Aerospace Science and Technology 76 (2018) 280–290

Contents lists available at ScienceDirect

Aerospace Science and Technology

www.elsevier.com/locate/aescte

Sequential convex programming for nonlinear optimal control problems in UAV path planning

Zhe Zhang [a], Jianxun Li [a,*], Jun Wang [b]

[a] Department of Automation, School of Electronic Information and Electric Engineering, Shanghai Jiao Tong University, 800 Dong Chuan Road, Shanghai, China
[b] Luoyang Electronic Equipment Test Center of China, Luoyang, China

ARTICLE INFO

Article history:
Received 16 March 2017
Received in revised form 9 November 2017
Accepted 10 January 2018
Available online 15 February 2018

Keywords:
UAV path planning
Sequential convex programming
Nonlinear optimal control
Globally convergent algorithm
Non-convex programming

ABSTRACT

Usually, an UAV (Unmanned Aerial Vehicle) path planning problem can be modeled as a nonlinear optimal control problem with non-convex constraints in practical applications. However, it is quite difficult to obtain stable solutions quickly for this kind of non-convex optimization with certain convergence and optimality. In this paper, an algorithm is proposed to solve the problem through approximating the non-convex parts by a series of sequential convex programming problems. Under mild conditions, the sequence generated by the proposed algorithm is globally convergent to a KKT (Karush–Kuhn–Tucker) point of the original nonlinear problem, which is verified by a rigorous theoretical proof. Compared with other methods, the convergence and effectiveness of the proposed algorithm is demonstrated by trajectory planning applications.

© 2018 Elsevier Masson SAS. All rights reserved.

1. Introduction

Nowadays, UAVs show high potential and rapid proliferation in civilian domains such as package delivery, precision agriculture, emergency management and imaging surveillance. Simultaneously, UAVs are widely employed in military fields such as detection, offensive and assessment [1]. Some applications mentioned above are tedious or unsafe for humans so the usage of UAVs increases dramatically, which greatly inspires researchers to develop their investigations. However, we encounter great technical challenges for UAVs in these applications. Path planning can help UAVs autonomously design the route that minimizes the cost and avoids collisions with obstacles or other vehicles; therefore, path planning plays a critical role in UAV applications.

Basically, the goal of UAV path planning is to minimize fuel usage, flying time or other factors with minimum task risk [2]. The UAV self-limiting conditions, atmospheric turbulence, partial information of environment and limited sensor capabilities bring UAV path planning and optimization more challenges and time-consumption [3]. Concerning the mentioned issues, researchers carry out their own approaches on UAV path planning [4,5]. The main methods of UAV path planning can be divided into two categories: artificial intelligence algorithms and numerical methods. Artificial intelligence algorithms such as Genetic Algorithm (GA) [6] and Particle Swarm Optimization (PSO) [7] can solve the problem well. However, the convergence of these algorithms cannot be guaranteed before the simulation. This uncertain factor prevents them from being utilized for widespread applications.

In most numerical methods, UAV path planning problems are modeled as nonlinear optimal control problems. Prior methods for optimal control problems with state constraints include [8,9] and the well-known constrained optimal control software [10] is developed. Methods mentioned above take the optimal control problem as a general non-convex programming problem. In common sense, it is quite difficult to solve a non-convex programming problem because of its nature: there is no bound of operation time and initial guess which should be supplied by a human is required. As the development of Sequential Quadratic Programming (SQP) algorithm, some nonlinear programming problems can be solved fast and stably with global convergence [11]; however, the solution is very likely to be feasible or local optimal rather than global optimal, that is, the optimality of the solution can not be guaranteed [12]. Therefore, directly solving the nonlinear optimal control problem by general non-convex programming is not appropriate for onboard applications and how to solve the nonlinear optimal control problem stably and quickly with certain optimality becomes a primary content of this paper. This problem will be solved perfectly if the non-convex problem can be transformed into a convex problem or a series of convex problems. For convex programming, the local optimal is global optimal, the solution time is bounded and initial guess is not necessary [12]. Unfortunately, the transformation expense, unguaranteed optimality and uncertain equivalence

* Corresponding author.
E-mail address: lijx@sjtu.edu.cn (J. Li).

https://doi.org/10.1016/j.ast.2018.01.040
1270-9638/© 2018 Elsevier Masson SAS. All rights reserved.

make this method not that practical. It is a feasible alternative that the solution to the approximated problem finds the key point (such as KKT point or local optimal point and so on) of the original problem.

Based on the idea above, in [13] authors present a methodology to nonlinear optimal control problems which only involve concave inequality constraints and its key process is to approximate the concave inequality constraints by successive linearization. However, the non-convex parts in our problem may arise in cost function or inequality constraints which forms a non-convex feasible region rather than convex or concave. To handle the non-convex part, the idea of Concave Convex Procedure (CCCP) is introduced, which indicates that under mild condition a function can be decomposed into the sum of a convex function and a concave function; and then a sequential convex programming algorithm with convergence analysis is proposed in this paper to solve the UAV path planning problem in nonlinear optimal control model. The main contributions of this paper are listed as follows.

- The UAV path planning problem is modeled as a nonlinear optimal control problem and approximated by a series of sequential convex programming problems through CCCP.
- A sequential convex programming algorithm is proposed to obtain the solution to the nonlinear optimal control problem more quickly and more stably.
- Global convergence of proposed algorithm is rigorously analyzed to guarantee the output of the algorithm being the KKT point of the original nonlinear problem.

The paper is organized as follows: Section 2 presents the nonlinear optimal control model of the UAV path planning problem and its nonlinear programming transformation. How to approximate to original problem and the sequential convex programming algorithm can be found in Section 3. The main technical result for optimality and equivalence between the two problems are analyzed in Section 4. Section 5 provides the simulation results to verify the algorithm, and Section 6 concludes this paper.

2. Problem statement

The UAV path planning problem is an optimization problem to obtain an optimal cost value under limited conditions in mathematics essentially, and it is usually modeled as a nonlinear optimal control problem in numerical methods. The model of the UAV path planning is denoted as Problem Optimal Control (**POC**) in the remainder of the paper.

POC

$$\min_{x,u} \quad J = \int_{t_0}^{t_f} L(x(t), u(t))dt \tag{1}$$

$$s.t. \quad \dot{x}(t) = f\left(x(t), u(t)\right), \tag{2}$$

$$\|M_i(t)y(t) + p_i(t)\| \le q_i^T(t)y(t) + r_i(t), \tag{3}$$

$$s_i(x,u) \le 0,\ i = 1, \ldots, n_s, \tag{4}$$

$$C(t)y(t) + d(t) = 0, \tag{5}$$

where Eq. (1) is the cost function, t_0 and t_f are the initial time and final time respectively. $L(x,u)$ is a cost function of state variable x and control variable u which depends on time implicitly. Eq. (2) is the state equation of state variable x and control variable u; $f(x,u)$ is the dynamic function of system and can be linear or nonlinear with bounded Hessain. In Eq. (3), $y = (x^T, u^T)^T$ is introduced to describe the problem better. $M_i(t)$, $p_i(t)$, $q_i(t)$, $r_i(t)$ are functional matrices of time t of proper dimensions and Eq. (3) are the second-cone constraints (therefore convex). In Eq. (4), $s_i(x,u)$ is a function of x and u, and may be convex, concave, or no-convex-no-concave but has bounded Hessian, which is the main non-convex part of the model. n_s is number of the non-convex constraints. The final Eq. (5) is the equality constraint of $x(t)$ and $u(t)$ as well as including the initial and final conditions of variables, and $C(t)$, $d(t)$ are functional matrices in proper dimensions.

POC can be expressed in a more concise form through mathematical transformation. The details can be found in [13]. The problem in new form is denoted as Problem 0 (**P0**).

P0

$$\min_{y} f(y) \tag{6}$$

$$s.t.\ g_i(y) \le 0,\ i = 1, \ldots, p, \tag{7}$$

$$h_j(y) = 0,\ j = 1, \ldots, q, \tag{8}$$

where $f(y) : \mathbb{R}^n \to \mathbb{R}$, $g_i(y) : \mathbb{R}^n \to \mathbb{R}$ and $h_j(y) : \mathbb{R}^n \to \mathbb{R}$ are linear or nonlinear and smooth functions on their domain with bounded Hessians, therefore, $f(y)$, $\{g_i(y)\}_{i=1}^p$ and $\{h_j(y)\}_{j=1}^q$ are twice continuously differentiable.

Remark 1. It is assumed that the constraints in **P0** satisfies the standard qualification requirement of optimization problems, that is, the optimal solution to **P0** is a regular point. A regular point is a point at which the gradient vectors of active constraints are linearly independent [14].

From Theorem 2 in Appendix B, a twice differentiable function with bounded Hessian can be decomposed into the sum of a convex function and a concave function; therefore, $f(y)$, $\{g_i(y)\}_{i=1}^p$ and $\{h_j(y)\}_{j=1}^q$ can be reformulated as:

$$\begin{aligned} f(y) &= f_{vex}(y) + f_{cave}(y), \\ g_i(y) &= g_{ivex}(y) + g_{icave}(y),\ i = 1, \ldots, p, \\ h_j(y) &= h_{jvex}(y) + h_{jcave}(y),\ j = 1, \ldots, q, \end{aligned} \tag{9}$$

where functions with subscript vex are convex functions with eigenvalues of their Hessians more than zero, or more precisely: strict convex functions, and strict concave functions with the subscript $cave$. Substituting Eq. (9) into **P0**, the equality constraint $h_{jvex}(y) + h_{jcave}(y) = 0$ can be expressed as two inequality constraints $h_{jvex}(y) + h_{jcave}(y) \le 0$ and $h_{jvex}(y) + h_{jcave}(y) \ge 0$. The second constraint is equivalent to $-h_{jvex}(y) - h_{jcave}(y) \le 0$, i.e. $(-h_{jcave}(y)) + (-h_{jvex}(y)) \le 0$. Then, the problem can be reformulated as Problem 1 (**P1**), which only has inequality constraints.

P1

$$\min_{y} \ f_{vex}(y) + f_{cave}(y) \tag{10}$$

$$s.t.\ g_{ivex}(y) + g_{icave}(y) \le 0,\ i = 1, \ldots, p, \tag{11}$$

$$h_{jvex}(y) + h_{jcave}(y) \le 0,\ j = 1, \ldots, q, \tag{12}$$

$$(-h_{jcave}(y)) + (-h_{jvex}(y)) \le 0,\ j = 1, \ldots, q. \tag{13}$$

Remark 2. The implicit relationship between new inequality constraints (12) and (13) is no longer considered, and they are treated as two independent inequality constraints in the remainder of the paper. Actually, in practical applications with linear dynamic system functions, overwhelming majority of equality constraints in UAV path planning problems are linear, which can be neglected in the proofs of subsequent sections because of their convexity [15].

For clarity, a new notation is applied.

$$\begin{cases} u_i(y) = f_{vex}(y),\ v_i(y) = -f_{cave}(y), & i = 0, \\ u_i(y) = g_{ivex}(y),\ v_i(y) = -g_{icave}(y), & i = 1, \ldots, p, \\ u_i(y) = h_{jvex}(y),\ v_i(y) = -h_{jcave}(y), & \\ \quad i = p+1, \ldots, p+q, & \\ u_i(y) = -h_{jcave}(y),\ v_i(y) = h_{jvex}(y), & \\ \quad i = p+q+1, \ldots, p+2q. & \end{cases} \tag{14}$$

Let $m \triangleq p + 2q$, then $\{u_i(y)\}_{i=0}^{m}$ and $\{v_i(y)\}_{i=0}^{m}$ are strict convex functions with positive definite Hessians. Substituting notation (14) to **P1**, it is reformulated as a standard non-convex programming problem and denoted as **NP**.

NP

$$\min_{y} \ u_0(y) - v_0(y) \tag{15}$$

$$s.t.\ u_i(y) - v_i(y) \le 0,\ i = 1, \ldots, m. \tag{16}$$

The feasible region of **OP** is denoted as

$$\eta = \{y | u_i(y) - v_i(y) \le 0,\ i = 1, \cdots, m\}. \tag{17}$$

Actually, as a non-convex optimization problem, **NP** is difficult to solve and sensitive to the initial guess. As mentioned in Section 1, to obtain a solution to **NP** quickly and stably with certain optimality and convergence, the idea of transformation is introduced, and **NP** is approximated by a series of sequential convex programming problems.

3. Sequential convex programming

Based on the idea of transformation, **NP** is approximated by a series of sequential convex programming problems. To better describe the approximated problems, some useful notations are defined:

$$\tilde{f}(x; \alpha) \triangleq f(\alpha) + \nabla f^T(\alpha)(x - \alpha), \tag{18}$$

$$\ddot{f}(x; \alpha) \triangleq \frac{1}{2}(x - \alpha)^T \nabla^2 f[\alpha + \theta(x - \alpha)](x - \alpha), \tag{19}$$

where f is a twice differentiable function on its domain and $\theta \in [0, 1]$. From the definition of $\tilde{f}(x; \alpha)$ (18), it can be easily derived that

$$\begin{aligned} \nabla \tilde{f}(x; \alpha) &= \nabla \left[f(\alpha) + \nabla f^T(\alpha)(x - \alpha) \right] \\ &= \nabla \left[\nabla f^T(\alpha) x \right] = \nabla f(\alpha). \end{aligned} \tag{20}$$

Applying the notation (18), the non-convex parts of **NP** are linearized at k-iterated solution y_k to form an approximated convex programming problem and denoted as $\mathbf{CP}(y_k)$.

CP(y_k)

$$\min_{y} \ u_0(y) - \tilde{v}_0(y; y_k) \tag{21}$$

$$s.t.\ u_i(y) - \tilde{v}_i(y; y_k) \le 0,\ i = 1, \ldots, m. \tag{22}$$

The feasible region of **CP**(y_k) is denoted as

$$\xi_k = \left\{ y | u_i(y) - \tilde{v}_i(y; y_k) \le 0,\ i = 1, \cdots, m \right\}. \tag{23}$$

In the common sense, **CP**(y_k) is a convex programming problem, because $u_i(y)$, $i = 0, \ldots, m$ are convex functions and the remaining parts $\tilde{v}_0(y; y_k)$, $i = 0, \ldots, m$ are linear functions.

Based on the new convex programming problem **CP**(y_k), the main algorithm of this paper is proposed to obtain a fast and stable solution to **NP** with certain convergence and optimality. The main algorithm is a series of sequential convex programming problem to approximate **NP** and called Sequential Convex Programming Algorithm, which is denoted as $\mathcal{A}_{scp}$ and listed in Algorithm: Sequential Convex Programming Algorithm.

Algorithm: Sequential Convex Programming Algorithm: $\mathcal{A}_{scp}$.

```
Initialization: Input: Any y_1 ∈ η, tolerance ε
for k = 1 until meeting terminal condition do
    Obtain y_{k+1}: {y_{k+1}} = CP(y_k) (Eqs. (21)–(22))
    if ‖y_{k+1} − y_k‖ ≤ ε then
        break;
    end
    update: Set k = k + 1
end
Output: y_k or y_{k+1}
```

Using $\mathcal{A}_{scp}$, a sequence $y_1, y_2, \ldots, y_k, \ldots$ is generated. However, the convergence of the sequence has not been proved, and the optimality of the final output y_k or y_{k+1}, i.e., the relationship between solution to **NP** and the output of $\mathcal{A}_{scp}$, is vague. All above will be clarified in subsequent sections.

In this section, the non-convex programming problem **NP** has been approximated by a series of convex programming problems **CP**(y_k) $(k = 1, \cdots,)$ with some useful notations, and then the main algorithm $\mathcal{A}_{scp}$ has been proposed. In next section, we will prove that $\mathcal{A}_{scp}$ is a globally convergent algorithm and the sequence generated by $\mathcal{A}_{scp}$ is convergent to the KKT point of **NP**. In addition, if the cost function of original problem (**NP**) has a special structure, the KKT point will be a local optimal point at least.

4. The main theorem and proof

In this section, the main technique result of the paper will be presented. As necessary conditions, the following assumptions are satisfied in the remainder of the paper.

Assumptions.

1. There exists an optimal solution to the problem **CP**(y_1) and the solutions to **NP** and **CP**(y_k) are bounded.
2. The discretization procedure do not change the existence and boundedness of solutions to **NP** and **CP**(y_k).

The main result is derived from Meyer's theorem [16]. Together with some useful definitions, Meyer's theorem can be found in Appendix. For meeting the conditions of Meyer's theorem, some lemmas are derived.

At first, the relationship between solutions to **CP**(y_k) and **NP** is proposed.

Lemma 1. *The solution to **CP**(y_k) is feasible to **NP**.*

Proof. Lemma 1 implies if y is a feasible solution to **CP**(y_k), then y will also be feasible to **NP**. Let α be an arbitrary feasible solution to **CP**(y_k), therefore satisfies the constraints in Eq. (22), that is,

$$u_i(\alpha) - \tilde{v}_i(\alpha; y_k) \le 0. \tag{24}$$

The second-order Taylor expansion of $v_i(\alpha)$ at y_k is

$$v_i(\alpha) = \tilde{v}_i(\alpha; y_k) + \ddot{v}_i(\alpha; y_k). \tag{25}$$

Using $u_i(\alpha)$ minus Eq. (25), we obtain

$$u_i(\alpha) - v_i(\alpha) = u_i(\alpha) - \tilde{v}_i(\alpha; y_k) - \ddot{v}_i(\alpha; y_k). \tag{26}$$

From Eq. (24), Eq. (26) leads to

$$u_i(\alpha) - v_i(\alpha) + \ddot{v}_i(\alpha; y_k) \le 0. \tag{27}$$

Since $v_i(y)$ is a convex function with positive definite Hessian, $\nabla^2 v_i[y_k - \theta(\alpha - y_k)] \succ 0$, that is, $\ddot{v}_i(\alpha; y_k) \ge 0$, therefore,

$$u_i(\alpha) - v_i(\alpha) \le u_i(\alpha) - v_i(\alpha) + \ddot{v}_i(\alpha; y_k) \le 0 \tag{28}$$

can be derived, which means α is feasible to **NP**. □

Remark 3. From the procedure of $\mathcal{A}_{scp}$, it can be found that the output of $\mathcal{A}_{scp}$ after k iterations must be the solution to **CP**(y_k), therefore, is feasible to **NP** according to Lemma 1.

Lemma 2. *Assuming suitable constraint qualification and the feasible region ξ_k is uniformly compact and closed, there will always be a sequence generated by $\mathcal{A}_{scp}$.*

Proof. Lemma 2 guarantees the continuity of $\mathcal{A}_{scp}$, that is, input any $y_1 \in \eta$, **CP**(y_1) exists an optimal solution according to assumptions and the optimal solution is denoted as y_2 to form the new convex problem **CP**(y_2); and then **CP**(y_2) is feasible and exists an optimal solution, y_3 can be calculated; ... until the terminal condition of $\mathcal{A}_{scp}$ is satisfied.

Assuming y_k is the optimal solution to **CP**(y_{k-1}), if there exists an optimal solution to the newly generated convex programming problem **CP**(y_k) based on y_k, the continuity of $\mathcal{A}_{scp}$ is guaranteed; therefore, existence of the optimal solution to **CP**(y_k) needs to be verified.

Substituting $y = y_k$ into the constraints (22) of **CP**(y_k), it can be derived that

$$\begin{aligned} &u_i(y_k) - \tilde{v}_i(y_k; y_k) \\ &= u_i(y_k) - \left[v_i(y_k) + (y_k - y_k)^T \nabla v_i(y_k)\right] \\ &= u_i(y_k) - v_i(y_k). \end{aligned} \tag{29}$$

y_k is the optimal solution to **CP**(y_{k-1}), therefore y_k is feasible to **NP** from Lemma 1, that is, $u_i(y_k) - v_i(y_k) \le 0$, which indicates y_k is a feasible solution to **CP**(y_k) and the feasible region ξ_k is nonempty. Assuming suitable constraint qualification, ξ_k is uniformly compact and closed and the cost function of **CP**(y_k) is continuous on ξ_k. From convex optimization theory [12], there exists an optimal solution to **CP**(y_k) in ξ_k, which proves the continuity of $\mathcal{A}_{scp}$ and completes the proof of Lemma 2. □

Lemma 3. *$\mathcal{A}_{scp}$ is a Point-to-Set Map, and an Iterative Algorithm, what is more, a Descent Algorithm (Monotone Algorithm).*

Proof. Using notation,

$$\xi = \bigcup_{k=1}^{\infty} \xi_k, \tag{30}$$

where ξ_k is the feasible region of **CP**(y_k). ξ denotes X and the sequence generated by $\mathcal{A}_{scp}$ denotes Y, therefore, $\mathcal{A}_{scp}$ is a Point-to-Set Map according to Definition 1 in Appendix A. $\mathcal{A}_{scp}$ indicates y_k is generated by **CP**(y_{k-1}) and from the Definition 3 in Appendix A, thus, $\mathcal{A}_{scp}$ is an Iterative Algorithm.

$\mathcal{A}_{scp}$ will be a Descent Algorithm, if the value of cost function decreases after each iteration. Supposing that y_{k+1} is the optimal solution to **CP**(y_k) and y_k is the optimal solution to **CP**(y_{k-1}), if the following inequality (31) is satisfied, $\mathcal{A}_{scp}$ will be a Descent algorithm.

$$\begin{aligned} u_0(y_{k+1}) - v_0(y_{k+1}) &\le u_0(y_{k+1}) - \tilde{v}_0(y_{k+1}; y_k) \\ &\le u_0(y_k) - v_0(y_k) \le u_0(y_k) - \tilde{v}_0(y_k; y_{k-1}) \end{aligned} \tag{31}$$

The equality meets when $y_{k+1} = y_k = y_{k-1}$.

It can be found that y_k is a feasible solution to **CP**(y_k) from Lemma 2, and y_{k+1} is the optimal solution to **CP**(y_k), therefore,

$$u_0(y_{k+1}) - \tilde{v}_0(y_{k+1}; y_k) \le u_0(y_k) - \tilde{v}_0(y_k; y_k). \tag{32}$$

From the definition $\tilde{v}_0(y_k; y_k) = v_0(y_k) + \nabla v_0^T(y_k)(y_k - y_k) = v_0(y_k)$, the second inequality of Eq. (31) has been derived. The second-order Taylor expansion of $v_0(y_{k+1})$ at y_k is

$$v_0(y_{k+1}) = \tilde{v}_0(y_{k+1}; y_k) + \ddot{v}_0(y_{k+1}; y_k). \tag{33}$$

Since $v_0(y)$ is a convex function with positive definite Hessian, $\nabla^2 v_0[y_k - \theta(y_{k+1} - y_k)] \succ 0$. Therefore, $\ddot{v}_0(y_{k+1}; y_k) > 0 \Rightarrow v_0(y_{k+1}) \ge \tilde{v}_0(y_{k+1}; y_k)$ which proves the first inequality of Eq. (31). The proof of last inequality of Eq. (31) is the same as the first inequality of Eq. (31).

CP(y_k) is a convex programming problem, the local optimal is global optimal, and u_i, v_i are strictly convex functions, so above inequalities become equalities when and only when $y_{k+1} = y_k = y_{k-1}$.

In conclusion, $\mathcal{A}_{scp}$ is a Descent Iterative Algorithm. □

Lemma 4. *The Fixed Point of $\mathcal{A}_{scp}$ is the KKT point of **NP**.*

Proof. Supposing y^* is the optimal solution to **CP**(y_k) and $y^* = y_k$, and then y^* is a Fixed point of $\mathcal{A}_{scp}$ from Definition 2. $\mathcal{A}_{scp}$ must stop at the point y^* because the terminal condition is satisfied from steps **4–5** of $\mathcal{A}_{scp}$. We need to prove $y^* = y_k$ is the KKT point of **NP**.

The Lagrangian of **CP**(y_k) is

$$L(s_i) = u_0(y) - \tilde{v}_0(y; y_k) + \sum_{i=1}^{m} s_i\left[u_i(y) - \tilde{v}_i(y; y_k)\right]. \tag{34}$$

The KKT condition of **CP**(y_k) is expressed as

$$\begin{cases} \nabla L(s_i) = \nabla u_0(y) - \nabla \tilde{v}_0(y; y_k) \\ \qquad\qquad + \sum_{i=1}^{m} s_i\left[\nabla u_i(y) - \nabla \tilde{v}_i(y; y_k)\right] = 0 \\ s_i\left[u_i(y) - \tilde{v}_i(y; y_k)\right] = 0 \\ s_i \ge 0 \end{cases}, \quad i = 1, \ldots, m. \tag{35}$$

y^* is the optimal solution to **CP**(y_k), therefore satisfies Eq. (35). Substituting $y = y^*$ and applying $y^* = y_k$ into Eq. (35) and combining with Eq. (20), it can be derived that

$$\begin{cases} \nabla L(s_i) \\ \quad = \nabla u_0(y^*) - \nabla \tilde{v}_0(y^*; y_k)) + \sum_{i=1}^{m} s_i\left[\nabla u_i(y^*) - \nabla \tilde{v}_i(y^*; y_k)\right] \\ \quad = \nabla u_0(y^*) - \nabla v_0(y_k) + \sum_{i=1}^{m} s_i\left[\nabla u_i(y^*) - \nabla v_i(y_k)\right] \\ \quad = \nabla u_0(y^*) - \nabla v_0(y^*) + \sum_{i=1}^{m} s_i\left[\nabla u_i(y^*) - \nabla v_i(y^*)\right] = 0 \\ s_i\left[u_i(y^*) - \tilde{v}_i(y^*; y_k)\right] \\ \quad = s_i\left[u_i(y^*) - v_i(y_k) + \nabla v_i^T(y_k)(y^* - y_k)\right] \\ \quad = s_i\left[u_i(y^*) - v_i(y_k)\right] \\ \quad = s_i\left[u_i(y^*) - v_i(y^*)\right] = 0 \\ s_i \ge 0, \qquad\qquad i = 1, \ldots, m, \end{cases} \tag{36}$$

which is the KKT condition of **NP** at the point y^*, in other words, y^* is the KKT point of **NP**. □

Above lemmas have discussed some properties of $\mathcal{A}_{scp}$ and the relationship between the solution to **NP** and the output of proposed algorithm.

1. Lemma 1 indicates that the output of $\mathcal{A}_{scp}$ is at least a feasible solution to the original problem (**NP**).
2. Lemma 2 has concluded that there will always be a sequence generated by $\mathcal{A}_{scp}$ with any initial point $y_1 \in \eta$.
3. Lemma 3 has proved $\mathcal{A}_{scp}$ is a Descent Iterative Algorithm.
4. Lemma 4 means if the algorithm $\mathcal{A}_{scp}$ terminates at the k-step with tolerance $\epsilon = 0$, then the solution y_k will be the KKT point of **NP**.

In the next, the main theorem is derived from Meyer's theorem [16] and above lemmas.

Theorem 1 *(The main result). Suppose $\mathcal{A}_{scp}$ is uniformly compact on the feasible region ξ and suitable constraint qualification is satisfied. Input any initial point $y_1 \in \eta$, then the output of $\mathcal{A}_{scp}$ will be the KKT point of the original problem **NP**.*

Proof. From Lemma 2, it can be found that there will always be a sequence $\{y_1, y_2, \ldots, y_k, \ldots\}$ generated by $\mathcal{A}_{scp}$.

The output of $\mathcal{A}_{scp}$ includes two conditions:

1. The sequence is finite, that is, suppose the algorithm stops after k iterations, the optimal solution y_{k+1} to **CP**(y_k) is equivalent to y_k; then y_k is output since the terminal condition of $\mathcal{A}_{scp}$ is satisfied. In that way, y_k is a Fixed Point according to the Definition 2 in Appendix A; applying Lemma 4, y_k is the KKT point of **NP**.
2. The sequence is infinite, that is, the output of $\mathcal{A}_{scp}$ is the limiting point of the sequence $\{y_k\}_{k=1}^{\infty}$. Defining $\phi = u_0(y) - \tilde{v}_0(y; y_k)$, then $\mathcal{A}_{scp}$ is strictly monotone with respect to ϕ from Lemma 3; ξ is a convex set, therefore a closed subset of $\mathbb{R}^n$; together with assumptions that $\mathcal{A}_{scp}$ is uniformly compact on the feasible region ξ_k and suitable constraint qualification is satisfied, all the conditions of Theorem 3 in Appendix B are satisfied. It can be concluded that

all limit points of sequence $\{y_k\}$ will be fixed points of $\mathcal{A}_{scp}$,

$$\phi(y_k) \to \phi(y^*) =: \phi^*, \text{ as } k \to \infty,$$

$$\|y_{k+1} - y_k\| \to 0,$$

where y^* is a Fixed Point, therefore y^* must be an optimal solution to the convex programming problem **CP**(y_k) (where k is just a symbol, not determined); as a result, the set $\mathcal{F}(\phi^*) := \{y \in \mathcal{F} : \phi(y) = \phi^*\}$ is finite. So the conclusion can be drawn that the sequence $\{y_k\}_{k=1}^{\infty}$ generated by $\mathcal{A}_{scp}$ converges to a Fixed Point. Applying Lemma 4 again completes the proof. □

Actually, the solution generated by $\mathcal{A}_{scp}$ will not only be a KKT point but also a local optimal point of **NP** if the cost function of the original problem is convex.

Corollary 1. *Let y^* be the last term of the finite sequence $\{y_k\}$ or the limiting point of the infinite sequence $\{y_k\}_{k=1}^{\infty}$ generated by $\mathcal{A}_{scp}$. Then y^* will be at least a local optimal point if the cost function (15) of **NP** is convex.*

Proof. In Theorem 1, y^* has been proved to be the KKT point of **NP**. This corollary will be proved by contradiction. Suppose y^* is not a local optimal of **NP** though the cost function (15) is convex. Then for any arbitrary small $\delta > 0$, there exists a feasible solution β to **NP** with a smaller cost function value in the δ-neighborhood of y^*, that is,

$$u_0(\beta) - v_0(\beta) < u_0(y^*) - v_0(y^*), \quad \|\beta - y^*\| \le \delta. \tag{37}$$

Since β is feasible to **NP**, therefore,

$$u_i(\beta) - v_i(\beta) \le 0, \quad i = 1, 2, \cdots, m. \tag{38}$$

Applying the second-order Taylor expansion of $v_i(\beta)$ at y^*, Eq. (38) becomes

$$u_i(\beta) - \tilde{v}_i(\beta; y^*) - \ddot{v}_i(\beta; y^*) \le 0. \tag{39}$$

Since β is in the δ-neighborhood of y^*, for sufficiently small δ, the second term $\ddot{v}_i(\beta; y^*)$ is so small that can be neglected; therefore, above Eq. (39) can be replaced by

$$u_i(\beta) - \tilde{v}_i(\beta; y^*) \le 0, \tag{40}$$

which means β is feasible to **CP**(y^*) and y^* is the optimal solution to **CP**(y^*) from Theorem 1, so we obtain

$$u_0(\beta) - \tilde{v}_0(\beta; y^*) \ge u_0(y^*) - \tilde{v}_0(y^*; y^*) = u_0(y^*) - v_0(y^*). \tag{41}$$

Since the cost function (15) is convex, it can be derived that the cost function (1) of **POC** is a convex function. Based on CCCP Theorem 2, a proper $u_0(y)$ can be chosen so that $v_0(y)$ is a linear function or equals zero. Hence, the Taylor formula of $v_0(\beta)$ at y^* and $\ddot{v}_0(\beta; y^*) = 0$ imply

$$\begin{aligned} u_0(\beta) - v_0(\beta) &= u_0(\beta) - \tilde{v}_0(\beta; y^*) - \ddot{v}_0(\beta; y^*) \\ &= u_0(\beta) - \tilde{v}_0(\beta; y^*). \end{aligned} \tag{42}$$

Combining Eq. (37) with Eqs. (41), (42), the contradiction

$$\begin{aligned} u_0(\beta) - \tilde{v}_0(\beta; y^*) = u_0(\beta) - v_0(\beta) &< u_0(y^*) - v_0(y^*) \\ &\le u_0(\beta) - \tilde{v}_0(\beta; y^*) \end{aligned} \tag{43}$$

appears, so y^* is a local optimal point of **NP** whose cost function is convex. □

The following corollary presents the global convergence of $\mathcal{A}_{scp}$ on the basis of Theorem 1.

Corollary 2. *$\mathcal{A}_{scp}$ is a globally convergent algorithm.*

Proof. Only infinite sequence is considered in the convergence of the algorithm. From Definition 5 in Appendix A, $\mathcal{A}_{scp}$ is denoted as $\mathcal{A}$, Γ is the set of the KKT points of **NP** and ϕ equals the cost function of **NP**, that is, $\phi(y) = u_0(y) - v_0(y)$. ξ has the same meaning with X; therefore, the triple $\{\mathcal{A}_{scp}, \Gamma, \phi\}$ is defined. From Lemma 3, it can be concluded that $\mathcal{A}_{scp}$ is a Descent Iterative Algorithm in regard to ϕ. From Theorem 1, the infinite sequence generated by $\mathcal{A}_{scp}$ converges to the KKT point of **NP**, i.e. a point of the set Γ, therefore, $\mathcal{A}_{scp}$ is a globally convergent algorithm. □

In this section, some assumptions and lemmas have been proposed as the basis of the main theorem. The main theorem has been rigorously proved so that the KKT point of **NP** can be obtained by the globally convergent algorithm $\mathcal{A}_{scp}$ without calculating the non-convex programming problem. What is more, the KKT point will be at least a local minimum if the cost function of **NP** is convex. The simulation is implemented in the next section to verify the validity of the proposed algorithm.

Table 1
Coefficients for robot route planning problem in nonlinear optimal control model.

Variable	Value	Variable	Value
N	20	Δt	0.75 s
v_{max}	2 m/s	u_{max}	13.33 m^2/s
$c(1)$	$[-1, 0]^T$ m	R(1)	3 m
$c(2)$	$[4, -1]^T$ m	R(2)	1.5 m
x_0	$[-8, -1]^T$ m	x_f	$[8, 1]^T$ m

5. Simulation results

In this section, nonlinear optimal control models of trajectory planning problems with obstacle avoidance are implemented to verify the validity and performance of the proposed algorithm. The simulation is firstly presented in a simple scene, and then is extended to a more complex scene. The proposed algorithm is compared with other non-convex methods as well as some heuristic algorithms. Convex programming problems in the simulation are implemented by CVX, and more information of CVX can found in [17]. Simulation programming operates on a 3.6 GHz personal computer with 8 GB of RAM.

5.1. Scene 1

In this subsection, the proposed algorithm will be demonstrated by a simple model: one robot route planning with obstacle avoidance. The discrete form of the nonlinear optimal control model is applied in the simulation.

$$\min_{u} \quad \sum_{i=1}^{N} \|u(i)\| \tag{44}$$

$$s.t. \quad x(i+1) = x(i) + v(i) \cdot \Delta t,\ i = 1, \ldots, N-1, \tag{45}$$

$$v(i+1) = v(i) + u(i) \cdot \Delta t,\ i = 1, \ldots, N-1, \tag{46}$$

$$\|v(i)\| \le v_{max},\ i = 1, \ldots, N, \tag{47}$$

$$\|u(i)\| \le u_{max},\ i = 1, \ldots, N, \tag{48}$$

$$\|x(i) - c(j)\| \ge R(j),\ i = 1, \ldots, N,\ j = 1, \ldots, N_{obs}, \tag{49}$$

$$x(1) = x_0, x(N) = x_f, \tag{50}$$

where $x(i), v(i), u(i) \in \mathbb{R}^2$ are position, velocity and acceleration variables, respectively. i is the time step and varies from 1 to N and Δt is the sampling time. v_{max} and u_{max} are the upper bound of velocity and acceleration variables, respectively. $c(j) \in \mathbb{R}^2$ is the center of jth obstacle and $R(j) \in \mathbb{R}$ is the radius of jth obstacle. N_{obs} is the number of obstacles. x_0 and x_f are the initial and final position of the robot.

(44) is the cost function only considers acceleration variable. (45) and (46) are system equations of the robot. (47) and (48) are upper bound constraints of velocity and acceleration. (49) are the obstacle avoidance constraints and the main non-convex parts of the model. (50) are initial and finial state boundary constraints. Values of coefficients in the model can be found in Table 1.

In $\mathcal{A}_{scp}$, the non-convex constraint (49) is linearized at the kth iteration solution $x_k(i)$ to acquire the new convex constraint:

$$\|x_k(i) - c(j)\| + \frac{[x_k(i) - c(j)]^T}{\|x_k(i) - c(j)\|}[x(i) - x_k(i)] \ge R(j),$$
$$i = 1, \ldots, N,\ j = 1, \ldots, N_{obs}. \tag{51}$$

Since the original problem is non-convex, the proposed algorithm is heuristic, which means there will be different results for different initial guess points. To verify the convergence of $\mathcal{A}_{scp}$, three different initial guess points are used in the simulation. To obtain feasible initial guess points, the non-convex constraint (49) is replaced by a series of linear constraints and binary logical constraints:

$$[x(i) - c(j)]\sin\left(\frac{l}{2\pi L}\right) + [x(i) - c(j)]\cos\left(\frac{l}{2\pi L}\right)$$
$$\ge R(j)/\cos\left(\frac{\pi}{L}\right) - M \cdot b(i, l),$$
$$i = 1, \ldots, N,\ j = 1, \ldots, N_{obs},\ l = 1, \ldots, L, \tag{52}$$

$$\text{and} \quad \sum_{l=1}^{L} b(i, j, l) \le L - 1,\ i = 1, \ldots, N,\ j = 1, \ldots, N_{obs}. \tag{53}$$

With the new constraints (52) and (53), the robot route planning problem becomes a Mixed Integer Non-Linear Programming (MINLP). The initial guess points are calculated by directly solving MINLP with $L = 6$, $R(1) = 4$ m, 3.25 m, 3.5 m and $R(2) = 2.5$ m, 1.75 m, 2 m in three scenarios. With different initial guess points, the sequence generated by $\mathcal{A}_{scp}$ are all convergent to local optimal trajectories, which are illustrated in Fig. 1. A conclusion can be drawn from Fig. 1 that the proposed algorithm is valid to solve the problem and sensitive to initial guess points. More detailed data of the results can be found in Table 2.

In Table 2, "Initial" means the initial trajectory while "Final" is the final convergent trajectory. "Iterations" is the number of the iteration of the algorithm. "Optimal Value" is the optimal value of the cost function in each scenario. "Time" is the computation time of the programming to obtain the result. Applying direct linearization to the non-convex constraints and directly solving MINLP can obtain feasible trajectories in one iteration, and it can also be implemented to solve the problem. However, binary variables are introduced in (53), which will greatly reduce the computation efficiency. Actually, the direct linearization method is also applied to solve the problem, and the "Optimal Value" is 1.2922, which

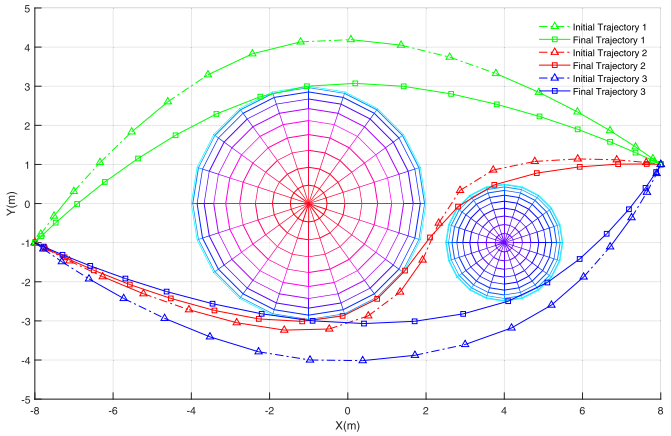

Fig. 1. Robot route planning with obstacle avoidance in three different initial guess points. The dash dotted lines are initial trajectories while the full lines are final convergent trajectories.

Table 2
Results for robot route planning with obstacle avoidance in three different initial guess points.

Scenario		Iterations	Optimal value	Time
1	Initial	1	1.8442	0.61 s
	Final	4	1.2921	21.1 ms
2	Initial	1	1.4223	1.27 s
	Final	5	1.3861	28.4 ms
3	Initial	1	1.5391	0.44 s
	Final	6	1.3005	33.6 ms

Table 3
Coefficients of GPOPS in robot route planning with obstacle avoidance.

(a) Constraint limits in GPOPS.

Limits	t_0	t_f	x	y	v_x, v_y
	0 s	15 s	$[-8; 8]$ m	$[-5; 5]$ m	$[-v_{max}; v_{max}]$ m/s
	u_x, u_y	$path_1$	$path_2$	$path_3$	$path_4$
	$[-u_{max}; u_{max}]$ m^2/s	$[R(1); 15]$ m	$[R(2); 15]$ m	$[0; v_{max}]$ m/s	$[0, u_{max}]$ m^2/s

(b) Initial guess in GPOPS.

Guess	t_0	t_f	x	y	v_x, v_y	u_x, u_y
	0 s	15 s	$[-8; 8]$ m	$[-1; 1]$ m	$[0; 0]$ m/s	$[0; 0]$ m^2/s

(c) Setup setting in GPOPS.

Setting				
	setup.derivatives	setup.autoscale	mesh.tolerance	mesh.iteration
	'finite-difference'	'on'	1e−6	20
	mesh.nodesPerInterval.min		mesh.nodesPerInterval.max	
	4		12	

is almost same as the cost of the optimal solution in $\mathcal{A}_{scp}$; but "Time" is 2.37 s, which is nearly 100 times slower than the proposed method.

Since the pseudospectral method is a popular approach for optimal control problems, it is also implemented to solve the problem as a comparison. The famous General Pseudospectral Optimal Control Software (GPOPS) version 5.0 is selected to solve the problem, more information about GPOPS can be found in [18]. In addition, the continuous model of the problem before discretization is adopted because the model acquired in GPOPS needs to be continuous. The coefficients used in GPOPS are listed in Table 3, and path constraints defined in the model of GPOPS are

$$path_1 = \|x - c(1)\|, \tag{54}$$

$$path_2 = \|x - c(2)\|, \tag{55}$$

$$path_3 = \|v\|, \tag{56}$$

$$path_4 = \|u\|. \tag{57}$$

With the coefficients in Table 3, it spends 125.75 s to calculate the result. The trajectory generated by GPOPS includes 153 nodes and is illustrated in Fig. 2. The output of GPOPS is listed in Table 4.

The meaning of variables in Table 4 is thoroughly discussed in [18]. The value of the cost function is 2.1929, which is quite larger than the results in Table 2; however, it needs to be aware that the cost function of the continuous model is: $\int_{t_0}^{t_f} \|u\| dt$; therefore the cost value is not at the same magnitude with the result of the discrete form. The conclusion can be drawn from Fig. 2 and Table 4 that GPOPS is able to solve the robot route planning problem in continuous nonlinear optimal control model, but the computation efficiency is quite low compared with $\mathcal{A}_{scp}$, which prevents it being utilized in more real-time applications.

In the subsection, the simulation results have verified $\mathcal{A}_{scp}$ can solve the nonlinear optimal control problem with certain convergence. In the example of robot route planning, the computation efficiency is higher than other method. To demonstrate the performance of $\mathcal{A}_{scp}$ further, it is implemented in a more complex scene and compared with more methods.

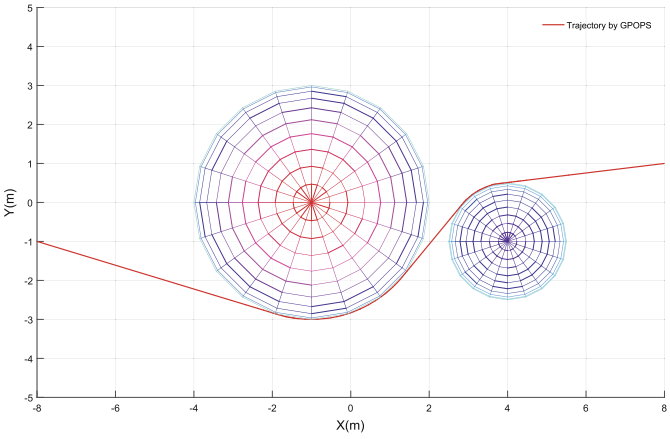

Fig. 2. Trajectory generated by GPOPS in robot route planning problem with obstacle avoidance.

5.2. Scene 2

In this subsection, the UAV path planning problem with obstacle avoidance in nonlinear optimal control model is presented to demonstrate the performance of $\mathcal{A}_{scp}$. The discrete optimal control model of UAV path planning problem is described as follows [19].

$$\min \sum_{i=1}^{N_{max}} weight_u \|u(i)\|_1 + weight_v \|v(i)\|_1 + weight_f \|x(i) - x_f\|_1 \tag{58}$$

$$s.t. \quad x(i+1) = x(i) + v(i) \cdot \Delta t, \ i = 1, \ldots, N_{max} - 1, \tag{59}$$

$$v(i+1) = v(i) + [u(i) + g] \cdot \Delta t, \ i = 1, \ldots, N_{max} - 1, \tag{60}$$

Table 4
Output of GPOPS in robot route planning with obstacle avoidance.

No. of iterations	5640	Objective value	2.1929181777E+00
No. of major iterations	588	Linear objective	0.0000000000E+00
Penalty parameter	1.191E+06	Nonlinear objective	2.1929181777E+00
No. of calls to funobj	1819	No. of calls to funcon	1819
No. of superbasics	299	No. of basic nonlinears	683
No. of degenerate steps	348	Percentage	6.17
Max x	1 5.0E−01	Max pi	1218 1.0E+00
Max Primal infeas	0 0.0E+00	Max Dual infeas	498 1.7E−01
Nonlinear constraint violn	3.4E−07	Total Time	1.2575E+02

Table 5
Coefficients for UAV path planning problem.

Variable	Value	Variable	Value
N_{max}	100	Δt	1 s
g	$[0, 0, -9.81]^T$ m^2/s	$weight_u$	0.01
$weight_v$	1	$weight_f$	1
v_{max}	50 m/s	u_{max}	10 m^2/s
$\hat{n}$	$[0, 0, 1]^T$	θ_{cone}	30°

$$\|v(i)\|_\infty \le v_{max},\ i = 1, \ldots, N_{max}, \tag{61}$$

$$\|u(i)\|_\infty \le u_{max},\ i = 1, \ldots, N_{max}, \tag{62}$$

$$\hat{n}^T u(i) \ge \|u(i)\| cos(\theta_{cone}),\ i = 1, \ldots, N_{max}, \tag{63}$$

$$\|Hx(i) - c(j)\| \ge R(j),\ i = 1, \ldots, N_{max},\ j = 1, \ldots, N_{obs}, \tag{64}$$

$$x(1) = x_0, v(1) = v_0, \tag{65}$$

where $x(i), v(i), u(i) \in \mathbb{R}^3$ are position, velocity and acceleration variables, respectively. i is the time step and varies from 1 to N_{max}. N_{max} is the maximum time step and set big enough. $weight_u$, $weight_v$ and $weight_f$ are weight coefficients which determine the main certain of the cost function. $g \in \mathbb{R}^3$ is the gravity vector. $\hat{n} \in \mathbb{R}^3$ is a unit vector and θ_{cone} is an angle used to define a cone. H is a constant matrix and $Hx(i)$ represents the horizontal coordinates of position. v_0 is the initial velocity of UAV. The remaining variables have the same meaning as those of the simulation model in previous subsection.

(58) is the cost function and $\|\cdot\|_1$ is the l_1 norm. Although the l_1 norm is nonlinear and discontinuous, it can be converted to linear form by introducing new supplementary variables [20]; therefore, Eq. (58) is regarded as a linear cost function. (59) and (60) are system equations of UAV. (61) and (62) are upper bound constraints of velocity and acceleration, where $\|\cdot\|_\infty$ is the l_∞ norm. Because Eqs. (61), (62) are equivalent to a series of linear constraints; thus, they are regarded as linear constraints. (63) is a thrust cone constraint which limits the thrust vector in a cone defined by θ_{cone} and $\hat{n}$, and it is a common convex constraint in flight dynamics. (64) are the obstacle avoidance constraints and the main non-convex parts of the model. (65) are initial boundary conditions. Values of coefficients in the model can be found in Table 5.

The same as Scene 1, (64) is linearized at the kth iteration solution $x_k(i)$ to obtain the new convex constraint:

$$\|Hx_k(i) - c(j)\| + \frac{[Hx_k(i) - c(j)]^T H}{\|Hx_k(i) - c(j)\|}[x(i) - x_k(i)] \ge R(j),$$
$$i = 1, \ldots, N_{max}, j = 1, \ldots, N_{obs}, \tag{66}$$

where

$$H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

The simulation area is $2000 \times 900 \times 300$ (m^3), including 24 threatening obstacles with different radii, which is illustrated in Fig. 3. The detailed information of the obstacles is listed in Table 6. Besides, the terrain constraints are neglected so that the size of vector-graphs can be reduced and the results of the simulation can be demonstrated more clearly. The simulation is firstly implemented in three different boundary conditions to verify the feasibility and convergence of $\mathcal{A}_{scp}$ and the conditions are listed in Table 7. Trajectories generated by $\mathcal{A}_{scp}$ are illustrated in Fig. 4 and the results are listed in Table 8, where "Iteration", "Optimal Value" and "Time" have the same meaning as the ones in Table 2. In the model, the maximum time step N_{max} is big enough and weight coefficients $weight_u$, $weight_v$, $weight_f$ are proper, and then "Node" N is the time step that the UAV reaches the final point x_f at the first time. With cost function (58), the UAV will stay at the final point, which means it will hover or stop at the position x_f, and the velocity and acceleration of the UAV will equal zero at the remaining $(N_{max} - N)$ time step. As a result, the value of cost function do not increase anymore.

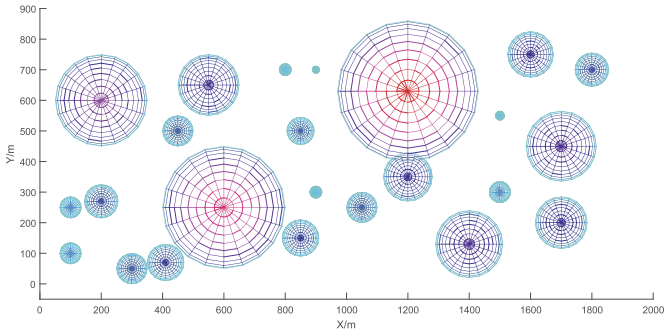

Fig. 3. Threatening obstacles with different radii in UAV path planning problem.

From trajectories in Fig. 4, it can be concluded that UAV path planning problem in nonlinear optimal control model can be solved by $\mathcal{A}_{scp}$ with certain convergence.

To verify the optimality of the trajectory generated by $\mathcal{A}_{scp}$, the non-convex problem is directly solved by CPLEX. CPLEX is a mature business software integrated many algorithms to solve optimization problems and more information can be found in [21]. Since the original problem is non-convex, the results calculated by CPLEX are various for different cases. For some cases, the results are convergent while others may fail to get a feasible solution. In this scene, the boundary condition is $x_0 = [0, 0, 0]^T$ m, $x_f = [2000, 600, 0]^T$ m and $v_0 = [0, 0, 0]^T$ m/s. Five convergent results generated by CPLEX with different initial cases are compared with the trajectory obtained from $\mathcal{A}_{scp}$ in Fig. 5. Some trapped results are illustrated in Fig. 6 and the details can be found in Table 9.

In Fig. 5, Case 1–Case 5 are the convergent results generated by CPLEX. For results of CPLEX, the best trajectory is Case 5 (minimum optimal value). It can be noticed that the trajectory generated by $\mathcal{A}_{scp}$ is smoother and shorter than other results. What is more, the calculating time of $\mathcal{A}_{scp}$ is nearly invariable at repeated experiments; however, that of CPLEX varies greatly, which suffers from the uncertainty in non-convex programming. Because solving con-

Table 6
Data of obstacles in UAV path planning problem.

No.	Center (m)	Radius (m)	No.	Center (m)	Radius (m)	No.	Center (m)	Radius (m)
1	$[600, 250]'$	200	2	$[100, 100]'$	35	3	$[800, 700]'$	20
4	$[850, 500]'$	45	5	$[100, 250]'$	35	6	$[450, 500]'$	50
7	$[200, 600]'$	150	8	$[900, 700]'$	12	9	$[550, 650]'$	100
10	$[200, 270]'$	55	11	$[1200, 630]'$	230	12	$[900, 300]'$	20
13	$[1500, 550]'$	15	14	$[1700, 200]'$	85	15	$[1050, 250]'$	50
16	$[850, 150]'$	60	17	$[1600, 750]'$	75	18	$[1400, 130]'$	110
19	$[1700, 450]'$	115	20	$[1500, 300]'$	35	21	$[1800, 700]'$	55
22	$[300, 50]'$	50	23	$[1200, 350]'$	80	24	$[410, 70]'$	60

Table 7
Different boundary conditions in UAV path planning problem.

Condition	1	2	3
x_0 (m)	$[0, 0, 0]^T$	$[0, 850, 0]^T$	$[0, 400, 0]^T$
x_f (m)	$[2000, 850, 0]^T$	$[2000, 0, 0]^T$	$[2000, 450, 0]^T$
v_0 (m/s)		$[0, 0, 0]^T$	

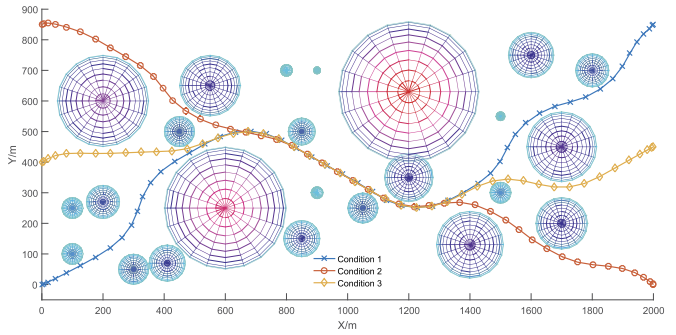

Fig. 4. Trajectories of UAV path planning problem with different boundary conditions.

Table 8
Simulation results for UAV path planning problem with different boundary conditions.

Condition	Iteration	Node	Optimal value	Time
1	4	48	4599.8	3154.0 ms
2	5	53	5014.0	5456.5 ms
3	4	46	4410.0	2926.7 ms

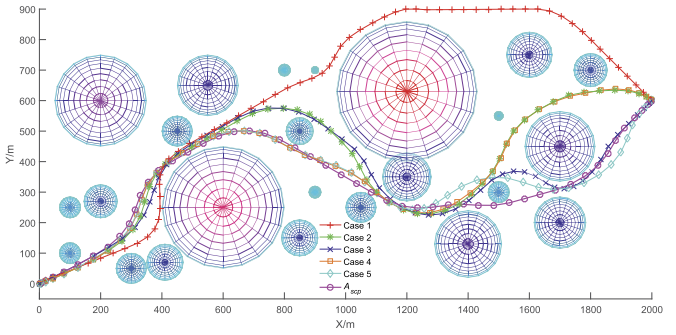

Fig. 5. The convergent trajectories generated by CPLEX and $\mathcal{A}_{scp}$. The pink one with circle sign is generated by $\mathcal{A}_{scp}$, and others are the convergent results of CPLEX with different initial cases. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

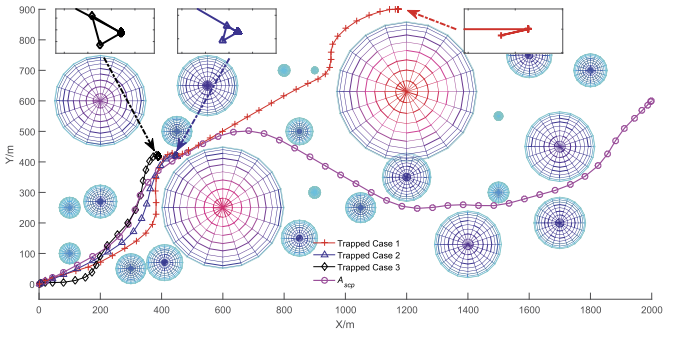

Fig. 6. The trapped trajectories generated by CPLEX with different initial cases. The pink one with circle sign is generated by $\mathcal{A}_{scp}$. The details of trapped points are magnified on the top of the figure.

Table 9
Simulation results for $\mathcal{A}_{scp}$ and CPLEX with different initial cases.

Convergent results.

Case	Node	Optimal value
1	67	6671.2
2	56	5472.4
3	59	5416.9
4	53	5146.4
5	60	5120.5
$\mathcal{A}_{scp}$	51	4906.3

Trapped results.

Case	Trapped node	Trapped point (m)
1	49	[1172.55, 900.00, 44.13]
2	19	[449.37, 418.28, 48.91]
3	24	[389.40, 418.36, 45.19]

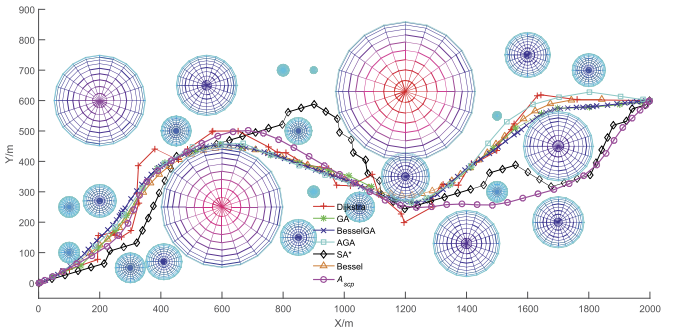

Fig. 7. Trajectories generated by $\mathcal{A}_{scp}$ and heuristic algorithms in UAV path planning problem. The pink one with circle sign is generated by $\mathcal{A}_{scp}$.

vex programming problem has a bounded time but non-convex programming do not have such results.

From Table 9(a), $\mathcal{A}_{scp}$ acquires the least "Node" with minimum "Optimal Value", that is, has the best performance, which also can be seen from Fig. 5. Because the calculating time of CPLEX to solve the non-convex problem with different initial cases varies greatly, therefore is not compared in Table 9(a). In Table 9(b), it can be found that the UAV is trapped in the process of flying to final point in different initial cases. From magnified parts of Fig. 5, Trapped Case 2 and Trapped Case 3 repeat last three points of the trajectory and Trapped Case 1 stay at the last point (1172.55, 900.00, 44.13) unchanged. Though more feasible results are listed in the Table 9, most results are trapped during actual solution process. Directly solving non-convex programming problems through non-convex algorithms does not have a certain convergence with bounded calculating time, but the proposed algorithm $\mathcal{A}_{scp}$ is effectively convergent to a KKT point with a better performance.

Compared to the results of CPLEX, the superiority of $\mathcal{A}_{scp}$ is verified. To ensure the effectiveness and performance further, $\mathcal{A}_{scp}$ is compared with heuristic algorithms: Dijkstra, Genetic Algorithm (GA), Bessel+Genetic Algorithm (BesselGA), Adaptive Genetic Algorithm (AGA), Simulated Annealing+A* (SA*) and Dijkstra+Bessel (Bessel) [22]. The simulation is implemented with the same boundary condition in Fig. 5, and the results are illustrated in Fig. 7.

From Fig. 7, it can be found that the trajectory generated by $\mathcal{A}_{scp}$ is much smoother and farther from the obstacles. The detailed results can be found in Table 10. From Table 10, GA and AGA use the least "Node" in the simulation. Dijkstra and Bessel algorithm use less time to solve the problem; however, the trajectory of Dijkstra is quite poor and some parts of Bessel are infeasible from Fig. 7. The cost function (58) considering distance to final point, velocity and acceleration of UAV, therefore, the less optimal value means trajectory with better performance. As a result,

Table 10
Simulation results for $\mathcal{A}_{scp}$ and heuristic algorithms in UAV path planning problem.

Algorithm	Node	Time	Optimal value
Dijkstra	42	*1871.3* ms	8396.5
GA	*30*	4537.9 ms	6628.4
BesselGA	113	5296.2 ms	5673.7
AGA	*30*	4896.2 ms	6043.2
SA*	61	3968.9 ms	7261.3
Bessel	50	2866.4 ms	7546.8
$\mathcal{A}_{scp}$	51	3372.1 ms	*4906.3*

the conclusion can be drawn that $\mathcal{A}_{scp}$ has a better performance than other methods. What is more, the convergence of $\mathcal{A}_{scp}$ can be confirmed before calculating the trajectory, but the convergence of heuristic algorithms can not be guaranteed. Therefore, the convergence and effectiveness of proposed algorithm is verified in this subsection again.

In this section, the performance of $\mathcal{A}_{scp}$ is verified by two scenes. Compared with other methods, the superiority of the proposed algorithm is demonstrated. However, there are some problems that need to be considered in $\mathcal{A}_{scp}$.

1. How to find a feasible initial point in a more complex non-convex feasible region.
2. Can the algorithm start with an infeasible initial guess and output convergent results.
3. Since $\mathcal{A}_{scp}$ is sensitive to initial guess, the convergent result will be the nearest local optimal point of the initial guess. How to jump out of the local optimal.

Above questions will be the main concerns in our future research.

6. Conclusion

In this paper, the UAV path planning problem was founded in a nonlinear optimal control model with non-convex constraints. Then, the Sequential Convex Programming Algorithm was proposed to solve the optimization problem, in which the non-convex programming problem was approximated by a series of sequential convex programming problems through first order Taylor formula. In the next, the proposed algorithm was rigorously proved to be globally convergent with the output being the KKT point of the original nonlinear problem under mild condition. Finally, two simulation scenes were applied to evaluate the convergence and effectiveness of the proposed method step by step. In comparison to reference algorithms, $\mathcal{A}_{scp}$ performed better in trajectory planning problems with threatening obstacles.

Conflict of interest statement

There is no conflict of interest.

Acknowledgements

This work was jointly supported by National Natural Science Foundation of China (61673265); 973 Project (6133190302); Aeronautical Science Foundation of China (20140157001); 2015 Industry-University-research cooperation project of AVIC.

Appendix A. Definitions

For proving the lemmas and the main theorem, some useful definitions and theorems are introduced from [23].

Definition 1 *(Point-to-set map).* X and Y are two prescribed sets, a map Φ assigning any element x of X to a subset $\Phi(x)$ of Y is called a **point-to-set map**, that is, $\Phi : X \to \mathcal{P}(Y)$, where $\mathcal{P}(Y)$ defines the power set of Y.

Definition 2 *(Fixed point).* A **fixed point** of a point-to-set map $\Phi : X \to \mathcal{P}(X)$ is defined as a point x satisfying $\{x\} = \Phi(x)$.

Definition 3 *(Iterative algorithm).* X is a prescribed set and x_0 is a known element of X. A point-to-set map $\mathcal{A} : X \to \mathcal{P}(X)$ iterating a sequence $\{x_k\}_{k=0}^{\infty}$ and satisfying

$$x_{k+1} \in \mathcal{A}(x_k),\ k = 0, 1, \ldots \tag{A.1}$$

is regarded as an **iterative algorithm**.

$\Gamma \subset X$ is introduced to represent a solution set of $\mathcal{A}$. For example, the minimizers of a convex programming or a set of KKT points of a nonlinear programming is a solution set.

Definition 4 *(Monotone algorithm).* Given a continuous function $\phi : X \to \mathbb{R}$ if any $y \in \mathcal{A}(x)$ implies $\phi(y) \leq \phi(x)$ then the iterative algorithm $\mathcal{A}$ is called to be **monotone** in regard to ϕ, what's more, $\mathcal{A}$ is called to be **strictly monotone** if and only if $y \in \mathcal{A}(x)$ and $\phi(y) \leq \phi(x)$ implies $x = y$.

Definition 5 *(Globally convergent).* A triple $\{\mathcal{A}, \Gamma, \phi\}$ is a prescribed iterative descent algorithm on the set X, where $\mathcal{A} : X \to \mathcal{P}(X)$ is a monotone iterative algorithm in regard to ϕ and Γ is a solution set of $\mathcal{A}$. For any initial point x_0 of X, the iterative algorithm $\mathcal{A}$ can generate a sequence $\{x_k\}_{k=0}^{\infty}$ and the sequence converges to a point of the set Γ, which implies that the necessary optimality condition is satisfied. Then the algorithm $\mathcal{A}$ is defined to be **globally convergent**.

Appendix B. Theorems

Theorem 2 *(The concave convex procedure [24]). Any twice differentiable function $f(x)$ with bounded Hessian $\partial^2 f(x)/\partial x^2$ can be reformulated as a difference of two strictly convex functions.*

Theorem 3 *(Meyer, 1976, Theorem 3.1, Corollary 3.2 [16]). For a triple $\{\mathcal{A}, \Gamma, \phi\}$, assuming $\mathcal{A} : X \to \mathcal{P}(X)$ is uniformly compact and closed. X is subset of $\mathbb{R}^n$. Then all limited points of the sequence $\{x_k\}_{k=0}^{\infty}$ generated by $\mathcal{A}$ will be the element of Γ. $\phi(x_k) \to \phi(x_*) =: \phi^*$, as $k \to \infty$, $\|x_{k+1} - x_k\| \to 0$, where x_* is a fixed point, and either $\{x_k\}_{k=0}^{\infty}$ converges or the set of fixed points of $\{x_k\}_{k=0}^{\infty}$ is connected. Define $\mathcal{F}(a) := \{x \in \mathcal{F} : \phi(x) = a\}$, where $\mathcal{F}$ is the set of fixed points of $\mathcal{A}$. If $\mathcal{F}(\phi^*)$ is finite, then any sequence $\{x_k\}_{k=0}^{\infty}$ generated by $\mathcal{A}$ converges to some x_* in $\mathcal{F}(\phi^*)$.*

References

[1] F. Nex, F. Remondino, UAV for 3D mapping applications: a review, Appl. Geomat. 6 (1) (2014) 1–15.
[2] J.L. Vian, J.R. Moore, Trajectory optimization with risk minimization for military aircraft, AIAAJ Guid. Control Dyn. 12 (3) (1989) 311–317.
[3] A. Atyabi, D.M. Powers, Review of classical and heuristic-based navigation and path planning approaches, Int. J. Adv. Comput. Technol. 5 (14) (2013) 1.
[4] O. Souissi, R. Benatitallah, D. Duvivier, A. Artiba, N. Belanger, P. Feyzeau, Path planning: a 2013 survey, in: Proceedings of 2013 International Conference on Industrial Engineering and Systems Management, IESM, IEEE, 2013, pp. 1–8.
[5] J.T. Betts, Survey of numerical methods for trajectory optimization, AIAA J. Guid. Control Dyn. 21 (2) (1998) 193–207.
[6] G. Winter, J. Periaux, M. Galan, P. Cuesta, Genetic Algorithms in Engineering and Computer Science, John Wiley & Sons, Inc., 1996.
[7] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, vol. 4, IEEE, 1995, pp. 1942–1948.
[8] K. Teo, L. Jennings, H. Lee, V. Rehbock, The control parameterization enhancing transform for constrained optimal control problems, J. Aust. Math. Soc. 40 (03) (1999) 314–335.

[9] L.S. Jennings, K.L. Teo, A computational algorithm for functional inequality constrained optimization problems, Automatica 26 (2) (1990) 371–375.
[10] L. Jennings, K. Teo, C. Goh, MISER3 v.2: Optimal Control Software, Theory and User Manual, Department of Mathematics, University of Western Australia, Australia, 1997.
[11] D.P. Bertsekas, Nonlinear Programming, Athena Scientific, Belmont, 1999.
[12] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.
[13] X. Liu, P. Lu, Solving nonconvex optimal control problems by convex optimization, J. Guid. Control Dyn. 37 (3) (2014) 750–765.
[14] J. Nocedal, S. Wright, Sequential Quadratic Programming, Springer New York, New York, NY, 2006, Ch. 11, 12 and 18.
[15] X. Yu, Y. Zhang, Sense and avoid technologies with applications to unmanned aircraft systems: review and prospects, Prog. Aerosp. Sci. 74 (Supplement C) (2015) 152–166, https://doi.org/10.1016/j.paerosci.2015.01.001.
[16] R.R. Meyer, Sufficient conditions for the convergence of monotonic mathematicalprogramming algorithms, J. Comput. Syst. Sci. 12 (1) (1976) 108–121.
[17] M. Grant, S. Boyd, CVX: matlab software for disciplined convex programming, version 2.1, http://cvxr.com/cvx, 2014.
[18] A.V. Rao, D.A. Benson, C. Darby, M.A. Patterson, C. Francolin, I. Sanders, G.T. Huntington, Algorithm 902: GPOPS, a Matlab software for solving multiple-phase optimal control problems using the Gauss pseudospectral method, ACM Trans. Math. Softw. 37 (2) (2010) 22.
[19] Z. Zhang, J. Wang, J. Li, X. Wang, UAV path planning based on receding horizon control with adaptive strategy, in: 29th Chinese Control and Decision Conference, CCDC, 2017, pp. 843–847.
[20] A. Richards, T. Schouwenaars, J.P. How, E. Feron, Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming, AIAA J. Guid. Control Dyn. 25 (4) (2002) 755–764.
[21] I.I. CPLEX, V12.1: user's manual for CPLEX, International Business Machines Corporation 46 (53) (2009) 157.
[22] H. Tao, Z. Wang, J. Li, Three-dimensional path planning for unmanned aerial vehicles based on multi-objective genetic algorithm, in: Proceedings of the 33rd Chinese Control Conference, 2014, pp. 8617–8621.
[23] W.I. Zangwill, Nonlinear Programming: A Unified Approach, Prentice–Hall, Englewood Cliffs, Upper Saddle River, NJ, 1969.
[24] A.L. Yuille, A. Rangarajan, The concave–convex procedure, Neural Comput. 15 (4) (2003) 915–936.