Chapter 3
Ascent Predictive Guidance for Thrust Drop Fault of Launch Vehicles Using Improved GS-MPSP

Xiaodong Yan and Cong Zhou

3.1 Introduction

Increasing complex space missions require launch vehicles to be with greater load-carrying capacity, better orbit injection accuracy and higher reliability. Such demands also cause the increased complexity of the vehicle, leading to a higher probability of fault, especially for the propulsion system. To remedy this issue, an advanced and robust ascent guidance capable of fault-tolerant is critical for the success of mission. Iterative guidance method [1] (IGM) and powered explicit guidance [2] (PEG) are two commonly used methods for the ascent phase of launch vehicles. These two guidance methods work well in the nominal condition and can adapt to many off-nominal conditions [3]. However, they lack of strong adaptive capacity, which cannot guarantee the reliability when the dynamic model or parameters change significantly. Alternatively, numerical approaches based on the optimal control theory may be the better choice. The existing algorithms can be divided into direct methods and indirect methods. Using the indirect methods, the guidance problem is transformed into Hamilton two-point boundary value problems [4] (TPBVP), but the solving process of this Hamilton two-point boundary value problem is complicated and highly sensitive to the initial guess. Using the direct method, the guidance problem is transformed into a nonlinear programming problem [5] (NLP). However, solving such problem is extremely computational intensive, which is difficult to meet the real-time requirement for online application.

In recent years, computational guidance has been proposed, which allows for onboard to generate guidance commands by the rapid trajectory optimization or

X. Yan (✉) · C. Zhou
School of Astronautics, Northwestern Polytechnical University,
710072 Xi'an, People's Republic of China
e-mail: yan804@nwpu.edu.cn

C. Zhou
e-mail: zhoucongnwpu@163.com

Z. Song et al. (eds.), *Autonomous Trajectory Planning and Guidance Control for Launch Vehicles*, Springer Series in Astrophysics and Cosmology,
https://doi.org/10.1007/978-981-99-0613-0_3

other numerical computation [6]. Convex optimization method is the most popular one in this field [7]. The advantage of convex optimization is that the convex problem can be reliably solved by the interior point method to gain global optimality in polynomial time, regardless of the initial guess. Owing to the high efficiency, the convex optimization methods have been applied into various guidance designs, such as entry guidance [8], landing guidance [9], as well as ascent guidance [10]. A Newton-Kantorovich pseudospectral convex programming method was presented in [10] to solve the ascent trajectory planning problem. The combination of Newton-Kantorovich and pseudospectral discrete furtherly improves the computational efficiency. Similarly, Li et al. [11] presented a pseudospectral based convex optimization approach to solve the ascent guidance problem in the presence of thrust drop fault. Song et al. [12] investigated online joint optimization of the target orbit and flight trajectory for launch vehicles under a propulsion system fault. Most recently, the optimal abort orbit problem is studied in [13], which employed the SOCP approach to solve the circular abort orbit with the maximum of the radius. Through these literature, the computational guidance and online programming methods for ascent problems have been studied extensively. However, most of them are still limited in terms of the computational efficiency.

Another promising approach for online application is model predictive static programming (MPSP) [14]. Owing to featuring an explicit closed-form solution and avoiding numerical complexities of optimal control theory, this method exhibits a higher computational efficiency over convex optimization methods. In our previous work, a new developed generalized quasi-spectral MPSP (GS-MPSP) has been proposed [15], which furtherly improves the computational speed by using spectral discretization and collocation method. This new algorithm also offers the advantage of smooth control and better usability, and hence holds great promise for online application.

In this chapter, based on the GS-MPSP philosophy, an ascent guidance for the thrust drop fault of the launch vehicle is presented. For the ascent guidance problem, the thrust drop fault may lead to the flight time substantially changing compared to the nominal trajectory, and a proper final time or orbit injection point is not easy to give out. In this case, the ability to search the final time in a large range is required for the guidance algorithm. The original GS-MPSP [15] is able to address the free terminal time problem by formulating a sensitive relation between the final time and final outputs, and taking the final time as the additional variable to be determined. However, this operation is quite rough and feasible just for slightly adjusting the terminal time when a proper initial guess is provided. Therefore, an improved GS-MPSP (IGS-MPSP) method is first proposed, which furtherly enhances the convergence robustness for the free final time in the presence of the poor initial guess by introducing a scale factor of time interval. Then, the ascent guidance is systematically formulated using this improved algorithm. Consequently, a numerical simulation for the case of injecting into a circular orbit is conducted to verify the effectiveness of the proposed method. The comparison with the SOCP based method is also carried out.

3.2 Generic Theory of the IGS-MPSP Method

The GS-MPSP method is proposed for efficiently solving a class of nonlinear terminal constraint problem. The considered nonlinear system dynamics is as follows:

$$\dot{\boldsymbol{X}}(t) = \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}, t), t \in [t_0, t_f] \tag{3.1}$$

$$\boldsymbol{Y}(t) = \boldsymbol{h}(\boldsymbol{X}, t) \tag{3.2}$$

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 purpose of this approach is to find suitable control history $\boldsymbol{U}(t)$ to ensure that the final system output $\boldsymbol{Y}_f(t_f)$ approaches the desired value $\boldsymbol{Y}_d$ with minimum control effort.

3.2.1 The Sensitivity Relation for Free-Terminal Time Continuous System

In this subsection, a sensitivity relation for the continuous system with the free -terminal time is derived. It is based on the sensitivity relation developed in Ref. [15, 16]. The brief introduction is presented in the following.

In the proposed method, it is considered that the terminal time is adjusted by uniformly scaling the length of time interval $[t_0, t_f]$. Thus, the updated terminal time can be written as:

$$t_f^{l+1} = t_f^l + \Delta\kappa \cdot (t_f^l - t_0) \tag{3.3}$$

where $\Delta\kappa \in R$ is the scale factor, and the superscript l and $l+1$ denotes the current step and the update step. The differential of Eq. (3.3) is given by

$$dt^{l+1} = dt^l + \Delta\kappa \cdot dt^l, t \in [t_0, t_f^l] \tag{3.4}$$

Note that in Eq. (3.4), the term $\Delta\kappa \cdot dt^l$ denotes the changed time length for each infinitesimal time interval, $dt, t \in [t_0, t_f^l]$. Next, the final output vector of the system (3.1) can be expressed as follow by introducing a weighting matrix $\boldsymbol{W}(t) \in \boldsymbol{R}^{p \times n}$

$$\boldsymbol{Y}(\boldsymbol{X}(t_f)) = \boldsymbol{Y}(\boldsymbol{X}(t_f)) + \int_{t_0}^{t_f} \left[\boldsymbol{W}(t) \cdot (\boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}, t) - \dot{\boldsymbol{X}})\right] dt \tag{3.5}$$

We then differentiate both sides of Eq. (3.5), it gives

$$d\boldsymbol{Y}(\boldsymbol{X}(t_f)) = \frac{\partial \boldsymbol{Y}(\boldsymbol{X}(t))}{\partial \boldsymbol{X}(t)} \cdot d\boldsymbol{X}(t_f) + \int_{t_0}^{t_f} \left[\boldsymbol{W}(t) \cdot \frac{\partial \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}, t)}{\partial \boldsymbol{X}(t)} \cdot \delta \boldsymbol{X}(t) + \boldsymbol{W}(t) \cdot \frac{\partial \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}, t)}{\partial \boldsymbol{U}(t)} \cdot \delta \boldsymbol{U}(t) - \boldsymbol{W}(t) \cdot d\dot{\boldsymbol{X}}(t) \right] dt \tag{3.6}$$

Note that in Eq. (3.6), $d\boldsymbol{X}(t)$ denotes the differential of $\boldsymbol{X}(t)$ taking into account the differential change of time interval, $\Delta\kappa \cdot dt$, and $\delta \boldsymbol{X}$ denotes the variation in $\boldsymbol{X}$ when the time interval is assumed to be fixed. They have the relationship as follow

$$d\boldsymbol{X}(t) = \delta \boldsymbol{X}(t) + \dot{\boldsymbol{X}}(t) \cdot \Delta\kappa \cdot dt \tag{3.7}$$

We conduct the first order differential of Eq.(3.7) respect to the time, it yields

$$d\dot{\boldsymbol{X}}(t) = \delta \dot{\boldsymbol{X}}(t) + \dot{\boldsymbol{X}}(t) \cdot \Delta\kappa \tag{3.8}$$

Substituting the $d\dot{\boldsymbol{X}}(t)$ for Eq. (3.8) into the term of the integrand on right side of Eq. (3.6) leads to

$$d\boldsymbol{Y}(\boldsymbol{X}(t_f)) = \frac{\partial \boldsymbol{Y}(\boldsymbol{X}(t))}{\partial \boldsymbol{X}(t)} \cdot d\boldsymbol{X}(t_f) + \int_{t_0}^{t_f} \left[\boldsymbol{W}(t) \cdot \frac{\partial \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}, t)}{\partial \boldsymbol{X}(t)} \cdot \delta \boldsymbol{X}(t) + \boldsymbol{W}(t) \cdot \frac{\partial \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}, t)}{\partial \boldsymbol{U}(t)} \cdot \delta \boldsymbol{U}(t) - \boldsymbol{W}(t) \cdot \dot{\boldsymbol{X}}(t) \cdot \Delta\kappa - \boldsymbol{W}(t) \cdot \delta\dot{\boldsymbol{X}}(t) \right] dt \tag{3.9}$$

By integrating the last term on the right side of Eq. (3.6), we obtain

$$\begin{aligned} d\boldsymbol{Y}(\boldsymbol{X}(t_f)) = {} & \frac{\partial \boldsymbol{Y}(\boldsymbol{X}(t_f))}{\partial \boldsymbol{X}(t_f)} \cdot \delta \boldsymbol{X}(t_f) - [\boldsymbol{W}(t) \cdot \delta \boldsymbol{X}(t)]_{t=t_f} + [\boldsymbol{W}(t) \cdot \delta \boldsymbol{X}(t)]_{t=t_0} \\ & + \int_{t_0}^{t_f} \left[\boldsymbol{W}(t) \cdot \frac{\partial \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}, t)}{\partial \boldsymbol{X}(t)} \cdot \delta \boldsymbol{X}(t) + \dot{\boldsymbol{W}}(t) \cdot \delta \boldsymbol{X}(t) \right. \\ & \left. + \boldsymbol{W}(t) \cdot \frac{\partial \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}, t)}{\partial \boldsymbol{U}(t)} \cdot \delta \boldsymbol{U}(t) - \boldsymbol{W}(t) \cdot \dot{\boldsymbol{X}}(t) \cdot \Delta\kappa \right] dt \end{aligned} \tag{3.10}$$

Equation (3.10) can be further rearranged as

$$d\boldsymbol{Y}(\boldsymbol{X}(t_f)) = \left(\frac{\partial \boldsymbol{Y}(\boldsymbol{X}(t_f))}{\partial \boldsymbol{X}(t_f)} - [\boldsymbol{W}(t)]_{t=t_f}\right) \cdot \delta \boldsymbol{X}(t_f) + [\boldsymbol{W}(t) \cdot \delta \boldsymbol{X}(t)]_{t=t_0}$$
$$+ \int_{t_0}^{t_f} \left[\left(\boldsymbol{W}(t) \cdot \frac{\partial \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}, t)}{\partial \boldsymbol{X}(t)} + \dot{\boldsymbol{W}}(t) \right) \cdot \delta \boldsymbol{X}(t) + \boldsymbol{W}(t) \cdot \frac{\partial \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}, t)}{\partial \boldsymbol{U}(t)} \cdot \delta \boldsymbol{U}(t) \right.$$
$$\left. - \boldsymbol{W}(t) \cdot \dot{\boldsymbol{X}}(t) \cdot \Delta\kappa \right] dt \tag{3.11}$$

Here, it is necessary to choose $\boldsymbol{W}(t)$ so that eliminates the coefficients of $\delta \boldsymbol{X}(t)$ and $\delta \boldsymbol{X}(t_f)$ in Eq. (3.11), which leads to the following weighting matrix dynamics with the associated boundary condition at the final time t_f:

$$\dot{\boldsymbol{W}}(t) = -\boldsymbol{W}(t) \cdot \frac{\partial \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}, t)}{\partial \boldsymbol{X}(t)} \tag{3.12}$$

$$\boldsymbol{W}(t_f) = \frac{\partial \boldsymbol{Y}(\boldsymbol{X}(t_f))}{\partial \boldsymbol{X}(t_f)} \tag{3.13}$$

In Eq. (3.11), it is straight to obtain $\delta \boldsymbol{X}(t_0) = 0$ with the fact that the initial conditions are specified. Then use this observation and the weighting matrix dynamic as presented in Eqs. (3.12) and (3.13), the Eq. (3.11) can be further simplified as

$$\begin{aligned} d\boldsymbol{Y}(\boldsymbol{X}(t_f)) &= \int_{t_0}^{t_f} \left[\boldsymbol{W}(t) \cdot \frac{\partial \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}, t)}{\partial \boldsymbol{U}(t)} \cdot \delta \boldsymbol{U}(t) \right] dt + \int_{t_0}^{t_f} \left[-\boldsymbol{W}(t) \cdot \dot{\boldsymbol{X}}(t) \cdot \Delta\kappa \right] dt \\ &= \int_{t_0}^{t_f} [\boldsymbol{B}_s(t) \cdot \delta \boldsymbol{U}(t)]\, dt + \boldsymbol{B}_\kappa \cdot \Delta\kappa \end{aligned} \tag{3.14}$$

where

$$\boldsymbol{B}_s(t) \triangleq \boldsymbol{W}(t) \cdot \frac{\partial \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}, t)}{\partial \boldsymbol{U}(t)} \tag{3.15}$$

$$\boldsymbol{B}_\kappa \triangleq -\int_{t_0}^{t_f} \left[\boldsymbol{W}(t) \cdot \dot{\boldsymbol{X}}(t) \right] dt \tag{3.16}$$

where $\boldsymbol{B}_s(t)$ is the sensitivity matrix that relates the error $\delta \boldsymbol{U}$ to $d\boldsymbol{Y}$ as per Ref. [16]. And $\boldsymbol{B}_\kappa$ can be interpreted as the sensitivity matrix that relates the scale factor of time internal, $\Delta\kappa$ to the final out error $d\boldsymbol{Y}$. Note that in here, the sensitive relation for terminal time is indirectly formulated by the scale factor of time internal, $\Delta\kappa$. Compared with the original way in GS-MPSP, this strategy is more accurate since

it uniformly scales each infinitesimal time interval on $[t_0, t_f]$, rather than roughly adjust the final time.

3.2.2 The Mathematical of IGS-MPSP Method

In IGS-MPSP, the control vector is represented by a weighted summation of some basic spectral functions to reduce the number of various to be optimized:

$$\boldsymbol{U}(t) = \sum_{i=1}^{N_p} \boldsymbol{C}_i P_i(t) \tag{3.17}$$

where $\boldsymbol{C}_j = [c_{1j}, c_{2j}, \ldots, c_{mj}]^T$ is the coefficient vector corresponding to the jth spectral function. N_p is the number of basic functions in the expression, and $\boldsymbol{P}_j(t)$ is the basic spectral function. The spectral functions can be selected as Legendre series, Chebyshev series, etc.

Next, the new developed relationship as presented in Eq. (3.14) is applied to derive the GS-MPSP method for the free-time problem. Since the control history $\boldsymbol{U}(t), t \in [t_0, t_f]$ has been represented by the spectral functions in Eq. (3.17), and a new scale factor of time internal, $\Delta\kappa$, is introduced to adjust the unspecified final time as in Eq. (3.3), both the coefficient vector $[\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_{N_p}]$ and scale factor $\delta\kappa$ are selected as variable to be determined.

First, the variation of the control history during the time $t = [t_0, t_f]$ can be obtained from Eq. (3.17):

$$d\boldsymbol{U}(t) = \sum_{j=1}^{N_p} d\boldsymbol{C}_j P_j(t) \tag{3.18}$$

Substituting Eq. (3.18) into Eq. (3.14), it yields

$$\begin{aligned} d\boldsymbol{Y}_N &= \int_{t_0}^{t_f} \left[\boldsymbol{B}_s(t) \cdot \sum_{j=1}^{N_p} \delta\boldsymbol{C}_j P_j(t) \right] dt + \boldsymbol{B}_\kappa \cdot \Delta\kappa \\ &= \sum_{j=1}^{N_p} \boldsymbol{A}_j \cdot d\boldsymbol{C}_j + \boldsymbol{B}_\kappa \cdot \Delta\kappa \end{aligned} \tag{3.19}$$

in which

$$\boldsymbol{A}_j = \int_{t_0}^{t_f} \boldsymbol{B}_s(t) \cdot P_j(t) dt,\ j = 1, 2 \ldots N_p \tag{3.20}$$

In Eq. (3.19), $\boldsymbol{A}_j$ is the spectral sensitivity matrix as per Ref. [15], which relates the error of the coefficient of the jth spectral function, $d\boldsymbol{C}_j$, to the error of the output $d\boldsymbol{Y}_N$. Thus, a linear formula for the error of the final output and the error of each coefficient vector as well as the scale factor is obtained. Then the desired coefficients and scale factor can be worked out by formulating a static programming problem:

The update equation for coefficient vectors can be written as

$$\boldsymbol{C}_j^{l+1} = \boldsymbol{C}_j^l - d\boldsymbol{C}_j^l \tag{3.21}$$

where $\boldsymbol{C}_j^l$ denotes as the jth coefficient at the current iteration (represented by superscript l), and $\boldsymbol{C}_j^{l+1}$ denotes the updated coefficient in the next iteration (represented by superscript $l+1$). After substituting the expression of $d\boldsymbol{C}_j^l$ in Eq. (3.21) into Eq. (3.19), the error of final output can be written as

$$d\boldsymbol{Y}_N = \sum_{j=1}^{N_p} \boldsymbol{F}_j \delta \boldsymbol{C}_j^l = \sum_{j=1}^{N_p} F_j(\boldsymbol{C}_j^l - \boldsymbol{C}_j^{l+1}) = \boldsymbol{c}_\lambda - \sum_{j=1}^{N_p} \boldsymbol{A}_j \boldsymbol{C}_j^{l+1} + \boldsymbol{B}_\kappa \cdot \Delta\kappa \tag{3.22}$$

where

$$\boldsymbol{c}_\lambda = \sum_{j=1}^{N_p} \boldsymbol{A}_j \boldsymbol{C}_j^l \tag{3.23}$$

Equation (3.22) is apparently a linear equations set about $\boldsymbol{C}_j^{l+1}$ and $\Delta\kappa$, which contains $N_p \times m + 1$ unknowns and p equations where $N_p \times m + 1 > p$. Hence, Eq. (3.22) represents an under-constrained system and it is necessary to impose an appropriate performance index to facilitate a solution. In here, we consider to minimum the control effort, that is

$$J = \frac{1}{2} \int_{t_0}^{t_f^l} \left[\boldsymbol{U}^{l+1}(t)^T \boldsymbol{R}(t) \boldsymbol{U}^{l+1}(t) \right] \cdot dt + R_\kappa \Delta\kappa^2 \tag{3.24}$$

where $\boldsymbol{R}(t)$ and R_κ are the weight matrix for control and scale factor. Substituting Eq. (3.17) into Eq. (3.24), it yields

$$J = \frac{1}{2} \int_{t_0}^{t_f^l} \left[\left(\sum_{i=1}^{N_p} \boldsymbol{C}_i^{l+1} P_i(t) \right)^T \boldsymbol{R}(t) \left(\sum_{i=1}^{N_p} \boldsymbol{C}_i^{l+1} P_i(t) \right) \right] \cdot dt + R_\kappa \Delta\kappa^2 \tag{3.25}$$

Then, combined with the cost function (3.25) and the constraints given in Eq. (3.22), a static programming problem for the coefficient vector and scale vector can be formulated. The augmented cost function of this problem is given by

$$\bar{J} = J + \boldsymbol{\lambda}^T \left(d\boldsymbol{Y}_N - \boldsymbol{c}_\lambda + \sum_{j=1}^{N_p} \boldsymbol{A}_j \boldsymbol{C}_j^{l+1} - \boldsymbol{B}_\kappa \cdot \Delta\kappa \right) \tag{3.26}$$

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

$$\frac{\partial \bar{J}}{\partial d\boldsymbol{C}_j} = \sum_{i=1}^{N_p} \boldsymbol{R}_{ij} \cdot \boldsymbol{C}_j^{l+1} - \boldsymbol{A}_j{}^T \cdot \boldsymbol{\lambda} = \boldsymbol{0}, \quad j = 1, 2 \cdots N_p \tag{3.27}$$

$$\frac{\partial \bar{J}}{\partial \Delta\kappa} = R_\kappa \cdot \Delta\kappa - \boldsymbol{B}_\kappa{}^T \cdot \boldsymbol{\lambda} = \boldsymbol{0} \tag{3.28}$$

where

$$\boldsymbol{R}_{ij} = \int_{t_0}^{t_f^l} \left[P_i(t) \boldsymbol{R}(t) P_j(t) \right] \cdot dt \tag{3.29}$$

Thus, Eqs. (3.22), (3.27) and (3.28) make up a equation set about $\boldsymbol{C}_j^{l+1}$, $\boldsymbol{\lambda}$ and $\delta\kappa$, which can be written as a compact form as follow

$$\boldsymbol{D} \cdot \tilde{\boldsymbol{X}} = \boldsymbol{E} \tag{3.30}$$

where

$$\boldsymbol{D} = \begin{bmatrix} \boldsymbol{R}_{11} & \cdots & \boldsymbol{R}_{1N_p} & -\boldsymbol{A}_1{}^T & 0 \\ \vdots & \ddots & \vdots & \vdots & 0 \\ \boldsymbol{R}_{N_p 1} & \cdots & \boldsymbol{R}_{N_p N_p} & -\boldsymbol{A}_{N_p}{}^T & 0 \\ \boldsymbol{A}_1 & \cdots & \boldsymbol{A}_{N_p} 0 & -\boldsymbol{B}_\kappa & \\ 0 & \cdots & 0 & -\boldsymbol{B}_\kappa{}^T & R_\kappa \end{bmatrix}, \tilde{\boldsymbol{X}} = \begin{bmatrix} \boldsymbol{C}_1^{l+1} \\ \vdots \\ \boldsymbol{C}_{N_p}^{l+1} \\ \boldsymbol{\lambda} \\ \Delta\kappa \end{bmatrix}, \boldsymbol{E} = \begin{bmatrix} \boldsymbol{0} \\ \vdots \\ \boldsymbol{0} \\ \boldsymbol{c}_\lambda - d\boldsymbol{Y}_N \\ 0 \end{bmatrix} \tag{3.31}$$

The Eq. (3.30) contains $N_p \times m + p + 1$ unknowns (i.e., $\boldsymbol{C}_1^{l+1}, \boldsymbol{C}_2^{l+1} \cdots \boldsymbol{C}_{N_p}^{l+1}$, $\boldsymbol{\lambda}, \Delta\kappa$) and the same number of equations. Assuming that the matrix $\boldsymbol{D}$ is nonsingular, the unknown vector $\tilde{\boldsymbol{X}}$ can be solved by

$$\tilde{\boldsymbol{X}} = \boldsymbol{D}^{-1} \cdot \boldsymbol{E} \tag{3.32}$$

Consequently, the updated coefficients $\boldsymbol{C}_j^{l+1}$ and scale factor $\Delta\kappa$ are obtained from the solution of $\tilde{\boldsymbol{X}}$. Then substituting the $\Delta\kappa$ into Eq. (3.3), the update terminal time can be obtained by

$$t_f^{l+1} = t_f^l + \Delta\kappa \cdot (t_f^l - t_0) \tag{3.33}$$

And substitute $\boldsymbol{C}_j^{l+1}(j = 1, 2 \cdots N_p)$ into Eq. (3.17), the updated control history at time $t \in [t_0, t_f^{l+1}]$ is eventually given by

$$\boldsymbol{U}^{l+1}(t) = \sum_{i=1}^{N_p} \boldsymbol{C}_i^{l+1} P_i(t), t \in [t_0, t_f^{l+1}] \tag{3.34}$$

Remark 1 To implement the IGS-MPSP algorithm, the sensitivity matrix $\boldsymbol{A}_j$, $\boldsymbol{B}_\kappa$ and $\boldsymbol{R}_{ij}$ are necessary to be worked out in each iteration. The Gauss Quadrature Collocation method can be applied to efficiently compute such matrix and ensure the computational efficiency of this approach. The detailed procedure will be presented in the next subsection.

Remark 2 Compared with the original algorithm [15], the improved method introduces a scale factor of time internal to adjust the terminal time, and accordingly a sensitive relation for this factor is derived. This way improves the accuracy of sensitive relation for terminal time and hence is able to search the final time in a wide range when a poor initial guess is provided. That is, the convergence robustness for initial guess of final time is improved.

3.2.3 The Computation of Sensitive Matrix by Gauss Quadrature Collocation

In this subsection, the Gauss Quadrature Collocation method is applied to efficiently compute the sensitive matrix $\boldsymbol{A}_j$, $\boldsymbol{B}_\kappa$ and $\boldsymbol{R}_{ij}$. The detailed procedure is presented as follow.

For the convenience of solving, the physical time $t \in [t_0, t_f]$ is converted to the scale time $\tau \in [-1, 1]$ by the following relation:

$$t \equiv t(\tau, t_0, t_f) = \frac{t_f - t_0}{2}\tau + \frac{t_f + t_0}{2} \tag{3.35}$$

Next, the collocation method is used to solve the weighting dynamic equation as presented in Eqs. (3.12) and (3.13). First, we rewrite the matrix equation (3.12) as the following vector equation with the independent variable τ:

$$\begin{aligned} &\dot{\boldsymbol{W}}_k(\tau) = -\boldsymbol{W}_k(\tau) \cdot \boldsymbol{f}_x(\tau) \\ &k = 1, 2, \ldots, p \end{aligned} \tag{3.36}$$

where $\boldsymbol{W}_k(\tau)$ denotes the kth row vector of matrix $\boldsymbol{W}(t)$, and $\boldsymbol{f}_x(\tau)$ is defined by

$$\boldsymbol{f}_x(\tau) \triangleq \frac{\partial \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}, t)}{\partial \boldsymbol{X}(t)} \cdot \frac{t_f - t_0}{2} \tag{3.37}$$

Then N Lagrange interpolating polynomials $L_i(\tau)(i = 1, 2, \ldots, N)$ are used to appropriate both sides of Eq. (3.36), by which the differential equation can be converted to a series of algeria equations at specified collocation points $\tau_i(i = 1, 2 \cdots N)$. In here, the Gauss-Lobatto type collocation is used, such as Legendre-Gauss-Lobatto (LGL), or Chebyshev-Gauss-Lobatto (CGL) series. Note that, in principle $\boldsymbol{W}_k(\tau)$ must satisfy Eq. (3.36) at all collocation points $\tau_i(i = 1, 2, \ldots, N)$. However, $\boldsymbol{W}_k$ is generally computed by integrating the matrix dynamics (3.36) backward from τ_N to τ_1 since the value at the final time $t_f(\tau_N)$ is known. This means $\boldsymbol{W}_k(\tau)|_{\tau=\tau_1}$ is the last integral step as well as the integration result. Therefore, $\boldsymbol{W}_k(\tau)|_{\tau=\tau_1}$ is not necessary to strictly satisfy the differential equation (3.36), and we just consider the $N-1$ collocation points $\tau_i(i = 1, 2, \ldots, N)$ for the according collocation equations. Consequently, the collocation equations are given in the compact form:

$$(\boldsymbol{D} \otimes \boldsymbol{I}_n) \cdot \Omega_k = -\boldsymbol{f} \cdot \Omega_k \tag{3.38}$$

where $\Omega_k = [\boldsymbol{W}_k(\tau_1), \boldsymbol{W}_k(\tau_2), \ldots \boldsymbol{W}_k(\tau_N)]^T$; $\boldsymbol{I}_n$ is an $n \times n$ identity matrix and $\boldsymbol{D} \otimes \boldsymbol{I}_n$ denotes the Kronecker product of $\boldsymbol{D}$ and $\boldsymbol{I}_n$; $\boldsymbol{D} \in \mathbb{R}^{(N-1)\times N}$ is known as the differential matrix. The matrix $\boldsymbol{D}$ and $\boldsymbol{f}$ are given by

$$\boldsymbol{D} = \begin{vmatrix} \dot{L}_1(\tau_2) & \dot{L}_2(\tau_2) & \cdots & \dot{L}_N(\tau_2) \\ \dot{L}_1(\tau_3) & \dot{L}_2(\tau_3) & \cdots & \dot{L}_N(\tau_3) \\ \vdots & \vdots & \ddots & \vdots \\ \dot{L}_1(\tau_N) & \dot{L}_2(\tau_N) & \cdots & \dot{L}_N(\tau_N) \end{vmatrix}, \boldsymbol{f} = \boldsymbol{diag} \begin{vmatrix} \mathbf{0}_{n(N-1)\times n} & & & \\ & \boldsymbol{f}_x^T(\tau_2) & & \\ & & \ddots & \\ & & & \boldsymbol{f}_x^T(\tau_N) \end{vmatrix} \tag{3.39}$$

Equation (3.38) can be further simplified by

$$\boldsymbol{A}\Omega_k = 0 \tag{3.40}$$

where $\boldsymbol{A} = \boldsymbol{f} + (\boldsymbol{D} \otimes \boldsymbol{I}_n) \in \boldsymbol{R}^{(N-1)n\times Nn}$. Equation (3.40) contains $(N-1)n$ linear equations and the same number of unknowns (that is $\boldsymbol{W}_k(\tau_i)(i = 1, 2, \ldots, N-1)$). Defining the unknown vector as $\boldsymbol{X}_k = \left[\boldsymbol{W}_k(\tau_1), \boldsymbol{W}_k(\tau_2), \ldots \boldsymbol{W}_k(\tau_{N-1})\right]^T$, it is easy to obtain $\Omega_{\mathbf{k}} = \left[\boldsymbol{X_k}^{\mathbf{T}} \ \boldsymbol{W_k}(\tau_{\mathbf{N}})\right]^{\mathbf{T}}$. Next $\boldsymbol{A}$ 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 $\boldsymbol{A}$, respectively. Using these relations, the linear equations can be further expressed as

$$\boldsymbol{A}\Omega_k = [\boldsymbol{A}_F, \boldsymbol{A}_N] \cdot \begin{bmatrix} \boldsymbol{X}_k \\ \boldsymbol{W}_k(\tau_N)^T \end{bmatrix} = \boldsymbol{A}_F\boldsymbol{X}_k + \boldsymbol{A}_N\boldsymbol{W}_k(\tau_N)^T = 0 \tag{3.41}$$

Assuming that the matrix $\boldsymbol{A}_F$ is nonsingular, the $\boldsymbol{X}_k$ is eventually solved by

$$\boldsymbol{X}_k = -\boldsymbol{A}_F^{-1}\boldsymbol{A}_N \cdot \boldsymbol{W}_k(\tau_N)^T \tag{3.42}$$

The solution of $\boldsymbol{X}_k$ gives the value of kth row of matrix $\boldsymbol{W}(\tau)$ at the collocation points τ_k $(\tau_1, \tau_2, \ldots, \tau_{N-1})$. 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.

Subsequently, the sensitivity matrix $\boldsymbol{B}_s(t)$ at collocation points can be calculated out according to Eq. (3.15):

$$\boldsymbol{B}_s(\tau_i) = \boldsymbol{W}(\tau_i) \cdot \left. \frac{\partial \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}, t)}{\partial \boldsymbol{U}(t)} \right|_{t=\tau_i}, i = 1, 2, \ldots, N \tag{3.43}$$

Lastly, the principle of Gaussian quadrature [16] is applied to compute the sensitive matrix $\boldsymbol{A}_j$,$\boldsymbol{B}_\kappa$ and $\boldsymbol{R}_{ij}$:

$$\begin{aligned} &\boldsymbol{A}_j = \int_{t_0}^{t_f} \boldsymbol{B}_s(t) \cdot P_j(t) dt = \frac{t_f - t_0}{2} \sum_{i=1}^{N} \boldsymbol{B}_s(\tau_i) \cdot P_j(\tau_i) \cdot \eta_i \\ &j = 1, 2, \ldots, N_p \end{aligned} \tag{3.44}$$

$$\boldsymbol{B}_\kappa \triangleq -\int_{t_0}^{t_f} \left[\boldsymbol{W}(t) \cdot \dot{\boldsymbol{X}}(t) \right] = -\frac{t_f - t_0}{2} \sum_{i=1}^{N} \boldsymbol{W}(\tau_i) \cdot \dot{\boldsymbol{X}}(\tau_i) \cdot \eta_i \tag{3.45}$$

$$\begin{aligned} &\boldsymbol{R}_{ij} = \int_{t_0}^{t_f} \left[P_i(t) \boldsymbol{R}(t) P_j(t) \right] \cdot dt = \frac{t_f - t_0}{2} \sum_{k=1}^{N} P_i(\tau_k) \boldsymbol{R}(\tau_k) P_j(\tau_k) \cdot \eta_k \\ &i, j = 1, 2, \ldots, N_p \end{aligned} \tag{3.46}$$

where η_i is the weight coefficient of Gaussian quadrature corresponding to the collocation point τ_i. In this way, such sensitive matrix is obtained by a set of algebraic operation at very few collocation points.

Remark 3 In the calculation loop for each row of the matrix $\boldsymbol{W}$, the matrix $\boldsymbol{A}$ remains unchanged and just need to be computed once, since which only upon to the given collocated points $\tau_i (i = 1, 2, \ldots, N)$ and $\boldsymbol{f}_x(\tau_i)$. This feature effectively reduces the computational complexity.

Remark 4 Since the spectral sensitivity matrix is directly worked out by the Gaussian quadrature method in Eqs. (3.42)–(3.46) and avoids heavy computational consumption produced by the numerical integration of a series of matrix differential equations, the computational efficiency is improved significantly.

3.2.4 The Implementation Step of IGS-MPSP

The implementation procedure of this approach is provided in Algorithm 1. This method starts from the initial guess for spectral coefficients and terminal time. Then the final output errors and trajectory state are evaluated out. 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 are updated. Then, the updated control history is generated, and the output errors are evaluated again. This iterative procedure is repeated until a specified criterion for the terminal output errors is met.

Algorithm 1: IGS-MPSP algorithm

Step 1: Initialize the initial guess $\boldsymbol{C}_j^0, t_f^0$, the stopping criterions δ_y, the number of spectral function N_p, the number of collocation points N

Step 2: For $k = 0, 1, 2, \ldots$

(2.1) Compute the control history $\boldsymbol{U}^k(t), t \in [t_0, t_f]$ by Eq.(3.17);
Integrate the system dynamic to obtain the trajectory state as well as the output error, $\|d\boldsymbol{Y}_N\|_\infty$;

(2.2) If $\|d\boldsymbol{Y}\|_\infty < \delta_y$
Output the current control history $\boldsymbol{U}^k(t)$ and break the iteration;
Otherwise continue the iteration.

(2.3) Compute the sensitive matrix $\boldsymbol{A}_j$, $\boldsymbol{B}_\kappa$ and $\boldsymbol{R}_{ij}$ according to Eq.(3.42) − Eq.(3.46);
Solve the linear equation (3.32) to obtain the updated solution;

(2.4) Update the terminal time by
$t_f^{l+1} = t_f^l + \Delta\kappa \cdot (t_f^l - t_0)$

(2.5) Obtain the update coefficient vector $\boldsymbol{C}_j^{k+1}$ from the updated solution $\tilde{\boldsymbol{X}}$;

Since the spectral coefficients have no physical meaning, it's not straightforward to assign an initial guess with appropriate values. Therefore, the least-squares algorithm is applied to obtain the initial guess of the spectral coefficients when an initial guess of control sequence is provided.

Denoting $\hat{\boldsymbol{C}}^0 = [\hat{\boldsymbol{C}}_1^0, \hat{\boldsymbol{C}}_2^0, \ldots, \hat{\boldsymbol{C}}_{N_p}^0]$ as the initial guess of the spectral coefficients, as per Eq. (3.17), the control vector represented by the guess $\hat{\boldsymbol{C}}^0$ at time step $t_k, k = 1, 2, \ldots, n$ can be written as

$$\hat{\boldsymbol{U}}_k^0 = \sum_{j=1}^{N_p} P_j(t_k)\hat{\boldsymbol{C}}_j^0, k = 1, 2, \ldots, n \tag{3.47}$$

In matrix form, Eq. (3.47) can be written as

$$\hat{\boldsymbol{U}}^0 = \hat{\boldsymbol{C}}^0 \boldsymbol{P} \tag{3.48}$$

where $\hat{\boldsymbol{U}}^0 = [\hat{\boldsymbol{U}}_1^0, \hat{\boldsymbol{U}}_2^0, \ldots, \hat{\boldsymbol{U}}_n^0]$, and

$$\boldsymbol{P} = \begin{vmatrix} P_1(t_1) & P_1(t_2) & \dots & P_1(t_n) \\ P_2(t_1) & P_2(t_2) & \dots & P_2(t_n) \\ \vdots & \vdots & \ddots & \vdots \\ P_{N_p}(t_1) & P_{N_p}(t_2) & \dots & P_{N_p}(t_n) \end{vmatrix} \tag{3.49}$$

If the initial control guess sequence $\boldsymbol{U}^0 = [\boldsymbol{U}_1^0, \boldsymbol{U}_2^0, \dots, \boldsymbol{U}_n^0]$ is given, the proper spectral function coefficients $\hat{\boldsymbol{C}}^0$ are to be found so as to minimize $|\hat{\boldsymbol{U}}^0 - \boldsymbol{U}^0|$. According to the principle of the least squares, the coefficients are estimated as

$$\hat{\boldsymbol{C}}^0 = \boldsymbol{U}^0 \boldsymbol{P}^T \cdot (\boldsymbol{P}\boldsymbol{P}^T)^{-1} \tag{3.50}$$

The initial guess of the spectral coefficients is obtained from Eq. (3.50).

3.3 The Ascent Predictive Guidance Under Thrust Drop Fault

In this subsection, the proposed method is employed to solve the ascent guidance problem of launch vehicle under thrust drop fault. The problem formula is firstly introduced, then the detailed procedure to address this problem is presented.

3.3.1 Problem Formulation

To be solved conveniently, a modified orbital inertial (MOI) coordinate system is firstly defined as follow. As shown in Fig. 3.1, P_F is the position of launch vehicle when the fault occurs; P_F' is the projection of P_F onto the injected orbital plane; P_f is the nominal injection point. Then, the origin of this modified orbital coordinate (MOC) is located at the center of the earth. The coordinate plane coincides with the injected orbital plane $OP_F'P_f$, in which the axis O_eY directing to the midpoint of the arc $P_F'P_f$ and the axis O_eX perpendicular to the O_eY. Lastly, the axis O_eZ is determined by the right-hand-thread rule.

Note that the MOI coordinate can be determined by the position of launch vehicle at the time that the fault occurs and the injected orbit information (the inclination i_f, longitude of ascending node Ω_f, and injection point for nominal trajectory). The relationship between the Modified orbital inertial (MOI) coordinate and the Earth-centered inertial (ECI) system is given by

$$\mathrm{X}^{MOI} = \mathrm{M}_{ECI}^{MOI}(\Omega_f, i_f, P'_F) \cdot \mathrm{X}^{ECI} \tag{3.51}$$

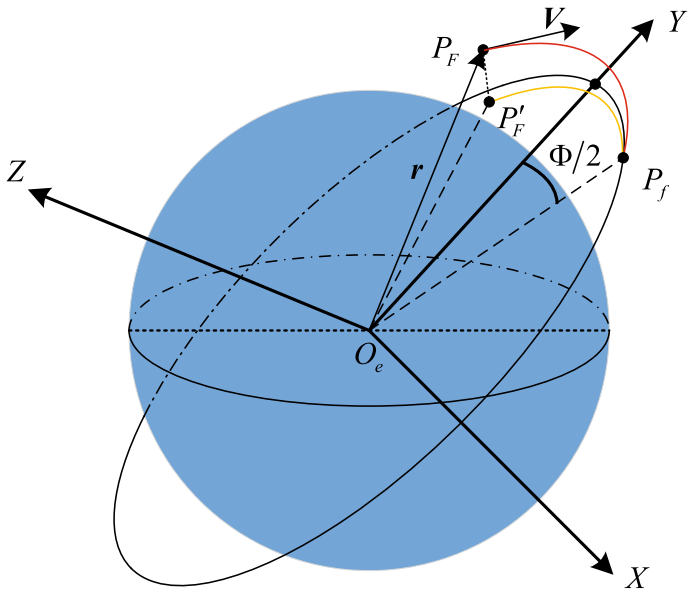


Fig. 3.1 Modified orbital inertial (MOI) coordinate system

where M_{ECI}^{MOI} denotes the transformation matrix from the ECI coordinate system to the MOI system.

It is considered the thrust drop fault occurs at the second stage of the launch vehicle. In this flight stage, the launch vehicle is assumed to fly out of the dense atmosphere and the aerodynamic forces can be ignored. Therefore, the three-dimensional point-mass dynamic equations of launch vehicles build in the (MOI) coordinate is given as follow:

$$\begin{cases} \dot{\boldsymbol{r}} = \boldsymbol{V} \\ \dot{\boldsymbol{V}} = T \cdot \boldsymbol{e}_T / m \ - \mu \boldsymbol{r} / r^3 \\ \dot{m} = -m_e \end{cases} \tag{3.52}$$

where $\boldsymbol{r} = [r_x, r_y, r_z]^T$ is the position vector in the MOI coordinate system; $\boldsymbol{V} = [V_x, V_y, V_z]^T$ is the inertial velocity vector; T is the thrust magnitude, which is considered to be constant; m is the mass of vehicle and m_e is the mass flow rate; $\boldsymbol{e}_T$ denotes the thrust direction vector, which is generally aligned with the body longitudinal axis of the vehicle and can be given by

$$\boldsymbol{e}_T = [\cos\varphi \cos\psi, \sin\varphi \cos\psi, -\sin\psi]^T \tag{3.53}$$

where φ and ψ are the pitch angle and yaw angle relative to the MOI coordinate system, respectively. The dynamic equations as presented in Eqs. (3.52) and (3.53) can be written as the compact form

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) \tag{3.54}$$

where $\boldsymbol{x} = [r_x, r_y, r_z, V_x, V_y, V_z]^T$ is the state vector and $\boldsymbol{u} = [\varphi, \psi]^T$ is the control vector of the system.

Remark 5 Since the flight path angle and angle of attack of the launch vehicle is generally small in the second stage, the defined MOI and the according dynamic equation will ensure the pitch angle of the vehicle remain a small value. This can effectively reduce the nonlinearity of the controls as presented in Eq. (3.53) and improve the convergence of the algorithm. Moreover, such definition can simplify the terminal constraint to be introduced later.

3.3.2 Terminal Constraints

It is assumed that the thrust fault of the launch vehicle takes place at the initial time t_0, and the corresponding states are given by:

$$\boldsymbol{X}(t_0) = \boldsymbol{X}_0 \tag{3.55}$$

The final orbital injection time t_f is constrained by

$$t_f \leq t_{f,\max} \tag{3.56}$$

where $t_{f,\max}$ is the maximum burn time of the vehicle, which is determined by the remaining fuel and mass flow rate.

The terminal constraints of ascent guidance are determined by the orbital insertion conditions, which are generally provided by the semi-major axis a_f, eccentricity e_f, orbital inclination i_f, and longitude of ascending node Ω_f. In here, we consider to entry into a circular orbit ($e_f = 0$). Then, the first two conditions can be equivalently described by

$$\left\| \boldsymbol{r}(t_f) \right\| = r_f^* \tag{3.57}$$

$$\left\| \boldsymbol{V}(t_f) \right\| = V_f^* \tag{3.58}$$

$$\boldsymbol{r}^T(t_f)\boldsymbol{V}(t_f) = 0 \tag{3.59}$$

Moreover, in the modified orbital inertial (MOI) coordinate system, the final two orbital insertion conditions i_f and Ω_f are equivalent to make the final position vector component $r_z(t_f)$ and velocity vector component $V_z(t_f)$ to be zero. Therefore, the terminal constraints of this ascent guidance problem are defined by

$$h(\boldsymbol{x}(t_f)) = \begin{bmatrix} \|\boldsymbol{r}(t_f)\| = r_f^* \\ \|\boldsymbol{V}(t_f)\| = V_f^* \\ \boldsymbol{r}^T(t_f)\boldsymbol{V}(t_f) \\ r_z(t_f) \\ V_z(t_f) \end{bmatrix} = \mathbf{0} \tag{3.60}$$

Thus, the ascent guidance problem can be organized by

$\boldsymbol{P}^0$: find t_f, $\mathrm{u}(t)$, $t \in [t_0, t_f]$

subject to:

$$\mathrm{x}(t) = \mathrm{f}(\mathrm{x}(t),\mathrm{u}(t),t) \tag{3.61}$$

$$\mathrm{x}(t_0) = \mathrm{x}_0 \tag{3.62}$$

$$h(\mathrm{x}(t_f)) = 0 \tag{3.63}$$

$$t_f \leq t_{f,\max} \tag{3.64}$$

3.3.3 Solved by the IGS-MPSP

As introduced earlier, the proposed algorithm is able to solve the free-final time guidance problem. However, this algorithm cannot directly handle the inequality constraint as given in Eq. (3.64). Therefore, a numerical trick is additionally conducted to address this constraint.

First, the proposed method is employed to solve the problem $\boldsymbol{P}^0$ in which the constraint (3.64) is omitted. Then the obtained terminal time t_f is checked. If this value is smaller than the maximum $t_{f,\max}$, it means the solution is feasible and the obtained control history can be used as the renewed commands of the launch vehicle. Otherwise, it implies the solution is infeasible for this problem. That is, the launch vehicle cannot directly entry into the required orbit. In this situation, a new guidance strategy is needed, such as entering into a new parking orbit or transfer orbit. This case is beyond the scope of this work.

The detailed implementation steps are summarized as follow. As presented in Fig. 3.2, the guidance strategy is triggered by a fault detection. Then the current state t_0, x_0 is employed as the initial state of the proposed algorithm, and the terminal time and control of nominal trajectory is used as the initial guess. Obviously, these controls and terminal time guess cannot steer the launch vehicle to well meet the required terminal states in the presence of thrust drop fault. Thus, the proposed algorithm is iteratively conducted to obtain the updated terminal time and control history. If the updated terminal time is smaller than the maximum value, then the corresponding control history is directly used as the new guidance for the launch vehicle. Otherwise, it implies that the launch vehicle cannot directly inject into the original orbit.

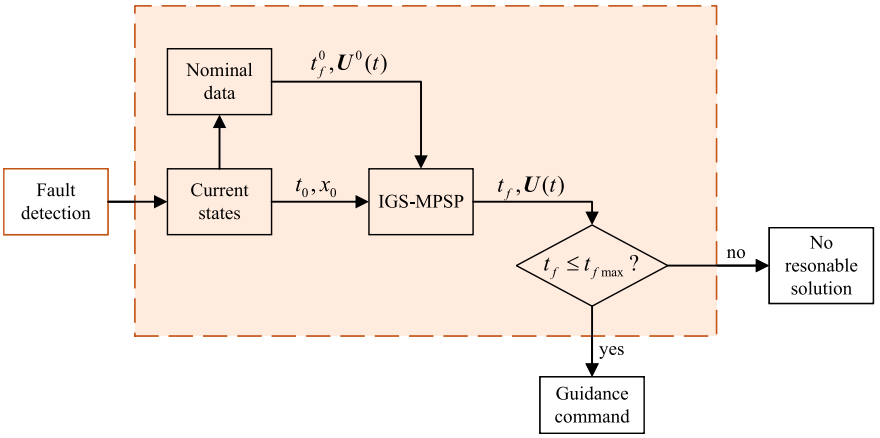


Fig. 3.2 Ascent guidance strategy for launch vehicle under thrust drop fault

3.4 Numerical Results

In this section, the numeric simulation is carried out to demonstrate the performance of the proposed method in term of accuracy and computational efficiency. It is considered that the fault of thrust drop occurs at the second stage flight phase of a launch vehicle. The nominal parameters of the launch vehicle are illustrated in Table 3.1. Both the initial conditions (at the fault occurring time) and target orbit's parameters are given in Table 3.2.

The proposed method is employed to solve such guidance problem when the thrust of the launch vehicle drops to 70% and 80% of the nominal value, respectively. In the algorithm implementation, the six order Legendre polynomials are used as the spectral function of control, and the Legendre-Gauss-Latto (LGL) points are selected as the collocation points in computing the spectral sensitivity matrix. In addition, the number of LGL nodes is taken as 15.

Additionally, a SOCP based method [11] is conducted in here as the comparison of the proposed method. This algorithm takes the minimum fuel consumption as the optimization object, and the classical Euler method is used to discrete the problems. The number of discretization nodes is set to be 50, which is determined by comprehensively considering the solving accuracy and efficiency.

Table 3.1 Nominal parameters of the launch vehicle

Parameter	Value	Unit
Initial mass m_0	27.769	ton
Dry mass m_{dry}	17	ton
Nominal thrust T	98	KN
Dry mass m_e	22.138	kg/s

Table 3.2 Initial conditions and target orbit

Parameter	Value	Unit
Initial position vector $\boldsymbol{r}_0$	[368502, 6508822, 7201]	m
Initial velocity vector $\boldsymbol{v}_0$	[6148.49, 99.79, −18.13]	m/s
Target orbit semi-major axis a_f	6578145	m
Target orbit eccentricity e_f	0	
Target orbit inclination i_f	0.5068	rad
Target orbit ascending node Ω_f	6.2368	rad

In simulations, the terminal time and control history of nominal trajectory are used as the initial guess of the proposed method and SOCP based method. Moreover, all numerical simulations are implemented in the MATLAB 2021a environment on a personal desktop (Intel i7-8750H, 3.2 GHz). The CVX [17] optimization toolbox with SDPT3 4.0 [18] is employed as the solver of the SOCP based method.

3.4.1 The Results by the Proposed Method

The proposed method reaches the required tolerance of terminal conditions by 8 iterations. Figure 3.3 depicts the altitude profiles of the trajectories with the thrust of 70% and 80% nominal value. Additionally, the nominal trajectory and the trajec-

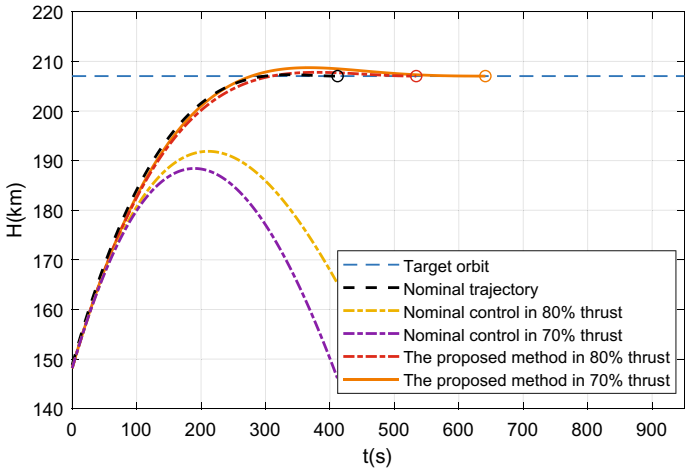
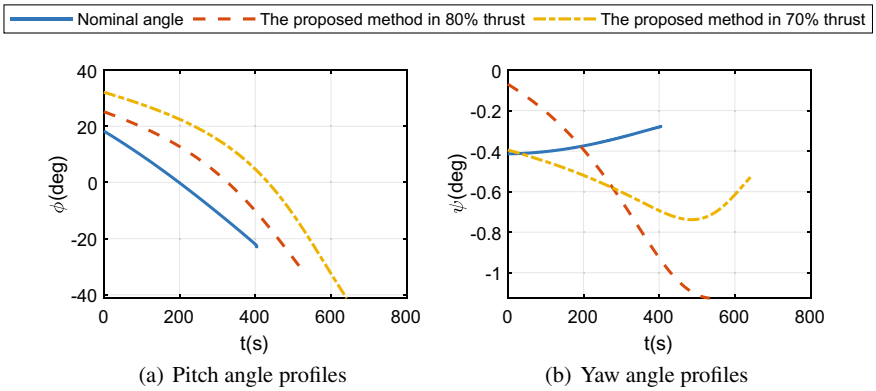


Fig. 3.3 Altitude profiles for nominal trajectory and the trajectories obtained by IGS-MPSP



(a) Pitch angle profiles (b) Yaw angle profiles

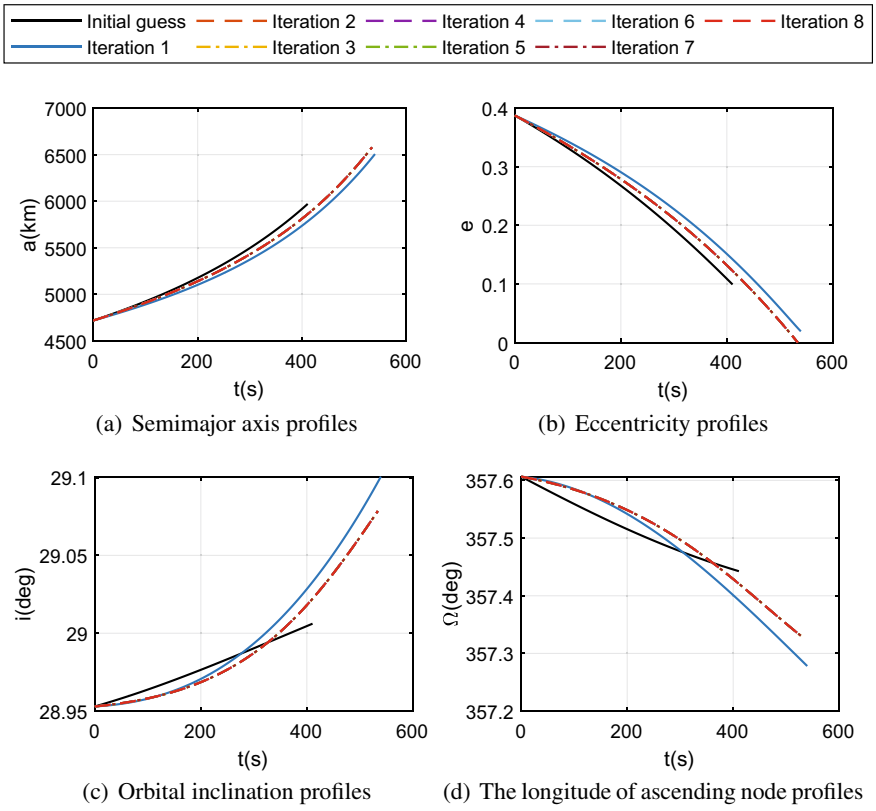
Fig. 3.4 Control profiles for the nominal trajectory

tories obtained by the nominal control in the case of thrust drop are also provided in Fig. 3.3. It can be seen that with the nominal control, the launch vehicle fails to enter into the target orbit under the thrust drop. The proposed method succeeds in regenerating the updated guidance commands (shown in Fig. 3.4) to steer the vehicle into the original target orbit. Specially, the trajectory states for each iteration of the proposed algorithm when the thrust is 80% of nominal value is presented in Fig. 3.5. It clearly reflects that the proposed method reaches the required orbit injection parameters by a few iterations, even if the relatively poor control and terminal time guess (nominal trajectory) are given. At the same time, the orbit injection time is considerably increased as compared to the nominal value, but still smaller than the maximum allowable value. These results demonstrate the effectiveness of the proposed method, which is able to re-plan the ascent trajectory under the thrust drop, and search the appropriate orbit injection time when a relatively accurate guess cannot not be given out.

3.4.2 Comparison with SOCP Method

Furthermore, the comparison between the proposed method and SOCP based method are provided in Table 3.3, Figs. 3.6 and 3.7.

The Table 3.3 illustrates the terminal mass and terminal time achieved by the proposed algorithm and SOCP based method, respectively. It can be noted that the results obtained by the proposed algorithm are very close to that produced by SOCP method. The deviations are less than 0.025%. This means the proposed method achieves a near optimality for the fuel consumption compared to the SOCP based method. Moreover, the control histories and trajectories for the proposed method and SOCP are depicted in Figs. 3.6 and 3.7. As it can be seen, the control profiles as well as the trajectories by the proposed method are similar to those of SOCP, no matter in



(a) Semimajor axis profiles (b) Eccentricity profiles

(c) Orbital inclination profiles (d) The longitude of ascending node profiles

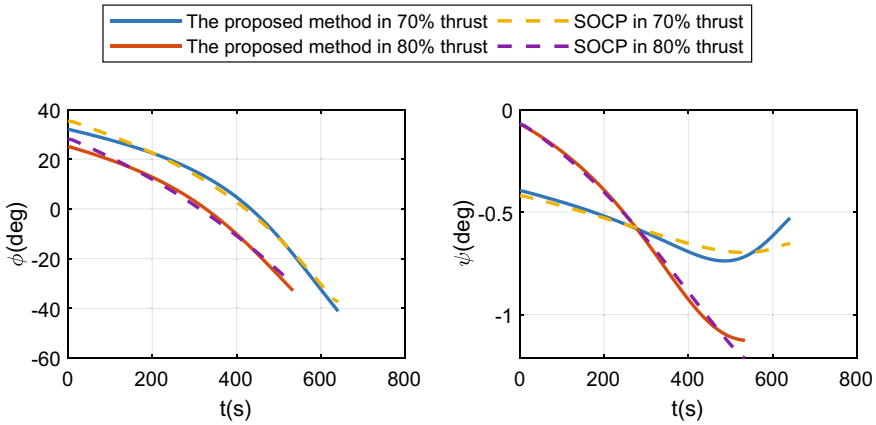
Fig. 3.5 The trajectory for each iteration in the 80% of nominal thrust

Table 3.3 The results of final time and final mass

	80% of nominal thrust		70% of nominal thrust	
	t_f (s)	m_f (kg)	t_f (s)	m_f (kg)
The proposed method	533.9370	18312.7483	641.2647	17831.5629
SOCP	533.6772	18317.3497	641.2007	17832.5554

the case of 70% of the nominal thrust or 80% of the nominal thrust. This demonstrates the proposed method produces approximate effect of SOCP based method to solve the ascent guidance problem with the thrust drop fault.

Lastly, the computational efficiency of the proposed method and SOCP method are comparatively investigated by conducting the same simulation case. Figure 3.8 and Table 3.4 present the CPU time consumed by two methods, in which the time elapses for one iteration and the total are shown. It can be seen that, the CPU time consumed by the proposed method is almost one-sixtieth or one-seventieth of that



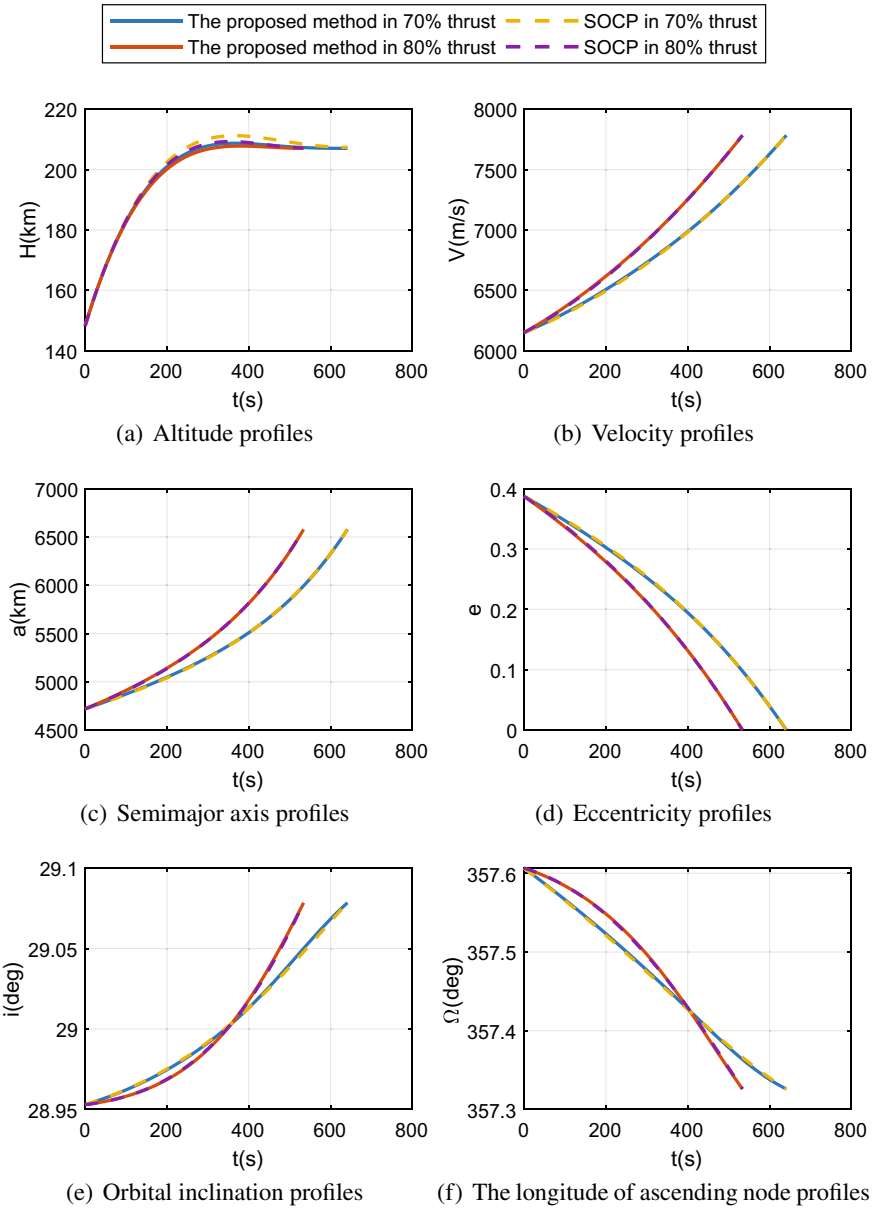
(a) Pitch angle profiles (b) Yaw angle profiles

Fig. 3.6 The Control profile for IGS-MPSP and SOCP method

by the SOCP method for one iteration and the total value. This result clearly demonstrates the superiority of the proposed method in the computational speed. Such highly computational efficiency is achieved by a series of careful design such as the spectral representation of control, sensitive matrix computation conducted by collocation method. Hence, the proposed method owns the great potential for online application.

3.5 Conclusion

In this chapter, a predictive ascent guidance based on the IGS-MPSP for the thrust drop fault of the launch vehicle is presented. Firstly, an IGS-MPSP method is derived. Compared with the original GS-MPSP method, this approach introduces a scale factor for the time internal as the additional variable to adjust the terminal time. Then, a new sensitive relation for the final time is established. Since the accuracy of the sensitive relation is improved, the approach owns the better performance to search the appropriate final time in the presence of the poor initial guess. Hence, it is more suitable for the ascent guidance problem. Secondly, the application of the proposed method for the ascent guidance problem under the thrust drop fault is detailed introduced. The numerical simulation for a typical case and comparison with the SOCP based method are carried out. The results indicate the effectiveness of the proposed method, which generate approximate results with the SOCP method but with considerably higher computing efficiency.



(a) Altitude profiles (b) Velocity profiles

(c) Semimajor axis profiles (d) Eccentricity profiles

(e) Orbital inclination profiles (f) The longitude of ascending node profiles

Fig. 3.7 Trajectories by various methods

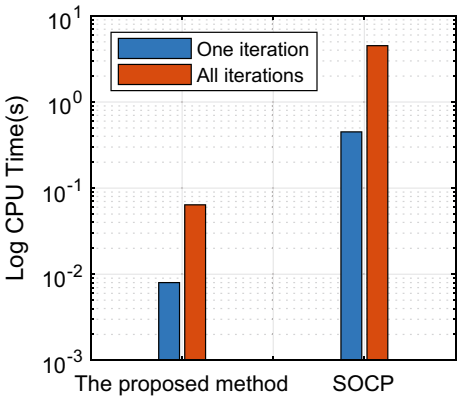


Fig. 3.8 CPU time consumed by various methods

Table 3.4 The CPU time consumed by various methods

Method	CPU time for one iteration (s)	Iterations number	Total CPU time(s)
SOCP	0.451	10	4.512
IGS-MPSP	0.008	8	0.064

References

1. I.E. Smith, General formulation of the iterative guidance mode, in *NASA TM X-53414* (1966)
2. R. Jaggers, An explicit solution to the exoatmospheric powered flight guidance and trajectory optimization problem for rocket propelled vehicles, in *Guidance and Control Conference* (1977), p. 1051
3. E.-J. Song, S. Cho, W.-R. Roh, A comparison of iterative explicit guidance algorithms for space launch vehicles. Adv. Space Res. **55**(1), 463–476 (2015)
4. K. Mall, M.J. Grant, E. Taheri, Uniform trigonometrization method for optimal control problems with control and state constraints. J. Spacecr. Rocket. **57**(5), 995–1007 (2020)
5. M.H. Gräßlin, J. Telaar, U.M. Schöttle, Ascent and reentry guidance concept based on NLP-methods. Acta Astronaut. **55**(3–9), 461–471 (2004)
6. P. Lu, Introducing computational guidance and control (2017)
7. X. Liu, L. Ping, B. Pan, Survey of convex optimization for aerospace applications. Astrodynamics **1**(1), 23–40 (2017)
8. X. Liu, Z. Shen, L. Ping, Entry trajectory optimization by second-order cone programming. J. Guid. Control Dyn. **39**(2), 227–241 (2016)
9. M. Szmuk, T.P. Reynolds, B. Açıkmeşe, Successive convexification for real-time six-degree-of-freedom powered descent guidance with state-triggered constraints. J. Guid. Control Dyn. **43**(8), 1399–1413 (2020)
10. X. Cheng, H. Li, R. Zhang, Efficient ascent trajectory optimization using convex models based on the Newton-Kantorovich/Pseudospectral approach. Aerosp. Sci. Technol. **66**, 140–151 (2017)
11. Y. Li, B. Pang, C. Wei, N. Cui, Y. Liu, Online trajectory optimization for power system fault of launch vehicles via convex programming. Aerosp. Sci. Technol. **98**, 105682 (2020)
12. Z. Song, C. Wang, Q. Gong, Joint dynamic optimization of the target orbit and flight trajectory of a launch vehicle based on state-triggered indices. Acta Astronaut. **174**, 82–93 (2020)

13. Z. Hao, R. Zhang, Onboard real-time generation of launch vehicle abort orbits. J. Guid. Control Dyn. **44**(8), 1541–1549 (2021)
14. R. Padhi, M. Kothari, Model predictive static programming: a computationally efficient technique for suboptimal control design. Int. J. Innov. Comput. Inf. Control **5**(2), 399–411 (2009)
15. C. Zhou, X. Yan, S. Tang, Generalized quasi-spectral model predictive static programming method using gaussian quadrature collocation. Aerosp. Sci. Technol. **106**, 106134 (2020)
16. 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), 1897–1913 (2014)
17. M. Grant, S. Boyd, CVX: Matlab software for disciplined convex programming, version 2.1. (2016)
18. K.-C. Toh, M.J. Todd, R.H. Tütüncü, On the implementation and usage of SDPT3–a Matlab software package for semidefinite-quadratic-linear programming, version 4.0, in *Handbook on Semidefinite, Conic and Polynomial Optimization* (Springer, Berlin, 2012), pp. 715–754