Research Innovation Foundation in Harbin Institute of Technology (HIT. NSRIF. 2014139).

References

[1] S.L. Scrivener, R.C. Thompson, Survey of time-optimal attitude maneuvers, J. Guidance, Control, Dyn., 17, 225–233.
[2] L.C Lai, C.C. Yang, C.J. Wu., Time-optimal maneuvering control of a rigid spacecraft, Acta Astronaut. 60 (2007) 791–800.
[3] C.C. Yang, C.J. Wu., Time-optimal de-tumbling control of a rigid spacecraft, J. Vib. Control 14 (2008) 553–570.
[4] B.L. Wu, D.W. Wang, Nonlinear optimization of low-thrust trajectory for satellite formation: Legendre pseudospectral approach, J. Guidance, Control, Dyn. 32 (4) (2009) 1371–1381.
[5] J.T. Betts, Survey of numerical methods for trajectory optimization, J. Guidance, Control, Dyn. 21 (2) (1998) 193–207.
[6] X.S. Ge, L.Q. Chen, Attitude control of a rigid spacecraft with two momentum wheel actuators using genetic algorithm, Acta Astronaut. 55 (1) (2004) 3–8.
[7] L. Tian, C. Collins, An effective robot trajectory planning method using a genetic algorithm, Mechatronics 14 (2004) 455–470.
[8] J.L. Foo, J. Knutzon, V. Kalivarapu, et al., Path planning of unmanned aerial vehicles using B-splines and particle swarm optimization J. Aerosp. Comput. Inf., Commun. 6 (2009) 271–290.
[9] R. Biesbroek, A Comparison of Differential Evolution Method with Genetic Algorithms for Orbit Optimization, International Astronautical Federation Paper IAF-06-C1.4.02, 2006.
[10] N. Yokoyama, S. Suzuki, Modified genetic algorithm for constrained trajectory optimization, J. Guidance, Control, Dyn. 28 (1) (2005) 139–144.
[11] L.C. Cagnina, S.C. Esquivel, C.A. Coello, Solving constrained optimization problems with a hybrid particle swarm optimization algorithm, Eng. Optim. 43 (8) (2011) 843–866.
[12] M.A. Vavrina, K.C. Howell. Global low-thrust trajectory optimization through hybridization of a genetic algorithm and a direct method, in: Proceedings of the AIAA/AAS Astrodynamics Specialist Conference and Exhibit, Honolulu, Hawaii, 2008.
[13] K. Subbarao, B.M. Shippey, Hybrid genetic algorithm collocation method for trajectory optimization, J. Guidance, Control, Dyn. 32 (4) (2009) 1396–1403.
[14] B. Shippey, K. Subbarao. Trajectory design using collocation and genetic algorithms: aircraft turning maneuver, in: Proceedings of the 17th IEEE International Conference on Control Applications Part of 2008 IEEE Multi-conference on Systems and Control, 2008, pp. 905–911.
[15] H. Modares, M.N. Sistani, Solving nonlinear optimal control problems using a hybrid IPSO–SQP algorithm, Eng. Appl. Artif. Intell. 24 (2011) 476–484.
[16] M. Fliess, J. Levine, P. Martin, P. Rouchon, Flatness and defect of nonlinear systems: introductory theory and example, Int. J. Control 61 (6) (1995) 1327–1361.
[17] J. Kennedy, R. Eberhart. Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural Networks, 1995, pp. 1942–1945.
[18] Q. Gong, W. Kang, I.M. Ross., A pseudospectral method for the optimal control of constrained feedback linearizable systems, IEEE Trans. Autom. Control 51 (7) (2006) 1115–1129.
[19] P. Tsiotras, J. Luo, Control of underactuated spacecraft with bounded inputs, Automatica 36 (8) (2000) 1153–1169.
[20] P. Tsiotras, V. Doumtchenko, Control of spacecraft subject to actuator failures: state-of-the-art and open problems, J. Astronaut. Sci. 48 (2) (2000) 337–358.
[21] C.O. Aguilar, Attitude control of a differentially flat underactuated rigid spacecraft (Ph.D. thesis), University of Alberta, Ann Arbor, MI, 2005.
[22] Y.F. Zhuang, G.F. Ma, C.J. Li, et al., Time-optimal trajectory planning for underactuated rigid spacecraft using differential flatness, J. Astronaut. 32 (8) (2011) 1753–1761.
[23] Y.F. Zhuang, H.B. Huang, Real-time trajectory optimization of an underactuated rigid spacecraft using differential flatness, Aerosp. Sci. Technol. 23 (2012) 32–139.