Aerospace Science and Technology 106 (2020) 106134

Contents lists available at ScienceDirect

Aerospace Science and Technology

www.elsevier.com/locate/aescte

Generalized quasi-spectral model predictive static programming method using Gaussian quadrature collocation

Cong Zhou [a,b], Xiaodong Yan [a,b,*], Shuo Tang [a,b]

[a] School of Astronautics, Northwestern Polytechnical University, Xi'an, Shaanxi, 710072, People's Republic of China
[b] Shaanxi Aerospace Flight Vehicle Design Key Laboratory, Northwestern Polytechnical University, Xi'an, Shaanxi, 710072, People's Republic of China

ARTICLE INFO

Article history:
Received 21 January 2020
Received in revised form 10 June 2020
Accepted 6 August 2020
Available online 13 August 2020
Communicated by Efstathios Bakolas

Keywords:
Model predictive static programming (MPSP)
Quasi-spectral
Gaussian quadrature collocation
Trajectory planning

ABSTRACT

A new generalized quasi-spectral model predictive static programming (GS-MPSP) method is proposed to efficiently solve a class of terminal-constrained optimal control problems with specified or free terminal time. A spectral representation method is used to model the profile of the control vector, then an infinite-dimensional optimization problem in a continuous-time framework is transformed into a small-dimensional static programming problem minimizing a certain performance index. Using the Gauss quadrature collocation method, the computation of the sensitivity matrix can be converted to the solution of a group of linear equations and algebraic summation at a few collocation nodes. Subsequently, the spectral coefficients and terminal time are efficiently obtained to eliminate terminal output deviations by solving the static programming problem. A simulation case with a scenario of intercepting a high-speed target with a specified impact angle in the midcourse phase was conducted. The results indicate that the proposed GS-MPSP approach has increased computational efficiency compared to traditional methods.

1. Introduction

The task of trajectory planning and guidance for aerospace vehicles is complicated when accounting for various constraints as well as expected objectives. Compared with analytical methods such as PN guidance, numerical methods can directly apply nonlinear models, consider more constraints, and require less control effort. Over the last few decades, many numerical methods of such ability were developed, such as the shooting method [1], the gradient method [2], the pseudospectral method [3,4]. These methods exhibit excellent performance, but are known to suffer from computational complexity and cannot be used to obtain solutions in real-time. In light of present trends, new computational guidance methods have been proposed, which rely extensively on onboard numerical computation to generate guidance commands [5]. Due to the fact that the onboard computational capability is extremely limited, the computational efficiency of the numerical methods is vital for its application.

One typical approach for this demand is model predictive static programming (MPSP) [6,7], which combines the philosophies of nonlinear model predictive control (MPC) theory [8–10] and approximate dynamic programming (ADP) [11]. This method features an explicit closed-form solution and avoids the numerical complexities of optimal control theory. As a result, this method exhibits relatively high computational efficiency and thus has the potential for online application. To date, this method has been widely applied to various guidance problems [12–20]. As well, various extensions and developments for this method have been proposed. Generalized Model Predictive Static Programming (GMPSP) [21] extended the solution process and principles of MPSP to a continuous-time framework. It provides the advantage that any higher-order numerical scheme can easily be incorporated into the solution process. Tracking-Model Predictive Static Programming (T-MPSP) was introduced by [22,23], to solve the reference command tracking problem. Pan et al. [24] gives an essential insight that the MPSP approach is just the application of the classic Newton method. Based on this understanding, the adaptive damped Newton method has been applied to the numerical solution of MPSP to improve its convergence robustness. Mondal and Padhi [25] extended the MPSP approach to solving the state and input constrained guidance problem, which is converted to a convex programming problem and solved using Interior-Point method. Similarly, Hong et al. [26] directly used the discrete state and control to formulate the convex programming problem and in which a flexible cost function can be used. This algorithm is referred to as the Model Predictive Convex Programming (MPCP). Recently, the Unscented Model Predictive Static Programming (U-MPSP) method

* Corresponding author at: School of Astronautics, Northwestern Polytechnical University, Xi'an, Shaanxi, 710072, People's Republic of China.
E-mail addresses: zhoucongnwpu@163.com (C. Zhou), yan804@nwpu.edu.cn (X. Yan), stang@nwpu.edu.cn (S. Tang).

https://doi.org/10.1016/j.ast.2020.106134
1270-9638/

was proposed in [27] to solve a class of problems with parameters and/or initial conditions uncertainties in time-invariant systems.

Throughout these developments, research has been mainly aimed to extend the ability to deal with more complex problem and consider more constraints. However, the requirement for sufficiently high computational efficiency for online application is still a challenging issue. From the fundamental point of view, for MPSP, the barrier of computation efficiency lies in the fact that this method is formulated by Euler discretization. Therefore, to ensure accuracy, this method needs to take a large number of discrete nodes, which leads to large computation have to be performed on these nodes. This is especially important in the computation of sensitivity matrix. Large matrix computations are time-consuming. In fact, it is the most time-consuming part of an MPSP approach, even when a recursive method is used. One attempt to address this was Quasi-Spectral MPSP (QS-MPSP) technique [28], which expresses the control variables as a weighted sum of basic functions and optimizes the weighted coefficients instead of the control variable itself at every grid point. The number of optimization variables is reduced leading to improved computational efficiency. However, this method still requires the discretization of the system dynamics and computation of the sensitivity matrix recursively at every discrete node. Consequently, the improvement of computational efficiency is limited.

Considering this, the present paper tried to put forward a higher computational efficiency method by further developing the QS-MPSP method [28]. Additionally, the case with the free terminal time is also considered in this work. For these purposes, the control profile is modeled using a spectral representation method and the terminal time is used as an optional variable to be optimized. Subsequently, a sensitivity relation is derived, which formulates the relationship between the final output deviations and the spectral coefficients as well as the terminal time. Based on this formulation, a static programming problem is created to solve the updated spectral coefficients and terminal time. The prerequisite to solving this static programming problem is to work out the spectral sensitivity matrix, which plays a similar role as the sensitivity matrix in MPSP approach but is represented in a continuous-time framework. By applying the Gaussian quadrature collocation method, the computation of spectral sensitivity matrix can be converted to solving a group of linear equations and algebraic summation at very fewer collocation nodes. Compared with the recursive computation of the sensitivity matrix in each discrete step in present MPSP approach, the present calculation strategy significantly improves the computational efficiency. It should be pointed out that this method is quite different from the present MPSP or QS-MPSP method in three ways. First, the update of coefficients are formulated by the spectral sensitivity matrix, which is represented in a continuous-time framework. Secondly, the spectral sensitivity matrix is computed using a Gaussian quadrature collocation rather than a recursive manner in each discrete step. Thirdly, the terminal time is used as an optional updated variable and therefore it can be extended to solve the problem with free time. This new technique is referred to as the Generalized Quasi-Spectral MPSP (GS-MPSP), to maintain compatibility with the existing literature [23,29]. Due to the efficiency of the sensitivity matrix computation as well as of the improved static programming, the proposed method has superiority computational efficiency compared to the existing MPSP approach. Additionally, it is able to solve the problem with free terminal time, and the advantages of QS-MPSP, such as smooth control, are maintained. As a result, the present approach may enhance the possibility of online application.

The performance of the proposed method is demonstrated in a scenario of intercepting a high-speed target with a specified impact angle constraint in the midcourse phase, both considering the case of fixed terminal time and free terminal time. A comparison of the proposed technique with the QS-MPSP, MPSP and MPCP methods is also conducted. The results indicate the effectiveness of the proposed GS-MPSP method to guarantee almost same performance and ability to satisfy the terminal conditions as MPSP in the case of fixed terminal time, and produce the better solution in the case of free terminal time. Meanwhile, a significantly increased computational efficiency (an approximate 7 times improvement) is achieved compared to the present MPSP methods.

2. Mathematical formulation of the GS-MPSP approach

This section presents the proposed GS-MPSP method for a class of nonlinear system. The considered nonlinear system dynamic are as follows:

$$\dot{\boldsymbol{X}}(t) = \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}, t) \tag{1}$$

$$\boldsymbol{Y}(t) = \boldsymbol{h}(\boldsymbol{X}, t) \tag{2}$$

where $\boldsymbol{X} \in \boldsymbol{R}^n$, $\boldsymbol{U} \in \boldsymbol{R}^m$ and $\boldsymbol{Y} \in \boldsymbol{R}^p$ denote the state, control and output vectors, respectively. The objective of this method is to find suitable control history $\boldsymbol{U}(t)$ so that the final system output $\boldsymbol{Y}_f(t_f)$ approaches the desired value $\boldsymbol{Y}_d$ with minimum control effort as shown in Eq. (3). At the same time, the terminal time $\boldsymbol{t}_f$ is considered to be unspecified. That means the algorithm also needs search the proper terminal time.

$$J = \int_{t_0}^{t_f} \left[\boldsymbol{U}^T(t) \boldsymbol{R}(t) \boldsymbol{U}(t) \right] \cdot dt \tag{3}$$

To develop this method, a sensitivity relation, that involves the control increments and the terminal error of the continuous system given in Eqs. (1) and (2) is needed. Therefore, it is first introduced in subsection 2.1. Then based on this relationship, the proposed GS-MPSP is derived in subsection 2.2.

2.1. The sensitivity relation for continuous system with unspecified terminal time

In this subsection, a sensitivity relation for the continuous nonlinear system is formulated. It is partly different from the sensitivity relation developed in Ref. [21], by considering the differential change of terminal time. For completeness, the brief derivation is provided in the following.

First, introducing Eq. (1), the final output vector of the system can be expressed as

$$\boldsymbol{Y}(\boldsymbol{X}(t_f)) = \boldsymbol{Y}(\boldsymbol{X}(t_f)) + \int_{t_0}^{t_f} \boldsymbol{W}(t) \cdot \left[\boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}, t) - \dot{\boldsymbol{X}} \right] dt \tag{4}$$

Where $\boldsymbol{W}(t) \in \boldsymbol{R}^{p \times n}$ is a weighting matrix. The differential of Eq. (4) considering the differential change in the terminal time, $\boldsymbol{t}_f$, is

$$\begin{aligned} d\boldsymbol{Y}(\boldsymbol{X}(t_f)) = & \frac{\partial \boldsymbol{Y}(\boldsymbol{X}(t))}{\partial \boldsymbol{X}(t)} \cdot d\boldsymbol{X}(t_f) \\ & + \int_{t_0}^{t_f} \left[\boldsymbol{W}(t) \cdot \frac{\partial \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}, t)}{\partial \boldsymbol{X}(t)} \cdot \delta \boldsymbol{X}(t) \right. \\ & \left. + \boldsymbol{W}(t) \cdot \frac{\partial \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}, t)}{\partial \boldsymbol{U}(t)} \cdot \delta \boldsymbol{U}(t) - \boldsymbol{W}(t) \cdot \delta \dot{\boldsymbol{X}}(t) \right] dt \end{aligned} \tag{5}$$

Integrating the last term on the right side of Eq. (5) by parts, yielding

$$
\begin{aligned}
d\boldsymbol{Y}(\boldsymbol{X}(t_f)) = & \frac{\partial \boldsymbol{Y}(\boldsymbol{X}(t_f))}{\partial \boldsymbol{X}(t_f)} \cdot d\boldsymbol{X}(t_f) - [\boldsymbol{W}(t) \cdot \delta \boldsymbol{X}(t)]_{t=t_f} \\
& + [\boldsymbol{W}(t) \cdot \delta \boldsymbol{X}(t)]_{t=t_0} \\
& + \int_{t_0}^{t_f} \left[\left(\boldsymbol{W}(t) \cdot \frac{\partial \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}, t)}{\partial \boldsymbol{X}(t)} + \dot{\boldsymbol{W}}(t) \right) \cdot \delta \boldsymbol{X}(t) \right. \\
& \left. + \boldsymbol{W}(t) \cdot \frac{\partial \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}, t)}{\partial \boldsymbol{U}(t)} \cdot \delta \boldsymbol{U}(t) \right] dt
\end{aligned} \tag{6}
$$

Note that in Eq (5) and Eq. (6), $d\boldsymbol{X}(t_f)$ denote the differential of $\boldsymbol{X}(t_f)$ taking into account the differential change of terminal time, and $\delta \boldsymbol{X}$ denotes the variation in $\boldsymbol{X}$ when the terminal time is assumed to be fixed. According to Ref. [30], they have the relationship as follow

$$
d\boldsymbol{X}(t_f) = \delta \boldsymbol{X}(t_f) - \dot{\boldsymbol{X}}(t_f) \cdot dt_f \tag{7}
$$

Substituting $\delta \boldsymbol{X}(t_f)$ from Eq. (7) into Eq. (6) and collecting terms give

$$
\begin{aligned}
d\boldsymbol{Y}(\boldsymbol{X}(t_f)) = & \left(\frac{\partial \boldsymbol{Y}(\boldsymbol{X}(t_f))}{\partial \boldsymbol{X}(t_f)} - \boldsymbol{W}(t_f) \right) \cdot d\boldsymbol{X}(t_f) \\
& + [\boldsymbol{W}(t) \cdot \delta \boldsymbol{X}(t)]_{t=t_0} \\
& + \int_{t_0}^{t_f} \left[\left(\boldsymbol{W}(t) \cdot \frac{\partial \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}, t)}{\partial \boldsymbol{X}(t)} + \dot{\boldsymbol{W}}(t) \right) \cdot \delta \boldsymbol{X}(t) \right. \\
& \left. + \boldsymbol{W}(t) \cdot \frac{\partial \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}, t)}{\partial \boldsymbol{U}(t)} \cdot \delta \boldsymbol{U}(t)\, dt \right] \\
& - \frac{\partial \boldsymbol{Y}(\boldsymbol{X}(t_f))}{\partial \boldsymbol{X}(t_f)} \cdot \dot{\boldsymbol{X}}(t_f) \cdot dt_f
\end{aligned} \tag{8}
$$

Here, $\boldsymbol{W}(t)$ must be chosen in a way that eliminates the coefficients of $\delta \boldsymbol{X}(t)$ and $d\boldsymbol{X}(t_f)$ in Eq. (8), which leads to the following weighting matrix dynamics with the associated boundary condition at the final time t_f:

$$
\dot{\boldsymbol{W}}(t) = -\boldsymbol{W}(t) \cdot \frac{\partial \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}, t)}{\partial \boldsymbol{X}(t)} \tag{9}
$$

$$
\boldsymbol{W}(t_f) = \frac{\partial \boldsymbol{Y}(\boldsymbol{X}(t_f))}{\partial \boldsymbol{X}(t_f)} \tag{10}
$$

It is easy to show that $\delta \boldsymbol{X}(t_0)=0$ in Eq. (8), because the initial conditions are specified. Along with this observation, substituting Eq. (9) and Eq. (10) into Eq. (8) yields the simplified expression

$$
d\boldsymbol{Y}(\boldsymbol{X}(t_f)) = \int_{t_0}^{t_f} [\boldsymbol{B}_s(t) \cdot \delta \boldsymbol{U}(t)]\, dt + \boldsymbol{B}_e \cdot dt_f \tag{11}
$$

where

$$
\boldsymbol{B}_s(t) \triangleq \boldsymbol{W}(t) \cdot \frac{\partial \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}, t)}{\partial \boldsymbol{U}(t)} \tag{12}
$$

$$
\boldsymbol{B}_e \triangleq -\frac{\partial \boldsymbol{Y}(\boldsymbol{X}(t_f))}{\partial \boldsymbol{X}(t_f)} \cdot \dot{\boldsymbol{X}}(t_f) \tag{13}
$$

Where $\boldsymbol{B}_s(t)$ is called the sensitivity matrix as per Ref. [21]. And $\boldsymbol{B}_e$ can be interpreted as the sensitivity matrix that relates the error dt_f to $d\boldsymbol{Y}$. Equation (9)-(13) provide a closed-form relation between $d\boldsymbol{Y}(\boldsymbol{X}(t_f))$ and the error $\delta \boldsymbol{U}(t)$ as well as dt_f. This relationship is applied to derive the proposed GS-MPSP method in the following subsection.

2.2. Generalized quasi-spectral model predictive static programming

To reduce the number of optimization variable, the quasi-spectral representation for control vector proposed in Ref. [10] is used. That is, to express the control vector as a weighted summation of some basic spectral functions

$$
\boldsymbol{U}(t) = \sum_{i=1}^{N_p} \boldsymbol{C}_i P_i(t) \tag{14}
$$

where $\boldsymbol{C}_j = [c_{1j}, c_{2j}, \cdots, c_{mj}]^T$ denotes the coefficient vector of the jth spectral function. N_p is the number of basic functions in the expression, and $P_j(t)$ is the basic spectral function. The spectral functions can be selected as arbitrary forms such as Legendre series, Chebychev series, etc.

Once the coefficient vectors and terminal time are determined, the control history in the time domain $[t_0, t_f]$ is given. Therefore, the primary objective of this method is translated to find a suitable coefficient vector $[\boldsymbol{C}_1, \boldsymbol{C}_2, \cdots, \boldsymbol{C}_{N_p}]$ and terminal time t_f so that the desired terminal conditions are reached with minimum control effort. In this subsection, a computationally efficient way to find this solution is presented, which computes the required the coefficient vectors and terminal time by an iterative method. Noted that in this process, the iterative variable are the coefficient vector and terminal time rather than the control history as used in the traditional MPSP approach.

First, as per Eq. (14), it can be shown that the variation of each coefficient vector $d\boldsymbol{C}_j$ leads to a variation of the control history during the time $t \in [t_0, t_f]$, that is:

$$
\delta \boldsymbol{U}(t) = \sum_{j=1}^{N_p} d\boldsymbol{C}_j P_j(t) \tag{15}
$$

Substituting Eq. (15) into Eq. (11), it yields the error of the final output as

$$
\begin{aligned}
d\boldsymbol{Y}_N = & \int_{t_0}^{t_f} \left[\boldsymbol{B}_s(t) \cdot \sum_{j=1}^{N_p} d\boldsymbol{C}_j P_j(t) \right] dt + \boldsymbol{B}_e \cdot dt_f \\
= & \sum_{j=1}^{N_p} d\boldsymbol{C}_j \cdot \int_{t_0}^{t_f} \left[\boldsymbol{B}_s(t) \cdot P_j(t) \right] dt + \boldsymbol{B}_e \cdot dt_f \\
= & \sum_{j=1}^{N_p} d\boldsymbol{C}_j \cdot \boldsymbol{F}_j + \boldsymbol{B}_e \cdot dt_f
\end{aligned} \tag{16}
$$

in which

$$
\boldsymbol{F}_j = \int_{t_0}^{t_f} \boldsymbol{B}_s(t) \cdot P_j(t) dt \tag{17}
$$

In Eq. (16), $\boldsymbol{F}_j$ relates the error of the coefficient of the jth spectral function, $d\boldsymbol{C}_j$, to the error of the output $d\boldsymbol{Y}_N$. It is named spectral sensitivity matrix here to distinguish from the sensitivity matrix used in MPSP and QS-MPSP methods. Note that Eq. (16) gives a linear formula for the error of the final output and the error of each coefficient vector as well as terminal time, if the sensitivity matrix $\boldsymbol{F}_j$ and $\boldsymbol{B}_e$ are obtained. Based on this relation, the desired coefficients and terminal time can be solved by formulating a static programming problem:

First, denote $\boldsymbol{C}_j^l$ and t_f^{l+1} as the jth coefficient and terminal time at the current iteration (represented by superscript l), the

updated coefficient and terminal time in the next iteration (represented by superscript $l+1$) are written as:

$$\boldsymbol{C}_j^{l+1} = \boldsymbol{C}_j^l - d\boldsymbol{C}_j^l \tag{18}$$

$$t_f^{l+1} = t_f^l - dt_f \tag{19}$$

Substituting the expression of $d\boldsymbol{C}_j^l$ from Eq. (18) into Eq. (16) yields

$$\begin{aligned} d\boldsymbol{Y}_N &= \sum_{j=1}^{N_p} \boldsymbol{F}_j(\boldsymbol{C}_j^l - \boldsymbol{C}_j^{l+1}) + \boldsymbol{B}_e \cdot dt \\ &= \boldsymbol{c}_\lambda - \sum_{j=1}^{N_p} \boldsymbol{F}_j \boldsymbol{C}_j^{l+1} + \boldsymbol{B}_e \cdot dt \end{aligned} \tag{20}$$

where

$$\boldsymbol{c}_\lambda = \sum_{j=1}^{N_p} \boldsymbol{F}_j \boldsymbol{C}_j^l \tag{21}$$

It is clear that Eq. (20) gives a linear equations set about $\boldsymbol{C}_j^{l+1}$ and dt, which contains $N_p \times m$ unknowns (i.e., $\boldsymbol{C}_1^{l+1}, \boldsymbol{C}_2^{l+1}, \cdots, \boldsymbol{C}_{N_p}^{l+1}$) and p equations where $N_p \times m > p$. Hence, Eq. (20) represents an under-constrained system. To facilitate a solution, the performance index in Eq (3) is introduced, and it is translated to an equivalent form by substituting Eqs. (18) and (19) into Eq (3) as:

$$\begin{aligned} J &= \frac{1}{2} \int_{t_0}^{t_f^l - dt_f} \left[\boldsymbol{U}^{l+1}(t)^T \boldsymbol{R}(t) \boldsymbol{U}^{l+1}(t) \right] \cdot dt \\ &= \frac{1}{2} \int_{t_0}^{t_f^l} \left[\left(\sum_{i=1}^{N_p} \boldsymbol{C}_i^{l+1} P_i(t) \right)^T \boldsymbol{R}(t) \left(\sum_{i=1}^{N_p} \boldsymbol{C}_i^{l+1} P_i(t) \right) \right] \cdot dt \\ &\quad - R_f \cdot dt_f \end{aligned} \tag{22}$$

where

$$\begin{aligned} R_f &= \frac{1}{2} \cdot \left[\boldsymbol{U}^{l+1}(t)^T \boldsymbol{R}(t) \boldsymbol{U}^{l+1}(t) \right]_{t=t_f^l} \\ &\approx \frac{1}{2} \cdot \left[\boldsymbol{U}^l(t)^T \boldsymbol{R}(t) \boldsymbol{U}^l(t) \right]_{t=t_f^l} \end{aligned} \tag{23}$$

From Eq. (22), it can be seen that the performance index contains two parts: the first term is the control effort at fixed time interval $[t_0, t_f^l]$ and the second term is that considering the variant terminal time, dt_f. When computing the second term, $\boldsymbol{U}^{l+1}(t)$ is approximated by $\boldsymbol{U}^l(t)$ for solving convenience as given in Eq. (23). Considering that $\boldsymbol{U}^l(t_f)$ will gradually approach $\boldsymbol{U}^{l+1}(t_f)$, such approximation is effective. Then, a static optimization problem can be formulated, which requires minimize the cost function (22) subject to the constraints given in Eq. (20). The augmented cost function of this problem is given by

$$\bar{J} = J + \boldsymbol{\lambda}^T (d\boldsymbol{Y}_N - \boldsymbol{c}_\lambda + \sum_{j=1}^{N_p} \boldsymbol{F}_j \boldsymbol{C}_j^{l+1} - \boldsymbol{B}_e \cdot dt) \tag{24}$$

Where $\boldsymbol{\lambda} \in R^p$ is Lagrange multiplier. The first-order optimality conditions are:

$$\frac{\partial \bar{J}}{\partial d\boldsymbol{C}_j} = \frac{\partial J}{\partial d\boldsymbol{C}_j} - \boldsymbol{F}_j{}^T \cdot \boldsymbol{\lambda} = \boldsymbol{0} \tag{25}$$

$$\frac{\partial \bar{J}}{\partial dt_f} = R_f \cdot dt_f - \boldsymbol{B}_e{}^T \cdot \boldsymbol{\lambda} = \boldsymbol{0} \tag{26}$$

In Eq. (25), the first term is expressed as

$$\begin{aligned} \frac{\partial J}{\partial d\boldsymbol{C}_j} &= \int_{t_0}^{t_f^l} \left[\frac{1}{2} \boldsymbol{R}(t) \left(\sum_{i=1}^{N_p} \boldsymbol{C}_i^{l+1} P_i(t) \right) \cdot P_j(t) \right] \cdot dt \\ &= \sum_{i=1}^{N_p} \left(\int_{t_0}^{t_f^l} \left[P_i(t) \boldsymbol{R}(t) P_j(t) \right] \cdot dt \right) \cdot \boldsymbol{C}_j^{l+1} \\ &= \sum_{i=1}^{N_p} \boldsymbol{R}_{ij} \cdot \boldsymbol{C}_j^{l+1} \end{aligned} \tag{27}$$

Where

$$\boldsymbol{R}_{ij} = \int_{t_0}^{t_f^l} \left[P_i(t) \boldsymbol{R}(t) P_j(t) \right] \cdot dt \tag{28}$$

The corresponding matrix $\boldsymbol{R}_{ij}$ of the performance index plays an important role, that represents the differential of performance index J as the form about the coefficients $\boldsymbol{C}_j^{l+1}$. Then combining Eqs. (20), (25) and (26), the updated $\boldsymbol{C}_j^{l+1}$ and dt_f can be worked out.

First, the expression of dt_f can be obtained from Eq. (25) as

$$dt_f = \frac{1}{R_f} \cdot \boldsymbol{B}_e{}^T \cdot \boldsymbol{\lambda} \tag{29}$$

Then, substituting dt_f into Eq. (20) and combining Eq. (25), an equation set about $\boldsymbol{C}_j^{l+1}$ and $\boldsymbol{\lambda}$ can be formulated as

$$\begin{cases} \sum_{i=1}^{N_p} \boldsymbol{R}_{ij} \cdot \boldsymbol{C}_i^{l+1} - \boldsymbol{F}_j{}^T \cdot \boldsymbol{\lambda} = \boldsymbol{0}, \left(j = 1, 2, \cdots N_p \right) \\ \sum_{j=1}^{N_p} \boldsymbol{F}_j \cdot \boldsymbol{C}_j^{l+1} - \boldsymbol{e}_\lambda \cdot \boldsymbol{\lambda} = \boldsymbol{c}_\lambda - d\boldsymbol{Y}_N \end{cases} \tag{30}$$

Where

$$\boldsymbol{e}_\lambda = \frac{1}{R_f} \boldsymbol{B}_e \cdot \boldsymbol{B}_e{}^T \tag{31}$$

Equation set (30) contains $N_p \times m + p$ unknowns (i.e., $\boldsymbol{C}_1^{l+1}$, $\boldsymbol{C}_2^{l+1} \cdots \boldsymbol{C}_{N_p}^{l+1}, \boldsymbol{\lambda}$) and the same number of equations. Defining $\boldsymbol{X}_C = [(\boldsymbol{C}_1^{l+1})^T, (\boldsymbol{C}_2^{l+1})^T, \cdots (\boldsymbol{C}_{N_p}^{l+1})^T, \boldsymbol{\lambda}^T]^T$, this equation set can be rewritten as a compact form

$$\boldsymbol{G} \cdot \boldsymbol{X}_C = \boldsymbol{H} \tag{32}$$

Where

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{R}_{11} & \cdots & \boldsymbol{R}_{1N_p} & -\boldsymbol{F}_1{}^T \\ \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{R}_{N_p 1} & \cdots & \boldsymbol{R}_{N_p N_p} & -\boldsymbol{F}_{N_p}{}^T \\ \boldsymbol{F}_1 & \cdots & \boldsymbol{F}_{N_p} & -\boldsymbol{e}_\lambda \end{bmatrix} \text{ and, } \quad \boldsymbol{H} = \begin{bmatrix} \boldsymbol{0} \\ \vdots \\ \boldsymbol{0} \\ \boldsymbol{c}_\lambda - d\boldsymbol{Y}_N \end{bmatrix} \tag{33}$$

Assuming that $\boldsymbol{G}$ is nonsingular, the matrix $\boldsymbol{X}_C$ is solved by Eq. (32) as

$$\boldsymbol{X}_C = \boldsymbol{G}^{-1} \cdot \boldsymbol{H} \tag{34}$$

Consequently, the updated coefficients $\boldsymbol{C}_j^{l+1}$ and the Lagrange multiplier $\boldsymbol{\lambda}$ are obtained from the solution of $\boldsymbol{X}_C$. Substituting $\boldsymbol{\lambda}$ and $\boldsymbol{C}_j^{l+1}$ into Eq. (29) and Eq. (14), respectively, the updated terminal time and control history at time $t \in [t_0, t_f^{l+1}]$ can be obtained as:

$$t_f^{l+1} = t_f^l - dt_f = t_f^l - \frac{1}{R_f} \cdot \boldsymbol{B}_e{}^T \cdot \boldsymbol{\lambda} \tag{35}$$

and

$$\boldsymbol{U}^{l+1}(t) = \sum_{i=1}^{N_p} \boldsymbol{C}_i^{l+1} P_i(t) \tag{36}$$

If the terminal time is fixed, the matrix $\boldsymbol{e}_\lambda$ in Eq. (33) is zero. Then the corresponding updated control can also be obtained from Eq. (34) and Eq. (36).

In above derivation, the update of the coefficient vector is formulated by the spectral sensitivity matrix, and the spectral sensitivity matrix is represented in the continuous-time framework. As a result, it is not necessary to discretize the system dynamic and compute the sensitive matrix at every time step as compared to MPSP or QS-MPSP.

Remark 1. To obtain the sensitivity matrix $\boldsymbol{F}_j$ and the matrix $\boldsymbol{R}_{ij}$, one has to solve the matrix differential equation Eq. (9) and compute the integral expression given by Eq. (17) and Eq. (28). Obviously, this means of calculation cannot gain high computational efficiency due to the complicated matrix integration, if it is performed using common methods of numerical integration. However, the continuous-time form makes it possible to apply Gauss Quadrature Collocation, which can simplify the calculation process and improve computational efficiency significantly. In the next section, this new method will be detailed.

3. GS-MPSP using Gaussian quadrature collocation

In this section, an efficient way to compute the spectral sensitivity matrix $\boldsymbol{F}_j$ and the matrix $\boldsymbol{R}_{ij}$ is presented. Then combining with this approach, the implement procedure of the proposed GS-MPSP method is introduced.

3.1. The computation of $\boldsymbol{W}(t)$ using the collocation method

As discussed in previous sections, the matrix $\boldsymbol{W}(t)$ is constrained by the differential equation given in Eq. (9) and the final boundary condition given in Eq. (10). This is an initial-value problem and, in general, can be solved by numerical integration. Here, the collocation method is put forward to solve the matrix differential equation.

The scale time $\tau \in [-1, 1]$ is defined as another independent variable which is matched with the physical time t as follows

$$t \equiv t(\tau, t_0, t_f) = \frac{t_f - t_0}{2}\tau + \frac{t_f + t_0}{2} \tag{37}$$

where $t \in [t_0, t_f]$, t_0 and t_f are the initial and final time, respectively. Furthermore, the collocation points $\tau_i (i = 1, 2 \cdots, N)$ are defined in the interval $[-1, 1]$. The collocation points can be selected in any arbitrary form as long as they have two endpoints on the interval $[-1, 1]$, that is $\tau_1 = -1, \tau_N = 1$. For example, Legendre-Gauss-Lobatto (LGL), or Chebyshev-Gauss-Lobatto (CGL) series can be optional choice.

In the proposed method, the solution of $\boldsymbol{W}(t)$ is obtained by successively solving each row of elements. First, consider the kth row vector of matrix $\boldsymbol{W}(t)$ which is denoted by $\boldsymbol{W}_k(t)(k = 1, 2, \cdots, p)$. This function satisfies the differential equation Eq. (9) and can be written as follows

$$\dot{\boldsymbol{W}}_k(t) = -\boldsymbol{W}_k(t) \cdot \frac{\partial \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}, t)}{\partial \boldsymbol{X}(t)} \tag{38}$$

Putting scale time τ as independent variable, Eq. (38) can be rewritten as

$$\dot{\boldsymbol{W}}_k(\tau) = -\boldsymbol{W}_k(\tau) \cdot \boldsymbol{f}_x(\tau) \tag{39}$$

where

$$\boldsymbol{f}_x(\tau) \triangleq \frac{\partial \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}, t)}{\partial \boldsymbol{X}(t)} \cdot \frac{t_f - t_0}{2} \tag{40}$$

Theoretically, $\boldsymbol{W}_k(\tau)$ must satisfy Eq. (39) at all collocation points $\tau_i (i = 1, 2 \cdots, N)$. However, $\boldsymbol{W}_k$ is generally computed by integrating the matrix dynamics (39) backward from τ_N to τ_1. Thus, $\boldsymbol{W}_k(\tau)|_{\tau=\tau_1}$ as the last integral step is the integral result. That means the matrix $\boldsymbol{W}_k(\tau)|_{\tau=\tau_1}$ is not necessary to strictly satisfy the equation given by Eq. (39). Hence, only considering the $N-1$ collocation point $\tau_i (i = 2, \cdots, N)$ the Eq. (39) is written as

$$\dot{\boldsymbol{W}}_k(\tau_i) = -\boldsymbol{W}_k(\tau_i) \cdot \boldsymbol{f}_x(\tau_i),\ \tau_i (i = 2, \cdots, N) \tag{41}$$

Next, $\vec{\boldsymbol{A}}$ is denoted as the straightening or vector-valued function of matrix $\boldsymbol{A}$, which is an ordered stock of rows of $\boldsymbol{A}$. For example, given $\boldsymbol{A} = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix}$, the straightening of $\boldsymbol{A}$ is $\vec{\boldsymbol{A}} = [a_1, a_2, a_3, a_4]^T$.

Defining the two matrices as follows

$$\begin{cases} \boldsymbol{\Omega}_k = \left[\boldsymbol{W}_k(\tau_1)^T, \boldsymbol{W}_k(\tau_2)^T, \cdots, \boldsymbol{W}_k(\tau_N)^T\right]^T \\ \boldsymbol{\Omega}'_k = \left[\boldsymbol{W}_k(\tau_2)^T, \boldsymbol{W}_k(\tau_3)^T, \cdots, \boldsymbol{W}_k(\tau_N)^T\right]^T \end{cases} \tag{42}$$

The straightened format of the above matrices are

$$\begin{cases} \vec{\boldsymbol{\Omega}}_k = [\boldsymbol{W}_k(\tau_1), \boldsymbol{W}_k(\tau_2), \cdots, \boldsymbol{W}_k(\tau_N)]^T \\ \vec{\boldsymbol{\Omega}}'_k = [\boldsymbol{W}_k(\tau_2), \boldsymbol{W}_k(\tau_3), \cdots, \boldsymbol{W}_k(\tau_N)]^T \end{cases} \tag{43}$$

The differentiation of $\vec{\boldsymbol{\Omega}}'_k$ is given by

$$\begin{aligned} \dot{\vec{\boldsymbol{\Omega}}}'_k &= \left[\dot{\boldsymbol{W}}_k(\tau_2), \dot{\boldsymbol{W}}_k(\tau_3), \cdots, \dot{\boldsymbol{W}}_k(\tau_N)\right]^T \\ &= -\Big[\boldsymbol{f}_x(\tau_2)^T \boldsymbol{W}_k(\tau_2)^T, \boldsymbol{f}_x(\tau_3)^T \boldsymbol{W}_k(\tau_3)^T, \\ &\quad \cdots, \boldsymbol{f}_x(\tau_N)^T \boldsymbol{W}_k(\tau_N)^T\Big] \\ &= -\bar{\boldsymbol{f}} \cdot \vec{\boldsymbol{\Omega}}'_k = -\boldsymbol{f} \cdot \vec{\boldsymbol{\Omega}}_k \end{aligned} \tag{44}$$

where $\bar{\boldsymbol{f}} = \mathrm{diag}\left[\boldsymbol{f}_x(\tau_2), \boldsymbol{f}_x(\tau_3) \cdots, \boldsymbol{f}_x(\tau_N)\right]^T$ and $\boldsymbol{f} = \left[\boldsymbol{0}_{n(N-1)\times n}, \bar{\boldsymbol{f}}\right]$.

If each element w_{kj} of the vector $\boldsymbol{W}_k(\tau)$ is approximated using N Lagrange interpolating polynomials $L_i(\tau)(i = 1, 2, \cdots, N)$, the result yields,

$$w_{kj}(\tau) \approx \sum_{i=1}^{N} L_i(\tau) w_{kj}(\tau_i) = \boldsymbol{L}(\tau) \boldsymbol{w}_{kj} \tag{45}$$

where $\boldsymbol{L}(\tau) = [L_1(\tau), L_2(\tau), \cdots, L_N(\tau)]$ and $\boldsymbol{w}_{kj} = \left[w_{kj}(\tau_1), w_{kj}(\tau_2), \cdots, w_{kj}(\tau_N)\right]^T$. Mathematically, the differentiation of w_{kj} at each collocated point $\tau_i (i = 2, \cdots, N)$ can be expressed as

$$\dot{w}_{kj}(\tau_i) = \sum_{i=1}^{N} \dot{L}_i(\tau_i) w_{kj}(\tau_i) = \dot{\boldsymbol{L}}(\tau_i) \boldsymbol{w}_{kj} \tag{46}$$

Eq. (46) can be expanded to all collocated points $\tau_i (i = 2, \cdots, N)$ into a matrix form, which can be written as

$$\dot{\boldsymbol{w}}'_{kj} = \boldsymbol{D} \boldsymbol{w}_{kj} \tag{47}$$

where $\boldsymbol{w}'_{kj} = \left[w_{kj}(\tau_2), w_{kj}(\tau_3), \cdots, w_{kj}(\tau_N)\right]^T$ and $\boldsymbol{D} = \left[\dot{\boldsymbol{L}}(\tau_2)^T, \dot{\boldsymbol{L}}(\tau_3)^T, \cdots, \dot{\boldsymbol{L}}(\tau_N)^T\right]^T$.

The matrices $\boldsymbol{\Omega}_k$ and $\boldsymbol{\Omega}'_k$ in Eq. (43) can be rewritten as:

$$\begin{cases} \boldsymbol{\Omega}_k = [\boldsymbol{w}_{k1}, \boldsymbol{w}_{k2}, \cdots, \boldsymbol{w}_{kn}] \\ \boldsymbol{\Omega}'_k = \left[\boldsymbol{w}'_{k1}, \boldsymbol{w}'_{k2}, \cdots, \boldsymbol{w}'_{kn}\right] \end{cases} \tag{48}$$

Correspondingly, with Eq. (47), the differentiation of $\boldsymbol{\Omega}'_k$ can be derived as

$$\begin{aligned} \dot{\boldsymbol{\Omega}}'_k &= \left[\dot{\boldsymbol{w}}'_{k1}, \dot{\boldsymbol{w}}'_{k2}, \cdots, \dot{\boldsymbol{w}}'_{kn}\right] \\ &= \boldsymbol{D}\left[\boldsymbol{w}_{k1}, \boldsymbol{w}_{k2}, \cdots, \boldsymbol{w}_{kn}\right] = \boldsymbol{D} \cdot \boldsymbol{\Omega}_k \end{aligned} \tag{49}$$

Straightening the matrix on both sides of Eq. (49), it leads to

$$\dot{\vec{\boldsymbol{\Omega}}}'_k = (\boldsymbol{D} \cdot \vec{\boldsymbol{\Omega}}_k) = (\boldsymbol{D} \otimes \boldsymbol{I}_n) \cdot \vec{\boldsymbol{\Omega}}_k \tag{50}$$

Here, $\boldsymbol{I}_n$ is an $n \times n$ identity matrix and $\boldsymbol{D} \otimes \boldsymbol{I}_n$ denotes the Kronecker product of $\boldsymbol{D}$ and $\boldsymbol{I}_n$. Combining Eqs. (44) and (50), a group of linear equations is obtained as

$$\boldsymbol{A} \vec{\boldsymbol{\Omega}}_k = 0 \tag{51}$$

Where $\boldsymbol{A} = \boldsymbol{f}_x + (\boldsymbol{D} \otimes \boldsymbol{I}_n) \in \boldsymbol{R}^{(N-1)n \times Nn}$. Eq. (51) contains $(N-1)n$ linear equations and the same number of unknowns (since $\boldsymbol{W}_k(\tau_N)$ as boundary conditions is specified, the unknown variables of Eq. (51) are $\boldsymbol{W}_k(\tau_i)(i = 1, 2, \cdots, N-1)$). Therefore Eq. (51) can be solved theoretically.

Defining $\boldsymbol{X}_k = [\boldsymbol{W}_k(\tau_1), \boldsymbol{W}_k(\tau_2), \cdots, \boldsymbol{W}_k(\tau_{N-1})]^T$ as the vector of the unknown variables, it is straightforward to obtain $\vec{\boldsymbol{\Omega}}_k = \left[\boldsymbol{X}_k, \boldsymbol{W}_k(\tau_N)^T\right]^T$. Next A is rearranged as $\boldsymbol{A} = [\boldsymbol{A}_F, \boldsymbol{A}_N]$, where $\boldsymbol{A}_F$ and $\boldsymbol{A}_N$ denote the first $(N-1)n$ columns and the rest n columns of A, respectively. Substituting these relations into Eq. (51), the linear equations can be further expressed as

$$\boldsymbol{A} \vec{\boldsymbol{\Omega}}_k = [\boldsymbol{A}_F, \boldsymbol{A}_N] \cdot \begin{bmatrix} \boldsymbol{X}_k \\ \boldsymbol{W}_k(\tau_N)^T \end{bmatrix} = \boldsymbol{A}_F \boldsymbol{X}_k + \boldsymbol{A}_N \boldsymbol{W}_k(\tau_N)^T = 0 \tag{52}$$

Assuming that the matrix $\boldsymbol{A}_F$ is nonsingular, $\boldsymbol{X}_k$ can be obtained as follows

$$\boldsymbol{X}_k = -\boldsymbol{A}_F^{-1} \boldsymbol{A}_N \cdot \boldsymbol{W}_k(\tau_N)^T \tag{53}$$

The solution of $\boldsymbol{X}_k$ gives the value of kth row of matrix $\boldsymbol{W}(\tau)$ at the collocation points $(\tau_1, \tau_2, \ldots, \tau_{N-1})$. By repeating the above calculation procedure for each row $(k = 1, 2, \cdots, p)$, the matrix $\boldsymbol{W}(\tau)$ at all the collocation points $(\tau_1, \tau_2, \ldots, \tau_N)$ can be obtained.

Remark 2. Once the collocated points $\tau_i (i = 1, 2, \cdots, N)$ are given and $\boldsymbol{f}_x(\tau_i)$ is derived, the matrix $\boldsymbol{A}$ remains unchanged in the calculation loop for each row of the matrix $\boldsymbol{W}$. Therefore, only the different boundary conditions, given by $\boldsymbol{W}_k(\tau_N)^T$, are used. This feather reduces computational complexity when calculating the matrix $\boldsymbol{W}$.

3.2. The computation of the spectral sensitivity matrix using the Gaussian quadrature method

In the previous subsection, the value of matrix $\boldsymbol{W}$ at the collocation points $(\tau_1, \tau_2, \ldots, \tau_N)$ is obtained efficiently. With this value and Eq. (12), the sensitivity matrix $\boldsymbol{B}_s(t)$ at ith collocation point can be computed as follows

$$\boldsymbol{B}_s(\tau_i) = \boldsymbol{W}(\tau_i) \cdot \left.\frac{\partial \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}, t)}{\partial \boldsymbol{U}(t)}\right|_{t=\tau_i} \tag{54}$$

According to the principle of Gaussian quadrature [29], the integral expression in Eq. (17) can be written as

$$F_j = \int_{t_0}^{t_f} \boldsymbol{B}_s(t) \cdot P_j(t) dt = \frac{t_f - t_0}{2} \sum_{i=1}^{N} \boldsymbol{B}_s(\tau_i) \cdot P_j(\tau_i) \cdot \eta_i \tag{55}$$

Similarity, the performance index corresponding matrix $\boldsymbol{R}_{ij}$ given in Eq. (28) can be computed as

$$\begin{aligned} \boldsymbol{R}_{ij} &= \int_{t_0}^{t_f} \left[P_i(t) \boldsymbol{R}(t) P_j(t)\right] \cdot dt \\ &= \frac{t_f - t_0}{2} \sum_{k=1}^{N} P_i(\tau_k) \boldsymbol{R}(\tau_k) P_j(\tau_k) \cdot \eta_k \end{aligned} \tag{56}$$

where η_i is the weight coefficient of Gaussian quadrature corresponding to the collocation point. In this way, the spectral sensitivity matrix $\boldsymbol{F}_j (j = 1, 2, \cdots, N_p)$ and the matrix $\boldsymbol{R}_{ij} (i, j = 1, 2, \cdots, N_p)$ are obtained by a set of algebraic operation at very few collocation points.

Remark 3. Since $\boldsymbol{W}(\tau_i)(i = 1, 2, \cdots, N)$ is solved using a group of linear equations, and accordingly $\boldsymbol{B}_s(\tau_i)(i = 1, 2, \cdots, N)$ is obtained using Eq. (54), the spectral sensitivity matrix $\boldsymbol{F}_j (j = 1, 2, \cdots, N_p)$ and the matrix $\boldsymbol{R}_{ij}$ can be directly worked out by the Gaussian quadrature method in Eq. (55) and Eq. (56). This algorithm avoids heavy computational consumption produced by the numerical integration of a series of matrix differential equations. Hence, the computational efficiency is improved significantly.

3.3. GS-MPSP using Gaussian quadrature collocation

In Section 2, the calculation logic of GS-MPSP is summarized. In this method, the relationship between control increment and final output error considering the unspecified terminal time is formulated for the nonlinear system in a continuous framework. Subsequently, the control vector is described in a spectral form instead of uniform discretization. Correspondingly, the terminal output errors are related to the spectral coefficients and terminal time instead of the control profiles. In order to work out the desired spectral coefficients, a series of sensitivity matrix as well as related matrix differential equations need to be solved, which takes a large amount of computational time. In order to increase computational efficiency, an algorithm applying the Gaussian Quadrature Collocation method is proposed in Section 3 to solve these spectral sensitivity matrix efficiently. The implementation procedure of this approach is presented in Fig. 1.

First, an initial guess for the spectral coefficients and terminal time are given. Then the final output errors are evaluated. If the tolerance of the output errors is small enough, the desired control sequence is obtained. Otherwise, the corresponding sensitivity matrices are recalculated and the spectral coefficients as well as terminal time is updated. Subsequently, a new control sequence is

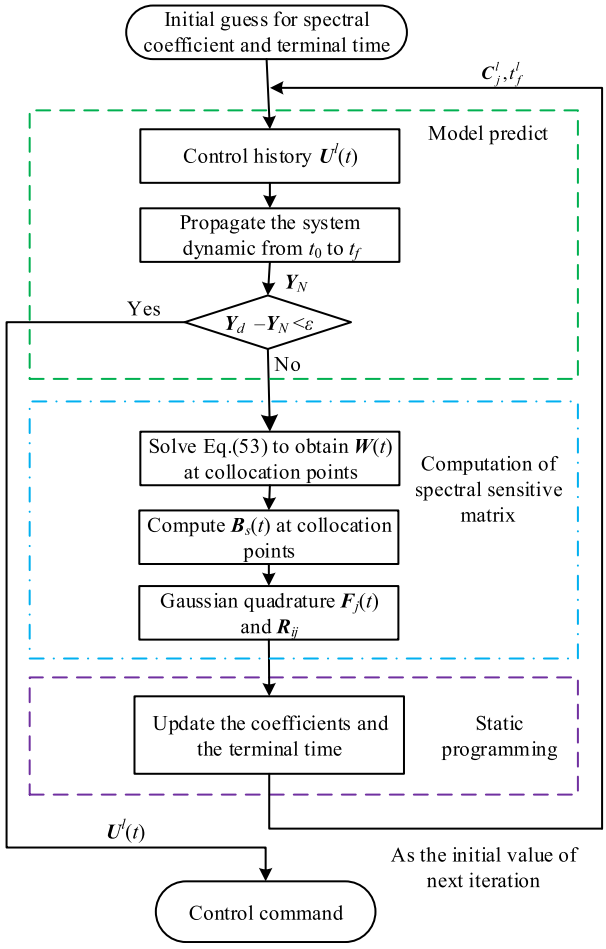


Fig. 1. The implement flowchart of GS-MPSP using Gaussian quadrature collocation.

generated, and the output errors are evaluated again. This iterative procedure is repeated until a specified criterion for the terminal output errors is met.

Since the spectral coefficients have no physical meaning, it's not straightforward to assign an initial guess with appropriate values. In practice, the least-squares algorithm [31] can be applied to obtain the initial guess of the spectral coefficients, by fitting a given guess of control sequence.

4. Simulation case

In this section, a three-dimensional mid-course interception guidance problem with a specified impact-angle constraint is carried out to demonstrate the performance of the proposed method. For comparison, the MPSP method [13], MPCP method [19] and QS-MPSP algorithm proposed in Ref. [28] are also implemented. The computational accuracy and efficiency of each method are evaluated under the same conditions and discussed.

4.1. Missile dynamics

In the simulation case, an aerodynamically controlled missile is engaging a high-speed target with impact angle constraints in the mid-course phase. It is assumed that the solid motor of the missile has cut off and the position of predict intercept point (PIP) is given. Midcourse guidance is required to guide the missile to reach the desired PIP point (x_d, y_d, z_d) with a desired flight path angle γ_d and heading angle ψ_d. The three-dimensional dynamics of the missile are given by the following equations:

$$\begin{aligned}
\dot{V}_m &= -D/m + g\sin\gamma_m \\
\dot{\gamma}_m &= a_y/V_m - g\cos\gamma_m/V_m \\
\dot{\psi}_m &= -a_z/(V_m\cos\gamma_m) \\
\dot{x}_m &= V_m\cos\gamma_m\cos\psi_m \\
\dot{y}_m &= V_m\sin\gamma_m \\
\dot{z}_m &= -V_m\cos\gamma_m\sin\psi_m
\end{aligned} \tag{57}$$

where V_m represents the missile velocity; γ_m and ψ_m denote the flight path and heading angles of the missile, respectively; (x_m, y_m, z_m) are the missile positions in the fixed inertial frame; a_y and a_z denote the vertical and horizontal control accelerations, which are produced by the lift force Y and the side force Z, respectively (i.e., $a_y = Y/m$, $a_z = Z/m$); and D is the drag force. Using the well-known drag polar [32], the drag coefficient is given as

$$C_D = C_{D0}(Ma) + K(Ma)\cdot(C_y^2 + C_z^2) \tag{58}$$

where C_{D0} denotes the zero-lift drag coefficient; K is the induced drag factor; C_y and C_z are the lift and side force coefficients, respectively. Then, the drag force can be written as

$$D = C_{D0}(Ma)\cdot q\cdot S + \frac{K(Ma)}{qs}\cdot(L^2 + Z^2) \tag{59}$$

Where q is the dynamic pressure, S is the reference area. Substituting $a_y = Y/m$ and $a_z = Z/m$ into Eq. (59) leads to

$$D = C_{D0}(Ma)\cdot q\cdot S + \frac{m^2}{qs}\cdot K(Ma)\cdot(a_y^2 + a_z^2) \tag{60}$$

The zero-lift drag coefficient C_{D0} and induced drag factor K can be obtained by interpolation according to the current Mach number. Thus, the drag force D can be determined by given control accelerations a_y and a_z. The purpose of the midcourse guidance is to find a proper control sequence that ensures the missile reaches the desired PIP point and achieves the desired angles, with minimal control effort. Therefore, the output vector is given by $\boldsymbol{Y} = [x_f, y_f, z_f, \gamma_f, \psi_f]^T$, and the control vector is $\boldsymbol{U} = [a_y, a_z]^T$. The autopilot delay is also considered, it is assumed to be first-order lags with a time constant of 0.3 s.

4.2. Problem setup

The proposed GS-MPSP, QS-MPSP, MPSP and MPCP methodology are all applied to solve the mid-course problem. The parameters for the various methods are ensured to be as similar as possible. For better approximation, both the GS-MPSP and QS-MPSP adopt Legendre polynomials as the basic spectral functions. Without loss of generality, the first four order Legendre polynomials are used here.

In addition, the Legendre-Gauss-Latto (LGL) points [29] are selected as the collocation points to compute the spectral sensitivity matrix in the GS-MPSP method, and the number of these points is set to 8. The proposed GS-MPSP, MPSP and MPCP method all adopt the performance index in Eq. (3), where the control weight matrix is chosen as $\boldsymbol{R}(t) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. The QS-MPSP method adopts the performance index as follow, which approximately represents the minimization of the control effort, as per [28]. The weight matrix for coefficient $\boldsymbol{R}_j$ is taken as same as $\boldsymbol{R}(t)$.

$$J = \sum_{j=1}^{N_p} (\boldsymbol{C}_j)^T \boldsymbol{R}_j (\boldsymbol{C}_j) \tag{61}$$

Table 1
Simulation parameters.

Parameters	Values
Missile's initial position (x_m, y_m, z_m)	(0,15,0) km
Missile's initial velocity V_m	2500 m/s
Missile's initial flight path angle γ_m	25 deg
Missile's initial azimuth angle ψ_m	-20 deg
The PIP's position (x_d, y_d, z_d)	(40,25,4) km
Desired flight path angle γ_d	0 deg
Desired azimuth angel d ψ_d	12 deg
Missile's mass m	225 Kg
Reference area S	0.1257 m^2
Zero-lift drag coefficient C_{D0}	0.35
Induced drag factor K	0.09
The time constant for Autopilot delay	0.3 s

Table 2
Simulation results for various methods.

Method	Miss (m)	γ_f (deg)	ψ_f (deg)	t_f (s)	Cost
PN	9.4560	13.4582	-0.7696	27.30	41960.29
GS-MPSP[1]	0.0855	-0.0011	11.9948	27.43	31475.56
GS-MPSP[2]	0.3250	0.0038	11.9744	27.30	32581.86
MPSP	0.3084	0.0017	11.9800	27.30	32581.09
QS-MPSP	0.3429	0.0005	11.9837	27.30	32605.39
MPCP	0.3084	0.0017	11.9800	27.30	32581.09

[1] Free terminal time.
[2] Fixed terminal time.

A guess of control history is necessary for the implementation of the MPSP, MPCP and QS-MPSP approaches. Similarly, the proposed GS-MPSP methods require an initial guess of the coefficients to start the iterative process, introduced in Section 3.3. The initial control guess $\boldsymbol{U}_k^0 (k = 1, 2, \cdots, N)$ for each method is generated by proportional navigation (PN) guidance as given in Ref. [33].

4.3. Performance test

The simulation parameters are presented in Table 1. The convergence tolerances for position and impact angle are set to 1 m and 0.1 degrees, respectively. All programs were implemented on a personal computer with a 3.4 GHz processor and MATLAB v.2018b. The MPCP method is performed using CVX with MOSEK solver [19]. In simulation, the proposed GS-MPSP method considers both the case of free and fixed terminal time, where the fixed terminal time is specified by the initial guess like other methods. The simulation results for various MPSP methods are provided in Fig. 2 and Table 2. Additionally, the PN guidance's results are also presented in Table 2 as the initial guess solution of the other methods. It can be noted the control cost in Table 2 represents the total control energy evaluated by the cost function in Eq. (3).

As shown in Fig. 2 and Table 2, the PN guidance successfully hits the target but fails to satisfy the terminal angle constraints. Using the PN solution as the initial guess, the proposed GS-MPSP, as well as other methods, hit the target accurately while achieving the desired impact angles. More detailed, it can be seen that the GS-MPSP with fixed terminal time gains almost the same terminal accuracy and control cost with MPSP method. And the acceleration command histories produced by the two methods are almost consistent, as can be observed in Fig. 2(e) and Fig. 2(f). This indicates that the proposed method can accurately provide the same control effort as the MPSP method, which demonstrates its effectiveness. As a comparison, the QS-MPSP spends a little more control cost than the MPSP and its control profiles also show a slight difference with that of MPSP method. It is caused by the difference between the performance index used by the two methods, as given in Eq. (3) and Eq. (61), respectively. Note that in this demonstration case, the difference caused by this is very little because the control weight matrix $R(t)$ used in MPSP is identity matrix and it can be approximated well by cost function (61) used in QS-MPSP. If a more complex $R(t)$ is used to achieve some specified purpose, the QS-MPSP method is hard to produce the same control effort as MPSP. Furthermore, in the case of the free terminal time, the proposed GS-MPSP gains the highest terminal accuracy and the least control cost among various methods, as presented in Table 2. At the same time, the control profile is smoother, as depicted in Fig. 2(e) and Fig. 2(f). It is important to note that the GS-MPSP in the case of free terminal time reached the desired convergence accuracy by 4 iterations, while 5 iterations for the case of fixed terminal time as well as other methods. The results indicate that the adjustment for terminal time not only obtains the better solution with less control cost and smoother control, but also improves the convergence accurate and rate of the algorithm. At last, the Table 2 shows the convex programming based MPCP method produces the similar terminal accuracy and control cost in solving the terminal constraint problem, as compared to other MPSP methods.

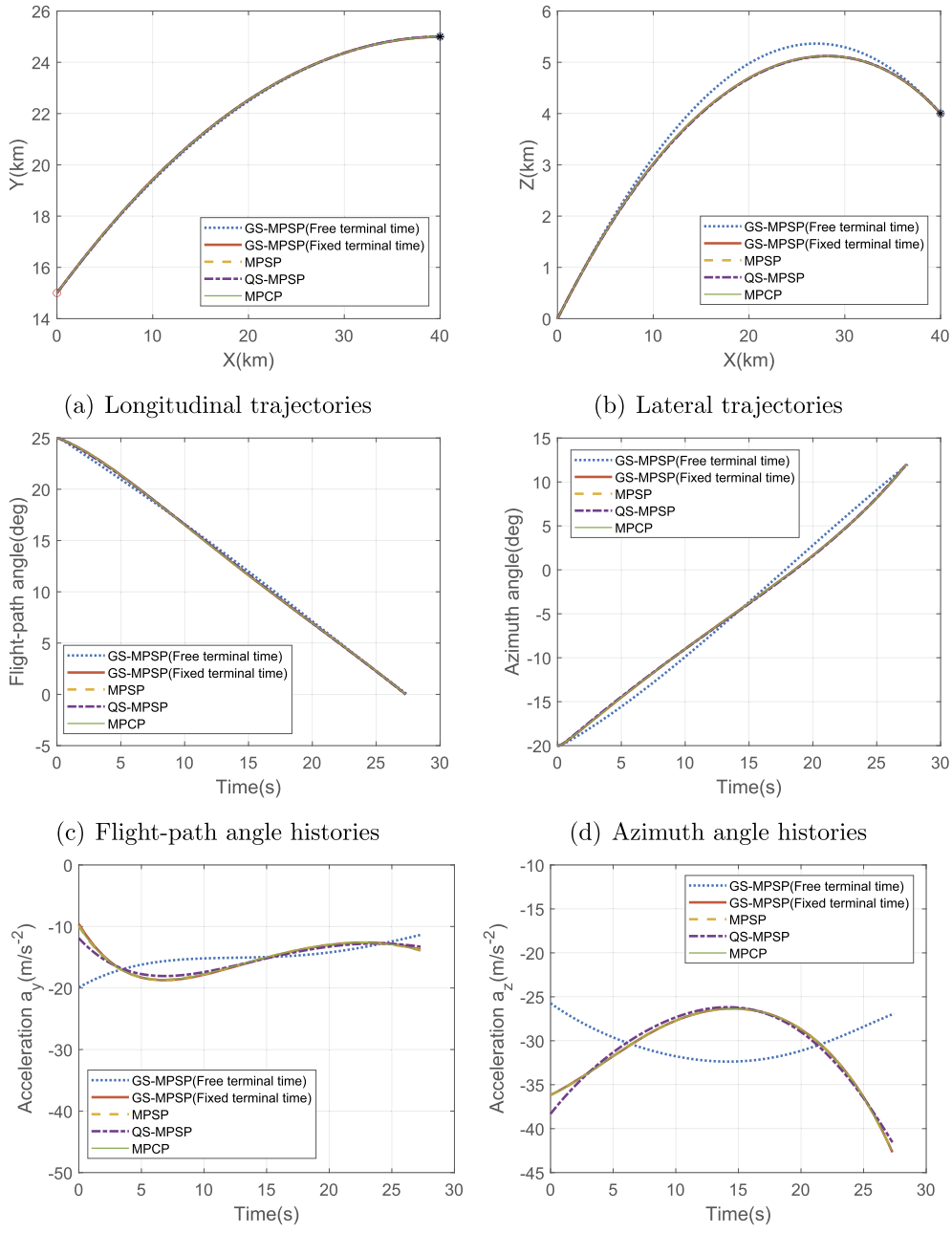
Table 3
The CPU time (s) consumed by various methods for one iteration.

Method	Model predict	Sensitivity matrix computation	Static/convex programming	Total
MPCP	0.0025147	0.0220437	0.1681947	0.1927531
MPSP	0.0025823	0.0218856	0.0108006	0.0352685
QS-MPSP	0.0025336	0.0247607	0.0013389	0.0286332
GS-MPSP	0.0025698	0.0009302	0.0013938	0.0048938

As mentioned in the prior sections, the computation of the spectral sensitivity matrix is the most important part of the GS-MPSP method. To address this, the proposed method uses Gaussian quadrature collocation to compute the spectral sensitivity matrix, which greatly improves computational efficiency. The values of the spectral sensitivity matrix in the first iteration when implementing GS-MPSP are given in Fig. 3 to demonstrate the computational accuracy of this strategy. For comparison, the fourth-order-Runge-Kutta method (RK4) is used to solve the differential equation (9) and the integral expression (17) with the same condition. This solution is considered to be the exact spectral sensitivity matrix and is presented in Fig. 3.

In Fig. 3, all of the elements of the spectral sensitivity matrix are rewritten in a sequence, and the abscissa axis stands for the number of elements (each spectral sensitivity matrix $\boldsymbol{F}_j (j = 1, 2, 3, 4)$ contains 10 elements since it is a 5×2 matrix). It can be observed that all the elements of the spectral sensitivity matrix produced by Gaussian Quadrature Collocation are consistent with that produced using RK4. The results indicate that the proposed strategy for computation of the spectral sensitivity matrix can obtain high numerical precision. Note that the accuracy of the sensitivity matrix significantly impacts the convergence property of the MPSP algorithm. Therefore, the proposed GS-MPSP is able to achieve high convergence accuracy.

To investigate the computational efficiency of the proposed method, Fig. 4 and Table 3 present the CPU time consuming for one iteration when implementing various methods. In general, the calculation procedure of an MPSP or MPCP approach consists of three parts: model prediction, (spectral) sensitivity matrix computation and static programming or convex programming. The time elapses for each part and the total of each MPSP method are listed in Table 3. The result for GS-MPSP only gives the value in the case of free terminal time since it has no significant difference with that of fixed terminal time. The results clearly demonstrate the superiority of the proposed method and the mechanisms that drive it. As illustrated in Table 3, the proposed GS-MPSP method has the least CPU time, about one-seventh of the MPSP method or one-sixth of the QS-MPSP method. This disparity is primarily attributed to the high efficiency of the calculation of the spectral sensitivity matrix with the application of Gaussian quadrature col-



(a) Longitudinal trajectories

(b) Lateral trajectories

(c) Flight-path angle histories

(d) Azimuth angle histories

(e) Vertical acceleration commands of the missile

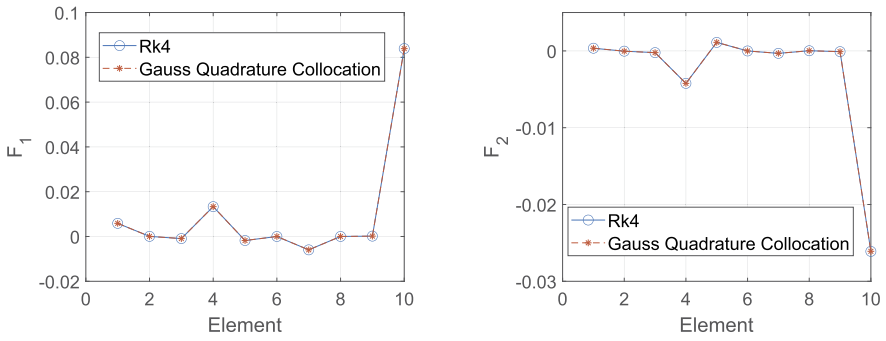
(f) Horizontal acceleration commands of the missile

Fig. 2. The simulation results for various methods. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

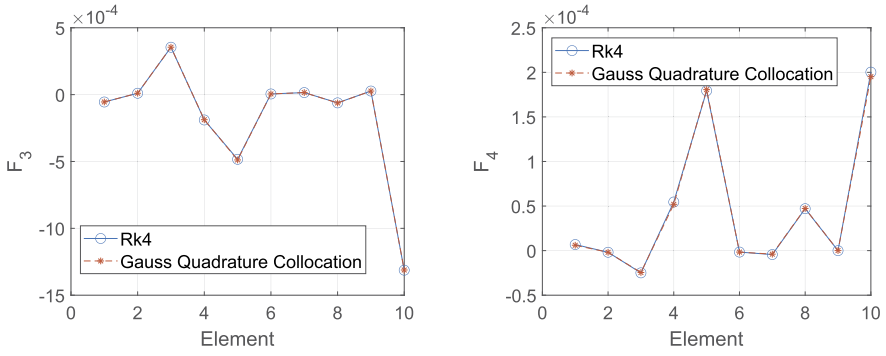
location. As can be seen, the time consumed in this part of the calculation is reduced by more than 20 times as compared to the MPSP and QS-MPSP approaches. Since the computation of sensitivity matrix is the most time-consuming step of the basic MPSP approach, a significant decrease in total CPU time is observed. In addition, the CPU time consumed in the static programming part is also lower than MPSP. This is because the number of the optimization variables is reduced. Thanks to these points, the proposed method achieves markedly enhanced computational efficiency. By comparison, the QS-MPSP just reduces the CPU time consumed by the static programming and therefore, the available improvement in computation efficiency is limited. This observation is consistent with the previous analysis. Besides, the Table 3 shows that the CPU time consumed by MPCP method is significantly higher than the MPSP methods. It should be noted that the MPCP method is proposed to specially address the problem with path constraint. Obviously, it has no advantage as compared to the MPSP methods in solving the problem with terminal constraints, no matter in accuracy and computational efficiency as presented in Table 2 and Table 3.

4.4. The cases for different terminal conditions

The GS-MPSP is applied to a scenario that considers varying terminal conditions. In this scenario, the initial conditions and desired position are fixed and taken from the values listed in Table 1. At these final positions, the desired terminal angles are varied over a specified range. The desired terminal flight path angle is con-



(a) The values of matrix F_1 (b) The values of matrix F_2



(c) The values of matrix F_3 (d) The values of matrix F_4

Fig. 3. Spectral sensitivity matrix obtained by Gaussian Quadrature Collocation and RK4 in the first iteration.

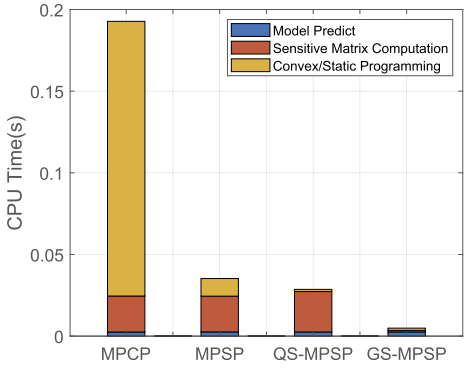


Fig. 4. The CPU time consumed by various methods.

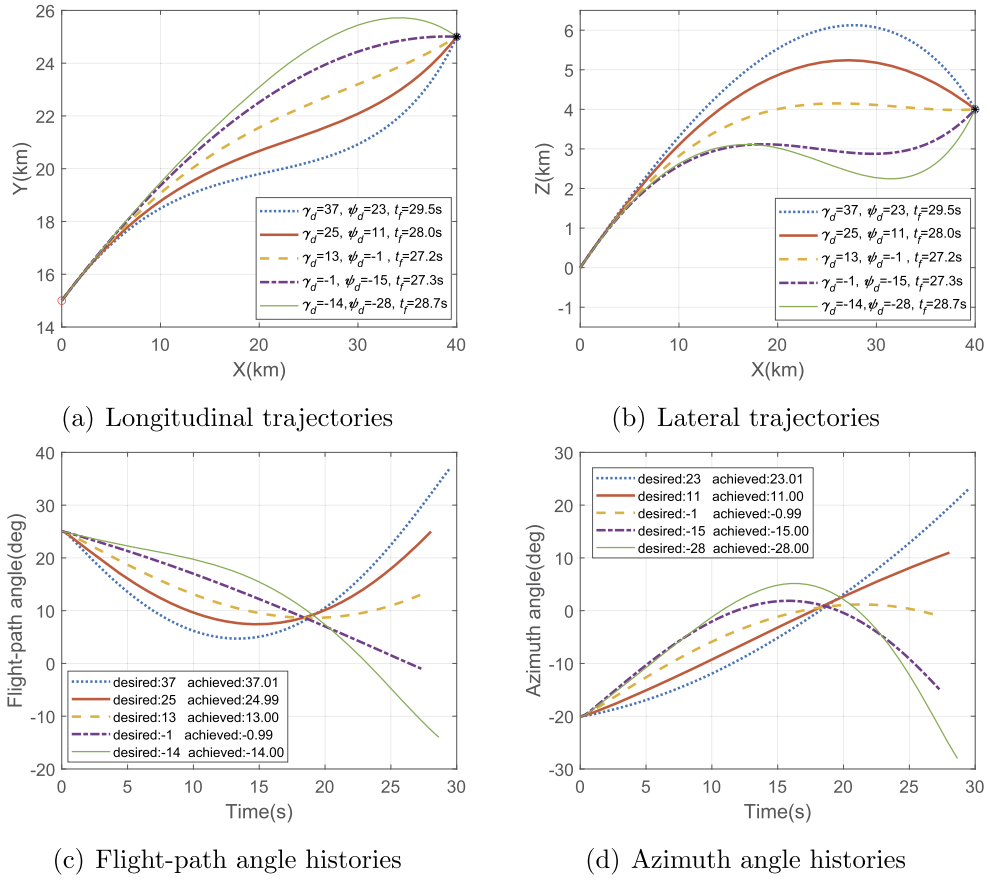
sidered to vary from -14 to 37 degrees and the desired azimuth angle is considered to vary from -28 to 23 degrees. In this case, the terminal time is considered to be free. The results generated by the GS-MPSP methods are depicted in Fig. 5. Figs. 5(a) and 5(b) show the engagement trajectories in the vertical and horizontal, and Figs. 5(c) and 5(d) give the corresponding histories of flight path angle and azimuth angle. It can be clearly observed that the desired terminal angles are achieved very accurately, with a maximum final impact-angle error below 0.02 deg. And the terminal time is adjusted from 27.2 s to 29.5 s in different terminal angles. It is interesting to note that, the initial guess solutions in the different terminal angle cases are same, which are generated by the PN guidance. Although the final flight-path and azimuth angles produced by PN are greatly vary from the desired values, the terminal constraints of the angles are satisfied well for all of test cases. The results demonstrate the robust convergence of the proposed method for the different constraints and the ability to accommodate the proper terminal time.

5. Conclusions

In this paper, a new generalized quasi-spectral model predictive static programming method is developed to improve its computational efficiency. To this end, a spectral representation of the control vector is used to reduce the dimensions of the static programming problem. In addition, a new spectral sensitivity matrix is derived, which represents the update of coefficients in a continuous-time framework and is then efficiently solved using Gauss Quadrature Collocation method. This strategy significantly reduces the runtime consumed in the sensitivity matrix computation, as compared to the recursive manner used to prevent MPSP. Besides, the terminal time is used as an optional optimize variable and updated as well as the spectral coefficients. As a result, the proposed method achieves markedly enhanced computational efficiency and is extended to solve the problem with free terminal time. At the same time, the additional advantages such as no need to discrete and more smooth control are also offered. A simulated interception scenario in the midcourse phase was provided. The simulation results indicate that: 1) the proposed GS-MPSP is effective and produces the same resulting control effort as present MPSP methods; 2) the proposed GS-MPSP is able to solve the problem with free terminal time, and in this case a better solution with less control cost, higher accuracy and smoother control can be obtained, as compared to the case of fixed terminal time; 3) the proposed GS-MPSP significantly improves the computational efficiency compared with the MPSP and QS-MPSP methods.

Declaration of competing interest

No conflict of interest exists in the submission of this manuscript, and the manuscript is approved by all authors for publication. I would like to declare on behalf of my co-authors that the work described was original research that has not been pub-



(a) Longitudinal trajectories (b) Lateral trajectories

(c) Flight-path angle histories (d) Azimuth angle histories

Fig. 5. The results for various desired flight path angle and azimuth angle.

lished previously, and not under consideration for publication elsewhere, in whole or in part. All the authors listed have approved the manuscript that is enclosed.

References

[1] A.L. Andrew, Two-Point Boundary Value Problems: Shooting Methods, Elsevier, New York, 1972.

[2] E. Kirk, Numerical Determination of Optimal Trajectories, Optimal Control Theory: An Introduction, Prentice–Hall, Upper Saddle River, NJ, 1975.

[3] D. Garg, M. Patterson, W.W. Hager, A.V. Rao, D.A. Benson, G.T. Huntington, A unified framework for the numerical solution of optimal control problems using pseudospectral methods, Automatica 46 (11) (2010) 1843–1851, https://doi.org/10.1016/j.automatica.2010.06.048.

[4] J. Wang, H. Liang, Z. Qi, D. Ye, Mapped Chebyshev pseudospectral methods for optimal trajectory planning of differentially flat hypersonic vehicle systems, Aerosp. Sci. Technol. 89 (2019) 420–430, https://doi.org/10.1016/j.ast.2019.04.017.

[5] P. Lu, Introducing computational guidance and control, J. Guid. Control Dyn. 40 (2) (2017) 193, https://doi.org/10.2514/1.G002745.

[6] P. Sarmah, C. Chawla, R. Padhi, A nonlinear approach for re-entry guidance of reusable launch vehicles using model predictive static programming, in: 2008 16th Mediterranean Conference on Control and Automation, Piscataway, NJ, 2008, pp. 41–46.

[7] P. Dwivedi, A. Bhattacharyya, R. Padhi, Computationally efficient sub-optimal mid course guidance using model predictive static programming, in: Proceedings of the 17th World Congress, Oxford, U.K., 2008, pp. 3550–3555.

[8] C.E. García, D.M. Prett, M. Morari, Model predictive control: theory and practice—a survey, Automatica 25 (3) (1989) 335–348, https://doi.org/10.1016/0005-1098(89)90002-2.

[9] J.A. Rossiter, Model Based Predictive Control: A Practical Approach, CRC Press, New York, 2003, pp. 1–84.

[10] M. Cannon, Efficient nonlinear model predictive control algorithms, Annu. Rev. Control 28 (2) (2004) 229–237, https://doi.org/10.1016/j.arcontrol.2004.05.001.

[11] P.J. Werbos, Approximate Dynamic Programming for Real Time Control and Neural Modelling, Handbook of Intelligent Control, Multiscience Press, New York, 1992.

[12] P.N. Dwivedi, A. Bhattacharya, R. Padhi, Suboptimal midcourse guidance of interceptors for high-speed targets with alignment angle constraint, J. Guid. Control Dyn. 34 (3) (2011) 860–877, https://doi.org/10.2514/1.50821.

[13] H.B. Oza, R. Padhi, Impact-angle-constrained suboptimal model predictive static programming guidance of air-to-ground missiles, J. Guid. Control Dyn. 35 (1) (2012) 153–164, https://doi.org/10.2514/1.53647.

[14] R. Padhi, C. Chawla, P.G. Das, Partial integrated guidance and control of interceptors for high-speed ballistic targets, J. Guid. Control Dyn. 37 (1) (2013) 149–163, https://doi.org/10.2514/1.61416.

[15] O. Halbe, R.G. Raja, R. Padhi, Robust reentry guidance of a reusable launch vehicle using model predictive static programming, J. Guid. Control Dyn. 37 (1) (2013) 134–148, https://doi.org/10.2514/1.61615.

[16] B. Zhang, S. Tang, B. Pan, Multi-constrained suboptimal powered descent guidance for lunar pinpoint soft landing, Aerosp. Sci. Technol. 48 (2016) 203–213, https://doi.org/10.1016/j.ast.2015.11.018.

[17] A. Maity, R. Padhi, G.M. Rao, G.M. Rao, M. Manickavasagam, A robust and high precision optimal explicit guidance scheme for solid motor propelled launch vehicles with thrust and drag uncertainty, Int. J. Syst. Sci. 47 (2016) 3078–3097, https://doi.org/10.1080/00207721.2015.1088100.

[18] F. Tavakoli, A.B. Novinzadeh, Designing a closed-loop guidance system to increase the accuracy of satellite-carrier boosters' landing point, Aerosp. Sci. Technol. 76 (2018) 242–249, https://doi.org/10.1016/j.ast.2018.01.031.

[19] Y. Wang, H. Hong, S. Tang, Geometric control with model predictive static programming on so(3), Acta Astronaut. 159 (2019) 471–479, https://doi.org/10.1016/j.actaastro.2019.01.023.

[20] S.H. Ramesh, R. Padhi, Three-dimensional nonlinear gravity assisted aiming point guidance, Aerosp. Sci. Technol. 85 (2019) 505–513, https://doi.org/10.1016/j.ast.2018.12.026.

[21] A. Maity, H.B. Oza, R. Padhi, Generalized model predictive static programming and angle-constrained guidance of air-to-ground missiles, J. Guid. Control Dyn. 37 (6) (2014) 1897–1913, https://doi.org/10.2514/1.G000038.

[22] A.K. Tripathi, R. Padhi, Autonomous landing for uavs using t-mpsp guidance and dynamic inversion autopilot, IFAC-PapersOnLine 49 (1) (2016) 18–23, https://doi.org/10.1016/j.ifacol.2016.03.022.

[23] P. Kumar, R. Padhi, Extension of model predictive static programming for reference command tracking, IFAC Proc. Vol. 47 (1) (2014) 855–861, https://doi.org/10.3182/20140313-3-IN-3024.00174.

[24] B. Pan, Y. Ma, R. Yan, Newton-type methods in computational guidance, J. Guid. Control Dyn. 42 (2) (2019) 377–383, https://doi.org/10.2514/1.G003931.

[25] S. Mondal, R. Padhi, State and input constrained missile guidance using spectral model predictive static programming, in: Guidance, Navigation, and Control and Co-Located Conferences, AIAA Paper 2018-1584, 2018, pp. 2487–2500.
[26] H. Hong, A. Maity, F. Holzapfel, S. Tang, Model predictive convex programming for constrained vehicle guidance, IEEE Trans. Aerosp. Electron. Syst. 55 (5) (2019) 2487–2500, https://doi.org/10.1109/TAES.2018.2890375.
[27] S. Mathavaraj, R. Padhi, Unscented mpsp for optimal control of a class of uncertain nonlinear dynamic systems, J. Dyn. Syst. Meas. Control 141 (6) (2019) 065001, https://doi.org/10.1115/1.4042549.
[28] S. Mondal, R. Padhi, Angle-constrained terminal guidance using quasi-spectral model predictive static programming, J. Guid. Control Dyn. 41 (3) (2018) 783–791, https://doi.org/10.2514/1.G002893.
[29] L.-L. Wang, Jie Shen, Tao Tang, Spectral methods algorithms, in: Analysis and Applications, in: Springer Series in Computational Mathematics, Springer, Berlin, Heidelberg, 2011, pp. 93–98 (Chapter 3).
[30] A.E. Bryson, Y. Ho, Applied Optimal Control, Hemisphere Publishing Corporation, New York, 1975.
[31] F.B. Hildebrand, Methods of Applied Mathematics, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1965.
[32] X. Liu, Z. Shen, P. Lu, Exact convex relaxation for optimal flight of aerodynamically controlled missiles, IEEE Trans. Aerosp. Electron. Syst. 52 (4) (2016) 1881–1892, https://doi.org/10.1109/TAES.2016.150741.
[33] P. Zarchan, Tactical and Strategic Missile Guidance, American Institute of Aeronautics and Astronautics, Washington D.C., 2012, p. 14 (Chapter 2).