Journal of Guidance, Control, and Dynamics, Volume 46, Issue 4, April 2023, pp. 742-751
DOI:10.2514/1.G007113

Game-Theoretic Flexible-Final-Time Differential Dynamic Programming using Gaussian Quadrature

Xiaobo Zheng *, Haodong Guan †, Defu Lin ‡, Shaoming He §
Beijing Institute of Technology, Beijing 100081, China

Hyo-Sang Shin ¶
Cranfield University, Cranfield MK430AL, UK

I. Introduction

Trajectory optimization is a key enabler to improve the level of autonomy of an aerial vehicle [1–3]. Differential dynamic programming (DDP) has become a widespread method to solve nonlinear trajectory optimization problems due to its well-established problem formulation framework and fast convergence characteristics in recent years [4–6]. DDP origins from classical dynamic programming (DP) that leverages the Bellman optimality principle and overcomes the curse of dimensionality of DP by a second-order approximation of the value function to decompose the original problem into a succession of subproblems of smaller dimensions [7, 8]. The key feature of DDP is that it provides linear complexity with respect to the length of time horizon and ensures a theoretical second-order convergence rate [9].

The main issue associated with the original DDP is that it requires accurate dynamics model in solving the optimization problem. The dynamics model, however, usually suffers from unknown uncertainties and external disturbances, e.g., unknown wind in fixed-wing aerial vehicle path optimization [10–12], unknown target maneuver in the missile-target interception problem [13–15]. Using inaccurate dynamics model in the original DDP, the performance of the resultant solution might degrade drastically and the local optimality is no longer guaranteed [16]. To address this issue, the authors in [16, 17] utilized differential game theory to develop a min-max DDP or so-called Game-Theoretic Differential Dynamic Programming (GT-DDP) algorithm. This approach mitigates the effect of model inaccuracy by formulating the system control input and uncertainty or disturbance as two competitors in a game theory framework. However, the implementation of GT-DDP requires to predefine a terminal/final time, which is difficult to set an optimal value beforehand for a free-final-time optimization problem since the final time is naturally a function of the control input. This drawback, therefore, prohibits the application of GT-DDP for some realistic trajectory optimization problems. Although we revisited the original DDP by dynamically optimizing the final time in conjunction with the control input in our previous work [18], the extension to the GT-DDP framework is not straightforward and needs careful adjustment.

*PhD Student, School of Aerospace Engineering, 5th Zhongguancun South Street
†Undergraduate Student, School of Aerospace Engineering, 5th Zhongguancun South Street
‡Professor, School of Aerospace Engineering, 5th Zhongguancun South Street
§Professor, School of Aerospace Engineering, 5th Zhongguancun South Street, Member AIAA, Email: shaoming.he@bit.edu.cn (Corresponding Author)
¶Professor, School of Aerospace, Transport and Manufacturing, College Road, Member AIAA

Published by AIAA. This is the Author Accepted Manuscript issued with: Creative Commons Attribution Non-Commercial License (CC:BY:NC 4.0).
The final published version (version of record) is available online at DOI:10.2514/1.G007113. Please refer to any applicable publisher terms of use.

In general, DDP and its variants operate in a discrete-time manner and the commonly-used discretization approach is the Euler discretization with equally spaced time instants, i.e., the time interval between two time instants is the same across the entire optimization process. Compared to the Euler discretization approach, it is known that the Gaussian quadrature rule provides improved algebraic precision with less discretization nodes [19–21]. Typical Gaussian quadrature rules that are leveraged in solving nonlinear programming problems are the Legendre-Gauss quadrature, Legendre-Gauss-Labatto quadrature and Legendre-Gauss-Radau (LGR) quadrature [22]. A comprehensive comparison of these three different methods was reported in [23] and the results indicate that the LGR method provides more accurate approximations while maintaining a relatively low-dimensional approximation of the problem. This observation motivates us to leverage the LGR quadrature in DDP for the purpose of further reducing the computational complexity.

This Note revisits the original GT-DDP method and proposes a new GT-FFT-LGRDDP algorithm: Game-Theoretic Flexible-Final-Time Legendre-Gauss-Radau Differential Dynamic Programming, to solve the typical zero-sum differential game problem. We first derive the system dynamics with LGR nodes discretization and leverage the Bellman optimality principle to find the optimal control correction of every iteration. The Taylor expansion of the value function with respect to the final time is then utilized and the first-order optimality condition is used to obtain the optimal final time variation in the neighborhood of the current solution at each iteration. Except for removing the heuristic setting of the final time, another benefit of optimizing the terminal time is that we can dynamically minimize flight time for aerial vehicles in some time-critical missions. Up to the best of our knowledge, this is the first time to leverage the Gaussian quadrature in DDP and revisit GT-DDP into a flexible-final-time framework.

An impact-angle-constraint maneuvering target interception guidance problem with energy and flight time minimization [13, 24] is considered as an example to apply the proposed GT-FFT-LGRDDP algorithm. Notice that the target maneuver is naturally unpredictable since the future movement of a target is generally uncertain, and setting a final time beforehand is also difficult for this problem. We first formulate this problem as a zero-sum differential game, where the objective is to minimize the control energy while satisfying the terminal impact angle constraint. In order to find a feedback guidance command of the interceptor, the proposed GT-FFT-LGRDDP algorithm is leveraged to solve an optimization problem at every time instant in a receding horizon manner. The comparison results under different conditions clearly verify that the proposed algorithm provides higher terminal impact precision than other methods.

II. Backgrounds and Preliminaries

In this section, we first introduce the general zero-sum differential game optimization problem and then review the min-max GT-DDP method for the completeness of the Note.

A. Zero-Sum Differential Game Problem

For a typical zero-sum differential game problem, there are two noncooperating players with opposite objectives in the game. The zero-sum differential game problem can be established in a min-max form, where the dynamics of the game system involving two players can be modeled as

$$\dot{\boldsymbol{x}} = \boldsymbol{F}\left(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}(t)\right), \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{1}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^{m_x}$ is the state of the dynamic game system, and $\boldsymbol{u}(t) \in \mathbb{R}^{m_u}$ and $\boldsymbol{v}(t) \in \mathbb{R}^{m_v}$ denote two opposite controls within the game system at $t \in [t_0, t_f]$. t_f is the terminal time of the game and $\boldsymbol{x}_0$ stands for the initial state of the system. The system of the game evolves with a dynamic function $\boldsymbol{F}\left(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}(t)\right)$. The two opposite objectives of the players can be incorporated in a single objective function of the following form

$$J\left(\boldsymbol{u}, \boldsymbol{v}\right) = \Phi(\boldsymbol{x}(t_f), t_f) + \int_{t_0}^{t_f} \mathcal{L}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}(t))\mathrm{d}t \tag{2}$$

where scalars function Φ and $\mathcal{L}$ denote the terminal cost and the running cost, respectively.

For a given game problem that $\boldsymbol{u}(t)$ minimizes the cost function while $\boldsymbol{v}(t)$ maximizes the cost function, the optimal strategy pair $(\boldsymbol{u}^*, \boldsymbol{v}^*)$ that finds the min-max value of the cost function satisfies

$$J(\boldsymbol{u}^*, \boldsymbol{v}) \leq J(\boldsymbol{u}^*, \boldsymbol{v}^*) \leq J(\boldsymbol{u}, \boldsymbol{v}^*) \tag{3}$$

B. Min-Max Game-Theoretic DDP

DDP is an approximate DP algorithm that has been well recognized in solving nonlinear optimization problems to address the issue of curse of dimensionality. The GT-DDP, which is a variant of DDP, is specifically developed to solve the differential game problem with a min-max form [16]. This subsection reviews the discrete-time GT-DDP algorithm for the completeness of the Note. To begin with, the dynamics model of the continuous-time game system in Eq. (1) at $t \in [t_0, t_f]$ is first discretized using simple Euler discretization method as

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{F}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k)\delta t, \quad t \in \{0, 1, \cdots, N\} \tag{4}$$

where the state and the controls at time instant t_k are represented as $\boldsymbol{x}_k$, $\boldsymbol{u}_k$ and $\boldsymbol{v}_k$, respectively, and $t_N = t_f$. Notice that the time interval between two time instants, i.e., $\delta t = t_{k+1} - t_k$, remains constant across the entire time duration.

The original DDP or GT-DDP requires to set a fixed value of the final time. With this in mind, the terminal constraint index $\Phi(\boldsymbol{x}_N, t_N)$ depends only on the terminal state, i.e., $\Phi(\boldsymbol{x}_N, t_N) = \Phi(\boldsymbol{x}_N)$. The objective function in

Eq. (2) consequently can be discretized as

$$J = \Phi(\boldsymbol{x}_N) + \sum_{k=0}^{N-1} \mathcal{L}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k)\delta t \tag{5}$$

The key idea of GT-DDP is to convert the problem of optimizing the entire control sequence into the problem of optimizing a single control input $\boldsymbol{u}_k$ backward sequentially using the Bellman optimality principle. Accordingly, the min-max value of the cost function, also called the value function, with two opposite controls can be mathematically formulated as the following form

$$\begin{aligned} V(\boldsymbol{x}_k) &= \min_{\boldsymbol{U}} \max_{\boldsymbol{V}} \left\{ \Phi(\boldsymbol{x}_N) + \sum_{k=0}^{N-1} \mathcal{L}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k)\delta t \right\} \\ &= \min_{\boldsymbol{U}} \max_{\boldsymbol{V}} \{ L(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k) + V(\boldsymbol{x}_{k+1}) \} \end{aligned} \tag{6}$$

where $L(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k) = \mathcal{L}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k)\delta t$, $\boldsymbol{U} = \{\boldsymbol{u}_0, \ldots, \boldsymbol{u}_k, \ldots, \boldsymbol{u}_N\}$ and $\boldsymbol{V} = \{\boldsymbol{v}_0, \ldots, \boldsymbol{v}_k, \ldots, \boldsymbol{v}_N\}$.

Notice that the system dynamics is generally nonlinear and hence it is usually intractable to find an analytical optimal control pair $\boldsymbol{u}_k$ and $\boldsymbol{v}_k$ using Eq. (6) in practice. To address this problem, the GT-DDP utilizes second-order Taylor expansion around the nominal system state $\boldsymbol{x}_k$ to approximate the value function as

$$V(\boldsymbol{x}_k + \delta\boldsymbol{x}_k) \approx V(\boldsymbol{x}_k) + V_{\boldsymbol{x}}(\boldsymbol{x}_k)^T \delta\boldsymbol{x}_k + \frac{1}{2}\delta\boldsymbol{x}_k^T V_{\boldsymbol{xx}}(\boldsymbol{x}_k)\delta\boldsymbol{x}_k \tag{7}$$

where $\delta\boldsymbol{x}_k$ denotes the state deviation from the nominal state trajectory at time t_k.

Define $Q(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k) = L(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k) + V(\boldsymbol{x}_{k+1})$ as the action-value function. Then, we can also approximate the action-value function using second-order Taylor series as

$$\begin{aligned} Q(\boldsymbol{x}_k + \delta\boldsymbol{x}_k, \boldsymbol{u}_k + \delta\boldsymbol{u}_k, \boldsymbol{v}_k + \delta\boldsymbol{v}_k) \approx{}& Q(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k) + Q_{\boldsymbol{x}}^T \delta\boldsymbol{x}_k + Q_{\boldsymbol{u}}^T \delta\boldsymbol{u}_k + Q_{\boldsymbol{v}}^T \delta\boldsymbol{v}_k \\ &+ \frac{1}{2} \begin{bmatrix} \delta\boldsymbol{x}_k \\ \delta\boldsymbol{u}_k \\ \delta\boldsymbol{v}_k \end{bmatrix}^T \begin{bmatrix} Q_{\boldsymbol{xx}} & Q_{\boldsymbol{xu}} & Q_{\boldsymbol{xv}} \\ Q_{\boldsymbol{ux}} & Q_{\boldsymbol{uu}} & Q_{\boldsymbol{uv}} \\ Q_{\boldsymbol{vx}} & Q_{\boldsymbol{vu}} & Q_{\boldsymbol{vv}} \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{x}_k \\ \delta\boldsymbol{u}_k \\ \delta\boldsymbol{v}_k \end{bmatrix} \end{aligned} \tag{8}$$

where $\delta\boldsymbol{u}_k$ and $\delta\boldsymbol{v}_k$ are the control deviations from the nominal trajectory at time t_k. The gradient vectors and Hessian

matrices of the state-action value function $Q(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k)$ can be readily obtained by the definition as

$$\begin{aligned}
Q_{\boldsymbol{x}} &= L_{\boldsymbol{x}} + A_k^T V_{\boldsymbol{x}}(\boldsymbol{x}_{k+1}), \quad Q_{\boldsymbol{u}} = L_{\boldsymbol{u}} + B_{\boldsymbol{u}_k}^T V_{\boldsymbol{x}}(\boldsymbol{x}_{k+1}) \\
Q_{\boldsymbol{v}} &= L_{\boldsymbol{v}} + B_{\boldsymbol{v}_k}^T V_{\boldsymbol{x}}(\boldsymbol{x}_{k+1}), \quad Q_{\boldsymbol{xx}} = L_{\boldsymbol{xx}} + A_k^T V_{\boldsymbol{xx}}(\boldsymbol{x}_{k+1}) A_k \\
Q_{\boldsymbol{uu}} &= L_{\boldsymbol{uu}} + B_{\boldsymbol{u}_k}^T V_{\boldsymbol{xx}}(\boldsymbol{x}_{k+1}) B_{\boldsymbol{u}_k}, \quad Q_{\boldsymbol{vv}} = L_{\boldsymbol{vv}} + B_{\boldsymbol{v}_k}^T V_{\boldsymbol{xx}}(\boldsymbol{x}_{k+1}) B_{\boldsymbol{v}_k} \\
Q_{\boldsymbol{xu}} &= L_{\boldsymbol{xu}} + A_k^T V_{\boldsymbol{xx}}(\boldsymbol{x}_{k+1}) B_{\boldsymbol{u}_k} \quad Q_{\boldsymbol{xv}} = L_{\boldsymbol{xv}} + A_k^T V_{\boldsymbol{xx}}(\boldsymbol{x}_{k+1}) B_{\boldsymbol{v}_k} \\
Q_{\boldsymbol{uv}} &= L_{\boldsymbol{uv}} + B_{\boldsymbol{u}_k}^T V_{\boldsymbol{xx}}(\boldsymbol{x}_{k+1}) B_{\boldsymbol{v}_k}, \quad Q_{\boldsymbol{vu}} = Q_{\boldsymbol{uv}}^T, \quad Q_{\boldsymbol{vx}} = Q_{\boldsymbol{xv}}^T, \quad Q_{\boldsymbol{ux}} = Q_{\boldsymbol{xu}}^T
\end{aligned} \tag{9}$$

where the notation $(\cdot)_{\boldsymbol{x}}$ refers to the partial derivative of $(\cdot)$ with respect to $\boldsymbol{x}$. The matrices A_k, $B_{\boldsymbol{u}_k}$ and $B_{\boldsymbol{v}_k}$ are state Jacobian and control Jacobian evaluated on the nominal trajectory at time t_k as $A_k = I + \boldsymbol{F}_{\boldsymbol{x}}\delta t$, $B_{\boldsymbol{u}_k} = \boldsymbol{F}_{\boldsymbol{u}}\delta t$, $B_{\boldsymbol{v}_k} = \boldsymbol{F}_{\boldsymbol{v}}\delta t$ with $\boldsymbol{F}_{\boldsymbol{x}}$, $\boldsymbol{F}_{\boldsymbol{u}}$, $\boldsymbol{F}_{\boldsymbol{v}}$ being $\boldsymbol{F}_{\boldsymbol{x}}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k)$, $\boldsymbol{F}_{\boldsymbol{u}}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k)$, $\boldsymbol{F}_{\boldsymbol{v}}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k)$, respectively.

Since Eq. (8) is a quadratic function with respect to control perturbations $\delta\boldsymbol{u}_k$ and $\delta\boldsymbol{v}_k$, the optimal control correction $\delta\boldsymbol{u}_k^*$ and $\delta\boldsymbol{v}_k^*$ can be readily derived by using the first-order optimality condition as

$$\begin{aligned}
\delta\boldsymbol{u}_k^* &= -Q_{\boldsymbol{uu}}^{-1}\left(Q_{\boldsymbol{u}} + Q_{\boldsymbol{ux}}\delta\boldsymbol{x}_k + Q_{\boldsymbol{uv}}\delta\boldsymbol{v}_k^*\right) \\
\delta\boldsymbol{v}_k^* &= -Q_{\boldsymbol{vv}}^{-1}\left(Q_{\boldsymbol{v}} + Q_{\boldsymbol{vx}}\delta\boldsymbol{x}_k + Q_{\boldsymbol{vu}}\delta\boldsymbol{u}_k^*\right)
\end{aligned} \tag{10}$$

Solving Eq. (10) results in the explicit optimal control correction as

$$\delta\boldsymbol{u}_k^* = k_{\boldsymbol{u}} + K_{\boldsymbol{u}}\delta\boldsymbol{x}_k, \quad \delta\boldsymbol{v}_k^* = k_{\boldsymbol{v}} + K_{\boldsymbol{v}}\delta\boldsymbol{x}_k \tag{11}$$

where the feed-forward terms $k_{\boldsymbol{u}}$, $k_{\boldsymbol{v}}$ and the feed-back terms $K_{\boldsymbol{u}}$, $K_{\boldsymbol{v}}$ are defined as

$$\begin{aligned}
k_{\boldsymbol{u}} &= -\left(Q_{\boldsymbol{uu}} - Q_{\boldsymbol{uv}}Q_{\boldsymbol{vv}}^{-1}Q_{\boldsymbol{vu}}\right)^{-1}\left(Q_{\boldsymbol{u}} - Q_{\boldsymbol{uv}}Q_{\boldsymbol{vv}}^{-1}Q_{\boldsymbol{v}}\right) \\
k_{\boldsymbol{v}} &= -\left(Q_{\boldsymbol{vv}} - Q_{\boldsymbol{vu}}Q_{\boldsymbol{uu}}^{-1}Q_{\boldsymbol{uv}}\right)^{-1}\left(Q_{\boldsymbol{v}} - Q_{\boldsymbol{vu}}Q_{\boldsymbol{uu}}^{-1}Q_{\boldsymbol{u}}\right) \\
K_{\boldsymbol{u}} &= -\left(Q_{\boldsymbol{uu}} - Q_{\boldsymbol{uv}}Q_{\boldsymbol{vv}}^{-1}Q_{\boldsymbol{vu}}\right)^{-1}\left(Q_{\boldsymbol{ux}} - Q_{\boldsymbol{uv}}Q_{\boldsymbol{vv}}^{-1}Q_{\boldsymbol{vx}}\right) \\
K_{\boldsymbol{v}} &= -\left(Q_{\boldsymbol{vv}} - Q_{\boldsymbol{vu}}Q_{\boldsymbol{uu}}^{-1}Q_{\boldsymbol{uv}}\right)^{-1}\left(Q_{\boldsymbol{vx}} - Q_{\boldsymbol{vu}}Q_{\boldsymbol{uu}}^{-1}Q_{\boldsymbol{ux}}\right)
\end{aligned} \tag{12}$$

Substituting the optimal control correction from Eq. (11) back into Eqs. (7) and (8) and then grouping the corresponding terms with the same order of $\delta\boldsymbol{x}$ yields

$$\begin{aligned}
V(\boldsymbol{x}_k) &= Q(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k) + k_{\boldsymbol{u}}^T Q_{\boldsymbol{u}} + k_{\boldsymbol{v}}^T Q_{\boldsymbol{v}} + \frac{1}{2}(k_{\boldsymbol{u}}^T Q_{\boldsymbol{uu}} k_{\boldsymbol{u}} + k_{\boldsymbol{v}}^T Q_{\boldsymbol{vv}} k_{\boldsymbol{v}} + k_{\boldsymbol{u}}^T Q_{\boldsymbol{uv}} k_{\boldsymbol{v}} + k_{\boldsymbol{v}}^T Q_{\boldsymbol{vu}} k_{\boldsymbol{u}}) \\
V_{\boldsymbol{x}}(\boldsymbol{x}_k) &= Q_{\boldsymbol{x}} + K_{\boldsymbol{u}}^T Q_{\boldsymbol{u}} + K_{\boldsymbol{v}}^T Q_{\boldsymbol{v}} + Q_{\boldsymbol{xu}} k_{\boldsymbol{u}} + Q_{\boldsymbol{xv}} k_{\boldsymbol{v}} + K_{\boldsymbol{u}}^T Q_{\boldsymbol{uu}} k_{\boldsymbol{u}} + K_{\boldsymbol{v}}^T Q_{\boldsymbol{vv}} k_{\boldsymbol{v}} + K_{\boldsymbol{u}}^T Q_{\boldsymbol{uv}} k_{\boldsymbol{v}} + K_{\boldsymbol{v}}^T Q_{\boldsymbol{vu}} k_{\boldsymbol{u}} \\
V_{\boldsymbol{xx}}(\boldsymbol{x}_k) &= Q_{\boldsymbol{xx}} + K_{\boldsymbol{u}}^T Q_{\boldsymbol{ux}} + K_{\boldsymbol{v}}^T Q_{\boldsymbol{vx}} + Q_{\boldsymbol{xu}} K_{\boldsymbol{u}} + Q_{\boldsymbol{xv}} K_{\boldsymbol{v}} + K_{\boldsymbol{u}}^T Q_{\boldsymbol{uu}} K_{\boldsymbol{u}} + K_{\boldsymbol{v}}^T Q_{\boldsymbol{vv}} K_{\boldsymbol{v}} + K_{\boldsymbol{u}}^T Q_{\boldsymbol{uv}} K_{\boldsymbol{v}} + K_{\boldsymbol{v}}^T Q_{\boldsymbol{vu}} K_{\boldsymbol{u}}
\end{aligned} \tag{13}$$

Given initial solution guesses of the control sequence $\boldsymbol{U}$ and $\boldsymbol{V}$, GT-DDP initializes the terminal value function $V(\boldsymbol{x}_N)$ as $V(\boldsymbol{x}_N) = \Phi(\boldsymbol{x}_N)$ as well as its derivatives and runs a backward process from $k = N$ to $k = 0$ to sequentially find the optimal control correction $\delta\boldsymbol{u}_k$ and $\delta\boldsymbol{v}_k$. Hence, the one-step control update is given by

$$\begin{aligned} \boldsymbol{u}_k &= \boldsymbol{u}_k + \delta\boldsymbol{u}_k^* = \boldsymbol{u}_k + K_{\boldsymbol{u}}\delta\boldsymbol{x}_k + k_{\boldsymbol{u}} \\ \boldsymbol{v}_k &= \boldsymbol{v}_k + \delta\boldsymbol{v}_k^* = \boldsymbol{v}_k + K_{\boldsymbol{v}}\delta\boldsymbol{x}_k + k_{\boldsymbol{v}} \end{aligned} \tag{14}$$

Once the backward pass is completed, a forward pass is triggered to find a new nominal state trajectory by substituting the updated control sequence into system dynamics (4). The GT-DDP algorithm iteratively runs the backward pass and forward pass until the convergence condition is triggered. A commonly used convergence criterion is that the difference of the cost function between two consecutive iterations is less than a lower threshold, i.e., $|J_i - J_{i-1}| \leq \epsilon$, where J_i represents the cost function at the ith iteration and $\epsilon > 0$ is a small positive constant. Notice that tuning the termination threshold factor ϵ is a tradeoff between the optimization performance and computational efficiency: smaller value ensures accurate convergence performance but a higher computational burden.

C. Motivation

Notice that implementing GT-DDP requires to set a fixed final time, which is generally intractable for some realistic mission scenarios since the time sequence of the system trajectory is naturally a function of the control input. For example, in the missile-target interception scenario, the interception time is usually unpredictable since the future movement of the target is uncertain and hence setting a heuristic value might degrade the optimization performance. We can also note from the previous descriptions that GT-DDP leverages simple Euler discretization with equally spaced time intervals. Since the Gaussian quadrature provides improved numerical accuracy with reduced number of discrete nodes, the objective of this Note is to revisit the GT-DDP using Gaussian quadrature in a flexible-final-time framework.

III. Flexible-Final-Time GT-DDP using Gaussian Quadrature Scheme

In this section, the details of the flexible-final-time GT-DDP using Gaussian quadrature scheme are presented with rigorous mathematical derivation. A LGR quadrature scheme is leveraged to improve the numerical precision of GT-DDP and then we use the first-order optimality to revisit GT-DDP in the flexible-final-time framework.

A. Game-Theoretic Legendre-Gauss-Radau DDP

In this subsection, we modify the GT-DDP algorithm by applying the Gaussian quadrature rule. Specifically, the LGR quadrature nodes and weights are utilized in the time discretization. The resultant algorithm is then termed as Game-Theoretic Legendre-Gauss-Radau Differential Dynamic Programming (GT-LGRDDP) method. Unlike the collocation method, there are no collocation points but integral points in the GT-LGRDDP method. Analogous to the

Gaussian quadrature rule [23], the time variable $t \in [t_0, t_f]$ is mapped into the scaled time variable $\tau \in [-1, 1]$ as

$$t = \frac{t_f - t_0}{2}\tau + \frac{t_f + t_0}{2} \tag{15}$$

With the preceding time transformation, a general form of a fucntion integral can be obtained as

$$\int_{t_0}^{t_f} h(t)\mathrm{d}t = \frac{t_f - t_0}{2}\int_{-1}^{1} h(\tau)\mathrm{d}\tau \tag{16}$$

where the integral over $\tau \in [-1, 1]$ can be approximated by the well-established Legendre-Gauss-type quadrature formula with a sum of n predefined weighted values of $h(\tau)$ as [23]

$$\int_{-1}^{1} h(\tau)\mathrm{d}\tau \approx \sum_{k=0}^{n} w_k h(\tau_k) \tag{17}$$

where w_k is the Gaussian quadrature weight of node τ_k. Note that w_k and τ_k depend only on the type of orthogonal polynomial that has been chosen [23]. Let $L_n(\tau)$ denote the n-th order Legendre polynomial, the LGR nodes are defined as the roots of $L_n(\tau) + L_{n+1}(\tau)$, where $L_{n+1}(\tau)$ is the $(n+1)$-th order Legendre polynomial and the nodes can be expressed as $\boldsymbol{\tau} = [\tau_0, \tau_1, \cdots, \tau_k, \cdots, \tau_{n-1}, \tau_n]$ with $\tau_k < \tau_{k+1}$. Notice that the location of the LGR nodes are distributed over $\tau \in [-1, 1)$, i.e., $\tau_0 = -1$ and $\tau_n < 1$. The corresponding weights w_k are determined by

$$w_k = \frac{1}{(n+1)^2}\frac{1 - \tau_k}{[L_n(\tau_k)]^2}, \quad k = \{0, 1, \cdots, n\} \tag{18}$$

which is proved to provide the $2n$ degree algebraic precision.

Using the LGR quadrature rule, the system state at scaled time instant τ_k after transforming the time duration from $t \in \left[t_0, t_f\right]$ to $\tau \in [-1, 1]$ can then be easily derived as

$$\boldsymbol{x}(\tau_k) = \boldsymbol{x}(\tau_0) + \frac{t_f - t_0}{2}\sum_{i=0}^{k-1} w_i \boldsymbol{F}(\boldsymbol{x}(\tau_i), \boldsymbol{u}(\tau_i), \boldsymbol{v}(\tau_i)) \tag{19}$$

which directly leads to the discrete system dynamics using LGR quadrature rule as

$$\boldsymbol{x}(\tau_{k+1}) = \boldsymbol{x}(\tau_k) + \frac{t_f - t_0}{2} w_k \boldsymbol{F}(\boldsymbol{x}(\tau_k), \boldsymbol{u}(\tau_k), \boldsymbol{x}(\tau_k)) \tag{20}$$

Since the LGR quadrature rule only changes the time scale of the original system and hence it can be easily verified

that the Bellman optimality principle still holds as

$$V(\boldsymbol{x}(\tau_k)) = \min_{\boldsymbol{u}(\tau_k)} \max_{\boldsymbol{v}(\tau_k)} \left\{ V(\boldsymbol{x}(\tau_{k+1})) + \hat{L}(\boldsymbol{x}(\tau_k), \boldsymbol{u}(\tau_k), \boldsymbol{v}(\tau_k)) \right\} \tag{21}$$

where

$$\hat{L}(\boldsymbol{x}(\tau_k), \boldsymbol{u}(\tau_k), \boldsymbol{v}(\tau_k)) = \frac{t_f - t_0}{2} w_k \mathcal{L}(\boldsymbol{x}(\tau_k), \boldsymbol{u}(\tau_k), \boldsymbol{v}(\tau_k)) \tag{22}$$

which means that we can use similar recursion as GT-DDP to derive the optimal solution. Hence, we define the action-value function as $\hat{Q}(\boldsymbol{x}(\tau_k), \boldsymbol{u}(\tau_k), \boldsymbol{v}(\tau_k)) = \hat{L}(\boldsymbol{x}(\tau_k), \boldsymbol{u}(\tau_k), \boldsymbol{v}(\tau_k)) + V(\boldsymbol{x}(\tau_{k+1}))$, and approximate $\hat{Q}(\boldsymbol{x}(\tau_k), \boldsymbol{u}(\tau_k), \boldsymbol{v}(\tau_k))$, $V(\boldsymbol{x}(\tau_{k+1}))$ using second-order Taylor expansion around the nominal trajectory. Following some simple algebraic manipulations, the gradient vectors and Hessian matrices of the action-value function in the LGR framework can be obtained as

$$\begin{aligned}
&\hat{Q}_{\boldsymbol{x}} = \hat{L}_{\boldsymbol{x}} + \hat{A}_k^T V_{\boldsymbol{x}}(\boldsymbol{x}(\tau_{k+1})), \quad \hat{Q}_{\boldsymbol{u}} = \hat{L}_{\boldsymbol{u}} + \hat{B}_{\boldsymbol{u}_k}^T V_{\boldsymbol{x}}(\boldsymbol{x}(\tau_{k+1})) \\
&\hat{Q}_{\boldsymbol{v}} = \hat{L}_{\boldsymbol{v}} + \hat{B}_{\boldsymbol{v}_k}^T V_{\boldsymbol{x}}(\boldsymbol{x}(\tau_{k+1})), \quad \hat{Q}_{\boldsymbol{xx}} = \hat{L}_{\boldsymbol{xx}} + \hat{A}_k^T V_{\boldsymbol{xx}}(\boldsymbol{x}(\tau_{k+1}))\hat{A}_k \\
&\hat{Q}_{\boldsymbol{uu}} = \hat{L}_{\boldsymbol{uu}} + \hat{B}_{\boldsymbol{u}_k}^T V_{\boldsymbol{xx}}(\boldsymbol{x}(\tau_{k+1}))\hat{B}_{\boldsymbol{u}_k}, \quad \hat{Q}_{\boldsymbol{vv}} = \hat{L}_{\boldsymbol{vv}} + \hat{B}_{\boldsymbol{v}_k}^T V_{\boldsymbol{xx}}(\boldsymbol{x}(\tau_{k+1}))\hat{B}_{\boldsymbol{v}_k} \\
&\hat{Q}_{\boldsymbol{xu}} = \hat{L}_{\boldsymbol{xu}} + \hat{A}_k^T V_{\boldsymbol{xx}}(\boldsymbol{x}(\tau_{k+1}))\hat{B}_{\boldsymbol{u}_k}, \quad \hat{Q}_{\boldsymbol{xv}} = \hat{L}_{\boldsymbol{xv}} + \hat{A}_k^T V_{\boldsymbol{xx}}(\boldsymbol{x}(\tau_{k+1}))\hat{B}_{\boldsymbol{v}_k} \\
&\hat{Q}_{\boldsymbol{uv}} = \hat{L}_{\boldsymbol{uv}} + \hat{B}_{\boldsymbol{u}_k}^T V_{\boldsymbol{xx}}(\boldsymbol{x}(\tau_{k+1}))\hat{B}_{\boldsymbol{v}_k}, \quad \hat{Q}_{\boldsymbol{vu}} = \hat{Q}_{\boldsymbol{uv}}^T, \quad \hat{Q}_{\boldsymbol{vx}} = \hat{Q}_{\boldsymbol{xv}}^T, \quad \hat{Q}_{\boldsymbol{ux}} = \hat{Q}_{\boldsymbol{xu}}^T
\end{aligned} \tag{23}$$

where $\hat{A}_k$, $\hat{B}_{\boldsymbol{u}_k}$, $\hat{B}_{\boldsymbol{v}_k}$ are Jacobians of state and controls, which are governed by

$$\begin{aligned}
&\hat{A}_k = I + \frac{t_f - t_0}{2} w_k \boldsymbol{F}_{\boldsymbol{x}}(\boldsymbol{x}(\tau_k), \boldsymbol{u}(\tau_k), \boldsymbol{v}(\tau_k)), \quad \hat{B}_{\boldsymbol{u}_k} = \frac{t_f - t_0}{2} w_k \boldsymbol{F}_{\boldsymbol{u}}(\boldsymbol{x}(\tau_k), \boldsymbol{u}(\tau_k), \boldsymbol{v}(\tau_k)) \\
&\hat{B}_{\boldsymbol{v}_k} = \frac{t_f - t_0}{2} w_k \boldsymbol{F}_{\boldsymbol{v}}(\boldsymbol{x}(\tau_k), \boldsymbol{u}(\tau_k), \boldsymbol{v}(\tau_k))
\end{aligned} \tag{24}$$

By replacing the gradients and Hessian matrices in the original action-value function (9) with their modifications (23), the optimal control correction at scaled time instant τ_k can be derived as

$$\begin{aligned}
\delta\hat{\boldsymbol{u}}^*(\tau_k) &= -\hat{Q}_{\boldsymbol{uu}}^{-1} \left(\hat{Q}_{\boldsymbol{u}} + \hat{Q}_{\boldsymbol{ux}} \delta\boldsymbol{x}(\tau_k) + \hat{Q}_{\boldsymbol{uv}} \delta\hat{\boldsymbol{v}}^*(\tau_k) \right) \\
\delta\hat{\boldsymbol{v}}^*(\tau_k) &= -\hat{Q}_{\boldsymbol{vv}}^{-1} \left(\hat{Q}_{\boldsymbol{v}} + \hat{Q}_{\boldsymbol{vx}} \delta\boldsymbol{x}(\tau_k) + \hat{Q}_{\boldsymbol{vu}} \delta\hat{\boldsymbol{u}}^*(\tau_k) \right)
\end{aligned} \tag{25}$$

Solving Eq. (25) results in

$$\delta\hat{\boldsymbol{u}}^*(\tau_k) = \hat{k}_{\boldsymbol{u}} + \hat{K}_{\boldsymbol{u}} \delta\boldsymbol{x}(\tau_k), \quad \delta\hat{\boldsymbol{v}}^*(\tau_k) = \hat{k}_{\boldsymbol{v}} + \hat{K}_{\boldsymbol{v}} \delta\boldsymbol{x}(\tau_k) \tag{26}$$

where

$$\begin{aligned}
\hat{k}_{\boldsymbol{u}} &= -\left(\hat{Q}_{\boldsymbol{uu}} - \hat{Q}_{\boldsymbol{uv}}\hat{Q}_{\boldsymbol{vv}}^{-1}\hat{Q}_{\boldsymbol{vu}}\right)^{-1}\left(\hat{Q}_{\boldsymbol{u}} - \hat{Q}_{\boldsymbol{uv}}\hat{Q}_{\boldsymbol{vv}}^{-1}\hat{Q}_{\boldsymbol{v}}\right) \\
\hat{k}_{\boldsymbol{v}} &= -\left(\hat{Q}_{\boldsymbol{vv}} - \hat{Q}_{\boldsymbol{vu}}\hat{Q}_{\boldsymbol{uu}}^{-1}\hat{Q}_{\boldsymbol{uv}}\right)^{-1}\left(\hat{Q}_{\boldsymbol{v}} - \hat{Q}_{\boldsymbol{vu}}\hat{Q}_{\boldsymbol{uu}}^{-1}\hat{Q}_{\boldsymbol{u}}\right) \\
\hat{K}_{\boldsymbol{u}} &= -\left(\hat{Q}_{\boldsymbol{uu}} - \hat{Q}_{\boldsymbol{uv}}\hat{Q}_{\boldsymbol{vv}}^{-1}\hat{Q}_{\boldsymbol{vu}}\right)^{-1}\left(Q_{\boldsymbol{ux}} - \hat{Q}_{\boldsymbol{uv}}\hat{Q}_{\boldsymbol{vv}}^{-1}\hat{Q}_{\boldsymbol{vx}}\right) \\
\hat{K}_{\boldsymbol{v}} &= -\left(\hat{Q}_{\boldsymbol{vv}} - \hat{Q}_{\boldsymbol{vu}}\hat{Q}_{\boldsymbol{uu}}^{-1}\hat{Q}_{\boldsymbol{uv}}\right)^{-1}\left(\hat{Q}_{\boldsymbol{vx}} - \hat{Q}_{\boldsymbol{vu}}\hat{Q}_{\boldsymbol{uu}}^{-1}\hat{Q}_{\boldsymbol{ux}}\right)
\end{aligned} \tag{27}$$

Replacing the feed-forward and feedback terms of the control update equation in Eq. (14) with their modifications derived in Eq. (27) gives the optimal control as

$$\begin{aligned}
\boldsymbol{u}(\tau_k) &= \boldsymbol{u}(\tau_k) + \delta\boldsymbol{u}^*(\tau_k) = \boldsymbol{u}(\tau_k) + \hat{K}_{\boldsymbol{u}}\delta\boldsymbol{x}(\tau_k) + \hat{k}_{\boldsymbol{u}} \\
\boldsymbol{v}(\tau_k) &= \boldsymbol{v}(\tau_k) + \delta\boldsymbol{v}^*(\tau_k) = \boldsymbol{v}(\tau_k) + \hat{K}_{\boldsymbol{v}}\delta\boldsymbol{x}(\tau_k) + \hat{k}_{\boldsymbol{v}}
\end{aligned} \tag{28}$$

Similar to GT-DDP, the update of the value function can be easily derived by comparing the second-order Taylor series of the value function and updated action-value function as

$$\begin{aligned}
V(\boldsymbol{x}(\tau_k)) &\approx \hat{Q}(\boldsymbol{x}(\tau_k), \boldsymbol{u}(\tau_k), \boldsymbol{v}(\tau_k)) + \hat{k}_{\boldsymbol{u}}^T\hat{Q}_{\boldsymbol{u}} + \hat{k}_{\boldsymbol{v}}^T\hat{Q}_{\boldsymbol{v}} + \frac{1}{2}(\hat{k}_{\boldsymbol{u}}^T\hat{Q}_{\boldsymbol{uu}}\hat{k}_{\boldsymbol{u}} + \hat{k}_{\boldsymbol{v}}^T\hat{Q}_{\boldsymbol{vv}}\hat{k}_{\boldsymbol{v}} + \hat{k}_{\boldsymbol{u}}^T\hat{Q}_{\boldsymbol{uv}}\hat{k}_{\boldsymbol{v}} + \hat{k}_{\boldsymbol{v}}^T\hat{Q}_{\boldsymbol{vu}}\hat{k}_{\boldsymbol{u}}) \\
V_{\boldsymbol{x}}(\boldsymbol{x}(\tau_k)) &\approx \hat{Q}_{\boldsymbol{x}} + \hat{K}_{\boldsymbol{u}}^T\hat{Q}_{\boldsymbol{u}} + \hat{K}_{\boldsymbol{v}}^T\hat{Q}_{\boldsymbol{v}} + \hat{Q}_{\boldsymbol{xu}}\hat{k}_{\boldsymbol{u}} + Q_{\boldsymbol{xv}}\hat{k}_{\boldsymbol{v}} + \hat{K}_{\boldsymbol{u}}^T\hat{Q}_{\boldsymbol{uu}}\hat{k}_{\boldsymbol{u}} + \hat{K}_{\boldsymbol{v}}^T\hat{Q}_{\boldsymbol{vv}}\hat{k}_{\boldsymbol{v}} + \hat{K}_{\boldsymbol{u}}^T\hat{Q}_{\boldsymbol{uv}}\hat{k}_{\boldsymbol{v}} + \hat{K}_{\boldsymbol{v}}^T\hat{Q}_{\boldsymbol{vu}}\hat{k}_{\boldsymbol{u}} \\
V_{\boldsymbol{xx}}(\boldsymbol{x}(\tau_k)) &\approx \hat{Q}_{\boldsymbol{xx}} + \hat{K}_{\boldsymbol{u}}^T\hat{Q}_{\boldsymbol{ux}} + \hat{K}_{\boldsymbol{v}}^T\hat{Q}_{\boldsymbol{vx}} + \hat{Q}_{\boldsymbol{xu}}\hat{K}_{\boldsymbol{u}} + \hat{Q}_{\boldsymbol{xv}}\hat{K}_{\boldsymbol{v}} + \hat{K}_{\boldsymbol{u}}^T\hat{Q}_{\boldsymbol{uu}}\hat{K}_{\boldsymbol{u}} + \hat{K}_{\boldsymbol{v}}^T\hat{Q}_{\boldsymbol{vv}}\hat{K}_{\boldsymbol{v}} + \hat{K}_{\boldsymbol{u}}^T\hat{Q}_{\boldsymbol{uv}}\hat{K}_{\boldsymbol{v}} + \hat{K}_{\boldsymbol{v}}^T\hat{Q}_{\boldsymbol{vu}}\hat{K}_{\boldsymbol{u}}
\end{aligned} \tag{29}$$

Likewise, with a given initial solution guesses of the opposite control sequence $\boldsymbol{U}$ and $\boldsymbol{V}$, GT-LGRDDP initializes the terminal time value function $V(\boldsymbol{x}(\tau_N))$ as $V(\boldsymbol{x}(\tau_N)) = \Phi(\boldsymbol{x}(\tau_N))$ as well as its derivatives, and runs the backward process and the forward pass iteratively until the terminal condition is triggered.

Remark 1 *For realistic aerial vehicles, the control input is naturally limited due to the physical constraints. To maintain the feasibility of the proposed algorithm in the case with two game controls, both of which are constrained by clamping constraints, the optimization problem in finding the control update should be modeled as a constrained minimax programming problem, i.e.,*

$$\begin{aligned}
&\min_{\delta\boldsymbol{u}_k}\max_{\delta\boldsymbol{v}_k} Q(\boldsymbol{x}_k + \delta\boldsymbol{x}_k, \boldsymbol{u}_k + \delta\boldsymbol{u}_k, \boldsymbol{v}_k + \delta\boldsymbol{v}_k) \\
s.t. \quad &\boldsymbol{u}_{min} \le \boldsymbol{u}_k + \delta\boldsymbol{u}_k \le \boldsymbol{u}_{max} \\
&\boldsymbol{v}_{min} \le \boldsymbol{v}_k + \delta\boldsymbol{v}_k \le \boldsymbol{v}_{max}
\end{aligned} \tag{30}$$

which needs to be solved by developing appropriate min-max optimization solvers. In order to make the algorithm

simple to implement and fast to compute, a simple truncation is used in this Note to approximately solve the problem, i.e.,

$$\begin{aligned}\boldsymbol{u}(\tau_k) &= \min(\max(\boldsymbol{u}(\tau_k) + \delta\boldsymbol{u}^*(\tau_k), \boldsymbol{u}_{\min}), \boldsymbol{u}_{\max}) \\ \boldsymbol{v}(\tau_k) &= \min(\max(\boldsymbol{v}(\tau_k) + \delta\boldsymbol{v}^*(\tau_k), \boldsymbol{v}_{\min}), \boldsymbol{v}_{\max})\end{aligned} \tag{31}$$

where $\boldsymbol{u}_{\min}$, $\boldsymbol{v}_{\min}$ represent the lower bounds and $\boldsymbol{u}_{\max}$, $\boldsymbol{v}_{\max}$ stand for the upper bounds.

B. GT-LGRDDP with Flexible Final Time

As stated before, it is intractable to set an optimal final time beforehand since the future trajectory of the system might be unpredictable. For this reason, this section presents a GT-LGRDDP algorithm in a flexible-final-time framework, termed as GT-FFT-LGRDDP, with rigorous mathematical derivation and introduces how the algorithm dynamically optimizes the final time in a recursive way. Notice that the final time only influences the value function at the endpoint. This means that optimizing the control input and final time can be decoupled with each other and hence we can find the optimal final time increment in the unscaled time framework $t \in \left[t_0, t_f\right]$. More specifically, the proposed GT-FFT-LGDDP algorithm updates the control history and the final time by determining their increments with two subsequent steps. First, with a given final time t_N, GT-LGRDDP updates the control history following the details shown in Sec. III.A. Second, the updated final time is computed after updating the control history and the resultant system trajectory. It is worthy pointing out that another benefit of optimizing the terminal time is that we can dynamically minimize flight time for aerial vehicles in some time-critical missions, except for removing the heuristic setting of the final time. This means that the terminal cost $\Phi(\boldsymbol{x}_N, t_N)$ in a flexible-final-time framework can be rewritten as $\Phi(\boldsymbol{x}_N, t_N) = \phi(\boldsymbol{x}_N) + k_N t_N$, where $\phi(\boldsymbol{x}_N)$ denotes the terminal state constraint and $k_N t_N$ with $k_N > 0$ is related to the penalty of the flight time. With this in mind, the objective function in Eq. (2) can be discretized as

$$J(\boldsymbol{u}, \boldsymbol{v}) = \phi(\boldsymbol{x}_N) + k_N t_N + \sum_{k=0}^{N-1} \hat{L}(\boldsymbol{x}(\tau_k), \boldsymbol{u}(\tau_k), \boldsymbol{v}(\tau_k)) \tag{32}$$

Considering the relationship between the value function and cost function at terminal time instant, we have $V(\boldsymbol{x}_N, t_N) = \phi(\boldsymbol{x}_N) + k_N t_N$. This means that the variation of the terminal value function includes both the variation of the terminal state $\boldsymbol{x}_N$ and the variation of the terminal time t_N. Similar to the control input update, a second-order Taylor series is utilized to approximate the terminal value function around nominal trajectory $(\boldsymbol{x}_N, t_N)$ as

$$\begin{aligned}V(\boldsymbol{x}_N + \delta\boldsymbol{x}_N, t_N + \delta t_N) \approx & V(\boldsymbol{x}_N, t_N) + V_{\boldsymbol{x}_N}\delta\boldsymbol{x}_N + V_{t_N}\delta t_N + \frac{1}{2}\delta\boldsymbol{x}_N^T V_{\boldsymbol{x}_N\boldsymbol{x}_N}\delta\boldsymbol{x}_N + \frac{1}{2}V_{t_N t_N}\delta t_N^2 \\ & + \frac{1}{2}\delta\boldsymbol{x}_N^T V_{\boldsymbol{x}_N t_N}\delta t_N + \frac{1}{2}\delta t_N V_{t_N\boldsymbol{x}_N}\delta\boldsymbol{x}_N\end{aligned} \tag{33}$$

Notice that the change of t_N also poses an effect on the terminal state $\boldsymbol{x}_N$ since the final time t_N is free. Consequently,

the chain rule of calculus is leveraged to derive the gradients and Hessian of the terminal value function, i.e.,

$$V_{t_N} = \frac{\partial V}{\partial t_N} = \frac{\partial \phi}{\partial \boldsymbol{x}_N}\frac{\partial \boldsymbol{x}_N}{\partial t_N} + \frac{\partial (k_N t_N)}{\partial t_N} = \phi_{\boldsymbol{x}_N}^T \boldsymbol{F}(\boldsymbol{x}_N, \boldsymbol{u}_N, \boldsymbol{v}_N) + k_N \tag{34}$$

$$V_{\boldsymbol{x}_N t_N} = V_{t_N \boldsymbol{x}_N}^T = \frac{\partial}{\partial \boldsymbol{x}_N}\left(\frac{\partial V}{\partial t_N}\right) = \phi_{\boldsymbol{x}_N \boldsymbol{x}_N} \boldsymbol{F}(\boldsymbol{x}_N, \boldsymbol{u}_N, \boldsymbol{v}_N) + \boldsymbol{F}_{\boldsymbol{x}_N}^T \phi_{\boldsymbol{x}_N} \tag{35}$$

$$\begin{aligned} V_{t_N t_N} =& \frac{\partial}{\partial t_N}\left(\frac{\partial V}{\partial t_N}\right) = \frac{\partial}{\partial t_N}\left(\frac{\partial \phi}{\partial \boldsymbol{x}_N}\frac{\partial \boldsymbol{x}_N}{\partial t_N} + \frac{\partial (k_N t_N)}{\partial t_N}\right) \\ =& \frac{\partial}{\partial \boldsymbol{x}_N}\left(\frac{\partial \phi}{\partial t_N}\right)\frac{\partial \boldsymbol{x}_N}{\partial t_N} + \frac{\partial \phi}{\partial \boldsymbol{x}_N}\frac{\partial}{\partial t_N}\boldsymbol{F}(\boldsymbol{x}_N, \boldsymbol{u}_N, \boldsymbol{v}_N) \end{aligned} \tag{36}$$

where $\boldsymbol{F}_{\boldsymbol{x}_N}$ is the gradient of the system dynamics at time instant t_N.

It should be noted that updating $\boldsymbol{u}_N$ and $\boldsymbol{v}_N$ are not explicitly given by GT-LGRDDP in the iteration but this information is useful in the calculation of the increment of the final time. Thus, we assume $\boldsymbol{u}_N = \boldsymbol{u}_{N-1}$ and $\boldsymbol{v}_N = \boldsymbol{v}_{N-1}$ during the implementation and this directly leads to $\partial \boldsymbol{F}(\boldsymbol{x}_N, \boldsymbol{u}_N, \boldsymbol{v}_N)/\partial \boldsymbol{u}_N = 0$ and $\partial \boldsymbol{F}(\boldsymbol{x}_N, \boldsymbol{u}_N, \boldsymbol{v}_N)/\partial \boldsymbol{v}_N = 0$. Hence, we can readily derive

$$\frac{\partial}{\partial t_N}\boldsymbol{F}(\boldsymbol{x}_N, \boldsymbol{u}_N, \boldsymbol{v}_N) = \frac{\partial}{\partial \boldsymbol{x}_N}\boldsymbol{F}(\boldsymbol{x}_N, \boldsymbol{u}_N, \boldsymbol{v}_N)\frac{\partial \boldsymbol{x}_N}{\partial t_N} \tag{37}$$

Substituting Eq. (37) into Eq. (36) gives

$$\begin{aligned} V_{t_N t_N} =& \frac{\partial}{\partial \boldsymbol{x}_N}\left(\frac{\partial \phi}{\partial \boldsymbol{x}_N}\frac{\partial \boldsymbol{x}_N}{\partial t_N}\right)\frac{\partial \boldsymbol{x}_N}{\partial t_N} + \frac{\partial \phi}{\partial \boldsymbol{x}_N}\frac{\partial}{\partial \boldsymbol{x}_N}\boldsymbol{F}(\boldsymbol{x}_N, \boldsymbol{u}_N, \boldsymbol{v}_N)\frac{\partial \boldsymbol{x}_N}{\partial t_N} \\ =& \left[\frac{\partial}{\partial \boldsymbol{x}_N}\left(\frac{\partial \phi}{\partial \boldsymbol{x}_N}\right)\frac{\partial \boldsymbol{x}_N}{\partial t_N} + \frac{\partial \phi}{\partial \boldsymbol{x}_N}\frac{\partial}{\partial \boldsymbol{x}_N}\left(\frac{\partial \boldsymbol{x}_N}{\partial t_N}\right)\right]^T \frac{\partial \boldsymbol{x}_N}{\partial t_N} + \frac{\partial \phi}{\partial \boldsymbol{x}_N}\frac{\partial}{\partial \boldsymbol{x}_N}\boldsymbol{F}(\boldsymbol{x}_N, \boldsymbol{u}_N, \boldsymbol{v}_N)\frac{\partial \boldsymbol{x}_N}{\partial t_N} \\ =& \left[\phi_{\boldsymbol{x}_N \boldsymbol{x}_N}\boldsymbol{F}(\boldsymbol{x}_N, \boldsymbol{u}_N, \boldsymbol{v}_N) + \phi_{\boldsymbol{x}_N}\boldsymbol{F}_{\boldsymbol{x}_N} + \phi_{\boldsymbol{x}_N}\boldsymbol{F}_{\boldsymbol{x}_N}\right]^T \boldsymbol{F}(\boldsymbol{x}_N, \boldsymbol{u}_N, \boldsymbol{v}_N) \end{aligned} \tag{38}$$

By the definition of the value function, we have

$$\begin{aligned} V(\boldsymbol{x}_N, t_N) &= \Phi(\boldsymbol{x}_N, t_N) = \phi(\boldsymbol{x}_N) + k_N t_N \\ V_{\boldsymbol{x}_N}(\boldsymbol{x}_N, t_N) &= \Phi_{\boldsymbol{x}_N}(\boldsymbol{x}_N, t_N) = \phi_{\boldsymbol{x}_N}(\boldsymbol{x}_N) \\ V_{\boldsymbol{x}_N \boldsymbol{x}_N}(\boldsymbol{x}_N, t_N) &= \Phi_{\boldsymbol{x}_N \boldsymbol{x}_N}(\boldsymbol{x}_N, t_N) = \phi_{\boldsymbol{x}_N \boldsymbol{x}_N}(\boldsymbol{x}_N) \end{aligned} \tag{39}$$

Notice that the value function at terminal time, expressed in Eq. (33), is a quadratic function with respect to terminal state variation $\delta \boldsymbol{x}_N$ and terminal time variation δt_N. To this end, the optimal terminal time correction δt_N^* can be derived by using the first-order optimality condition of Eq. (33) as

$$\delta t_N^* = -V_{t_N t_N}^{-1}\left(V_{t_N} + V_{t_N \boldsymbol{x}_N}\delta \boldsymbol{x}_N\right) \tag{40}$$

which indicates that the optimal final time correction δt_N^* is composed of a feed-forward term $-V_{t_N t_N}^{-1} V_{t_N}$ and a state feedback term $-V_{t_N t_N}^{-1} V_{t_N \boldsymbol{x}_N} \delta \boldsymbol{x}_N$. Notice that V_{t_N}, $V_{t_N \boldsymbol{x}_N}$ and $V_{t_N t_N}$ are determined by using Eqs. (34), (35) and (38).

Substituting Eq. (40) back into Eq. (33), the update of the terminal value function can be then expressed as

$$\begin{aligned} V\left(\boldsymbol{x}_N + \delta \boldsymbol{x}_N, t_N + \delta t_N\right) \approx & V\left(\boldsymbol{x}_N, t_N\right) - \frac{1}{2} V_{t_N} V_{t_N t_N}^{-1} V_{t_N} + \left[V_{\boldsymbol{x}_N} - V_{t_N \boldsymbol{x}_N}^T V_{t_N t_N}^{-1} V_{t_N}\right]^T \delta \boldsymbol{x}_N \\ & + \frac{1}{2} \delta \boldsymbol{x}_N^T \left[V_{\boldsymbol{x}_N \boldsymbol{x}_N} - V_{t_N \boldsymbol{x}_N}^T V_{t_N t_N}^{-1} V_{t_N \boldsymbol{x}_N}\right] \delta \boldsymbol{x}_N \end{aligned} \tag{41}$$

which subsequently gives the derivatives of the value function at time instant t_N as

$$\begin{aligned} V_{\boldsymbol{x}}\left(\boldsymbol{x}_N\right) &= V_{\boldsymbol{x}_N}^T - V_{t_N \boldsymbol{x}_N}^T V_{t_N t_N}^{-1} V_{t_N} \\ V_{\boldsymbol{x}\boldsymbol{x}}\left(\boldsymbol{x}_N\right) &= V_{\boldsymbol{x}_N \boldsymbol{x}_N} - V_{t_N \boldsymbol{x}_N}^T V_{t_N t_N}^{-1} V_{t_N \boldsymbol{x}_N} \end{aligned} \tag{42}$$

Then, substituting Eq. (42) into Eq. (23) gives the gradients and Hessian matrices of the action-value function at the $(N-1)$-th discretization points as

$$\begin{aligned} \hat{Q}_{\boldsymbol{x},N-1} =& \hat{L}_{\boldsymbol{x},N-1} + \hat{A}_{N-1}^T \hat{V}_{\boldsymbol{x}}(\boldsymbol{x}(\tau_N)), \quad \hat{Q}_{\boldsymbol{u},N-1} = \hat{L}_{\boldsymbol{u},N-1} + \hat{B}_{\boldsymbol{u}_{N-1}}^T \hat{V}_{\boldsymbol{x}}(\boldsymbol{x}(\tau_N)) \\ \hat{Q}_{\boldsymbol{v},N-1} =& \hat{L}_{\boldsymbol{v},N-1} + \hat{B}_{\boldsymbol{v}_{N-1}}^T \hat{V}_{\boldsymbol{x}}(\boldsymbol{x}(\tau_N)), \quad \hat{Q}_{\boldsymbol{x}\boldsymbol{x},N-1} = \hat{L}_{\boldsymbol{x}\boldsymbol{x},N-1} + \hat{A}_{N-1}^T \hat{V}_{\boldsymbol{x}\boldsymbol{x}}(\boldsymbol{x}(\tau_N)) \hat{A}_{N-1} \\ \hat{Q}_{\boldsymbol{u}\boldsymbol{u},N-1} =& \hat{L}_{\boldsymbol{u}\boldsymbol{u},N-1} + \hat{B}_{\boldsymbol{u}_{N-1}}^T \hat{V}_{\boldsymbol{x}\boldsymbol{x}}(\boldsymbol{x}(\tau_N)) \hat{B}_{\boldsymbol{u}_{N-1}}, \quad \hat{Q}_{\boldsymbol{v}\boldsymbol{v},N-1} = \hat{L}_{\boldsymbol{v}\boldsymbol{v},N-1} + \hat{B}_{\boldsymbol{v}_{N-1}}^T \hat{V}_{\boldsymbol{x}\boldsymbol{x}}(\boldsymbol{x}(\tau_N)) \hat{B}_{\boldsymbol{v}_{N-1}} \\ \hat{Q}_{\boldsymbol{x}\boldsymbol{u},N-1} =& \hat{L}_{\boldsymbol{x}\boldsymbol{u},N-1} + \hat{A}_{N-1}^T \hat{V}_{\boldsymbol{x}\boldsymbol{x}}(\boldsymbol{x}(\tau_N)) \hat{B}_{\boldsymbol{u}_{N-1}}, \quad \hat{Q}_{\boldsymbol{x}\boldsymbol{v},N-1} = \hat{L}_{\boldsymbol{x}\boldsymbol{v},N-1} + \hat{A}_{N-1}^T \hat{V}_{\boldsymbol{x}\boldsymbol{x}}(\boldsymbol{x}(\tau_N)) \hat{B}_{\boldsymbol{v}_{N-1}} \\ \hat{Q}_{\boldsymbol{u}\boldsymbol{v},N-1} =& \hat{L}_{\boldsymbol{u}\boldsymbol{v},N-1} + \hat{B}_{\boldsymbol{u}_{N-1}}^T \hat{V}_{\boldsymbol{x}\boldsymbol{x}}(\boldsymbol{x}(\tau_N)) \hat{B}_{\boldsymbol{v}_{N-1}}, \quad \hat{Q}_{\boldsymbol{v}\boldsymbol{u},N-1} = \hat{Q}_{\boldsymbol{u}\boldsymbol{v},N-1}^T, \ \hat{Q}_{\boldsymbol{v}\boldsymbol{x},N-1} = \hat{Q}_{\boldsymbol{x}\boldsymbol{v},N-1}^T, \ \hat{Q}_{\boldsymbol{u}\boldsymbol{x},N-1} = \hat{Q}_{\boldsymbol{x}\boldsymbol{u},N-1}^T \end{aligned} \tag{43}$$

where the terms with subscripts $N-1$ and N denote the values of the corresponding terms evaluated at the $(N-1)$-th and N-th discretization points, respectively.

Since the update of the final time only affects the action-value function at the $(N-1)$-th discretization point, the gradients and Hessian matrices of the action-value function at other discretization points are the same as Eq. (23).

Remark 2 *It is known that a poor initial solution guess may result in the divergence of a numerical solver and failure of convergence to a local optimum. Therefore, the update of the final time is restricted to a bounded region as*

$$t_N = \min(\max(t_N + \delta t_N^*, t_{\min}), t_{\max}) \tag{44}$$

where $t_{\min}$ and $t_{\max}$ represent the lower and the upper bounds of the final time, respectively.

Remark 3 *Note that the proposed algorithm revisits the GT-DDP using the following two-step to update the control*

history and the final time: (1) Given a t_N , we can run the classical GT-DDP to find the updated control history; and (2) Using the updated control history, we compute the update of the final time. As a result, the preceding iterative process does not need to add the weighted final time to the terminal cost to cater for the flexible-final-time framework. Hence, the proposed algorithm is capable of solving a general free-final-time optimization problem without adding a weighted final time to the terminal cost, i.e., $k_N = 0$.

C. Summary of the Proposed Algorithm

By summarizing the results of the previous two subsections, we summarize the pseudocode of the proposed GT-FFT-LGRDDP in Algorithm 1, which clearly reveals that the optimal control correction is first found before the final time is optimized. Notice that the final time t_N is varying during the iteration and hence we re-scale the time duration using the LGR nodes at every iteration with a fixed number of discretization points. This helps to maintain the efficiency of the proposed GT-FFT-LGRDDP algorithm.

Algorithm 1 Game-Theoretic Flexible-Final-Time Legendre-Gauss-Radau Differential Dynamic Programming

Initial state $\boldsymbol{x}_0$, initialized control sequence $\boldsymbol{U}$ and $\boldsymbol{V}$, initialized terminal time t_N, number of discretization points N, termination condition threshold ϵ
Generate $n = N - 1$ LGR nodes
Scale the time horizon using Eq. (15)
for $k = 0, ..., N - 1$ **do**
  $\boldsymbol{x}(\tau_{k+1}) = \boldsymbol{x}(\tau_k) + \frac{t_f - t_0}{2} w_k \boldsymbol{F}(\boldsymbol{x}(\tau_k), \boldsymbol{u}(\tau_k), \boldsymbol{x}(\tau_k))$
end for
Initialize the cost function J_0
while $|J_i - J_{i-1}| \geq \epsilon$ **do**
  $V_N = \phi(\boldsymbol{x}_N) + t_N$, $V_{\boldsymbol{x}_N} = \phi_{\boldsymbol{x}_N}(\boldsymbol{x}_N)$, $V_{\boldsymbol{x}_N \boldsymbol{x}_N} = \phi_{\boldsymbol{x}_N \boldsymbol{x}_N}(\boldsymbol{x}_N)$
  $V_{t_N} \leftarrow$ Eq. (34), $V_{t_N \boldsymbol{x}_N} \leftarrow$ Eq. (35), $V_{t_N t_N} \leftarrow$ Eq. (38)
  for $k = N - 1, ..., 0$ **do**
    if $k = N - 1$ **then**
      $V_{\boldsymbol{x}}(\boldsymbol{x}_N)$, $V_{\boldsymbol{x}\boldsymbol{x}}(\boldsymbol{x}_N) \leftarrow$ Eq. (42)
      Calculate the gradients and Hessian matrices of $\hat{Q}(\boldsymbol{x}_{N-1}, \boldsymbol{u}_{N-1})$ using Eq. (43)
    else
      Calculate the gradients and Hessian matrices of $\hat{Q}(\boldsymbol{x}_k, \boldsymbol{u}_k)$ using Eq. (23)
    end if
    $\boldsymbol{u}_k$,$\boldsymbol{v}_k \leftarrow$ Eq. (28)
    $\boldsymbol{u}_k = \min(\max(\boldsymbol{u}_k + \delta \boldsymbol{u}_k, \boldsymbol{u}_{\min}), \boldsymbol{u}_{\max})$
    $\boldsymbol{v}_k = \min(\max(\boldsymbol{v}_k + \delta \boldsymbol{v}_k, \boldsymbol{v}_{\min}), \boldsymbol{v}_{\max})$
    Update of the modified value function and its derivatives using Eq. (13)
  end for
  for $k = 0, ..., N - 1$ **do**
    $\boldsymbol{x}(\tau_{k+1}) = \boldsymbol{x}(\tau_k) + \frac{t_f - t_0}{2} w_k \boldsymbol{F}(\boldsymbol{x}(\tau_k), \boldsymbol{u}(\tau_k), \boldsymbol{x}(\tau_k))$
  end for
  $\delta t_N^* \leftarrow$ Eq. (40)
  $t_N = \min(\max(t_N + \delta t_N^*, t_{\min}), t_{\max})$
  Scale the time horizon using Eq. (15)
end while

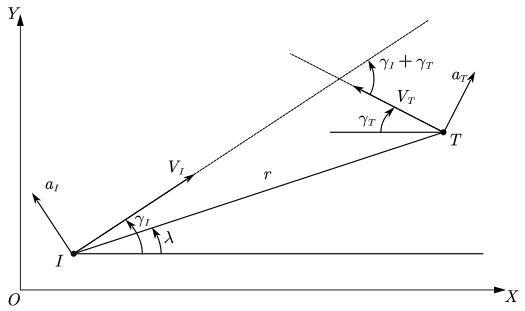


Fig. 1 Planar Engagement geometry.

IV. Application Example: Impact-Angle-Constrained Pursuer-Evader Differential Game

In this section, we consider a pursuer-evader game between an interceptor I and a target T as an example to apply the proposed algorithm. We first introduce the dynamic model of the missile-target system and formulate a receding horizon differential game problem with multiple terminal constraints. The proposed GT-FFT-LGRDDP algorithm is then leveraged to solve the optimization problem formulated.

A. Kinematics and Dynamics Models

The engagement geometry in the vertical plane of two players for the game is shown in Fig. 1, where $X - O - Y$ is a Cartesian inertial reference frame. The symbols V_I and V_T represent the velocity of I and T. The flight-path angles of these two vehicles are denoted by γ_I and γ_T. The notations a_I and a_T are the lateral lift accelerations perpendicular to the velocity vectors. The relative distance between I and T is r and the line-of-sight (LOS) angle is represented by λ. To verify the effectiveness of the proposed algorithm for real interception scenarios, realistic interceptor and target models with thrust, drag and gravity are considered. Thus, the differential equations that govern this he vehicle kinematics are given by

$$\begin{aligned}
\dot{x}_i &= V_i \cos\gamma_i \\
\dot{y}_i &= V_i \sin\gamma_i \\
\dot{V}_i &= \frac{T_i - D_i}{m_i} - g\sin\gamma_i \\
\dot{\gamma}_i &= \frac{a_i - g\cos\gamma_i}{V_i}, \quad i = I, T
\end{aligned} \tag{45}$$

where g is the expression of gravitational acceleration, T_i is the engine thrust and m_i denotes the mass of the vehicle. The aerodynamic drag of the vehicles, denoted by D_i, is modeled as

$$D_i = C_{D_i} q_i S_i + \frac{K_i^2 a_i^2 m_i^2}{q_i S_i}, \quad i = I, T \tag{46}$$

where C_{D_i} represents the zero-lift drag coefficient and K_i is the induced drag coefficient. The symbol S_i is the reference area, and $q_i = 0.5\rho V_i^2$ is the aerodynamic pressure with ρ being the atmosphere density.

B. Optimization Problem Formulation

To ensure that the missile can precisely intercept the target with desired impact angle, it is desirable to impose constraints on minimizing terminal miss distance along with the terminal impact angle error. The terminal distance between the interceptor and the target $r(t_f)$, known as terminal miss distance, is determined by the Euclidean distance as $r^2(t_f) = \left[x_T(t_f) - x_I(t_f)\right]^2 + \left[y_T(t_f) - y_I(t_f)\right]^2$, where $x_I(t_f)$, $y_I(t_f)$ and $x_T(t_f)$, $y_T(t_f)$ represent the terminal inertial positions of the interceptor and the target, respectively. The impact angle is defined as the angle between the velocity vectors between the interceptor and the target. From Fig. 1, the impact angle can be mathematically expressed as $\eta = \gamma_I + \gamma_T$. Let the desired impact angle be η_d and the impact angle error is then given by $e_\eta = \gamma_I + \gamma_T - \eta_d$. For real interception scenarios, the missile is desirable to intercept the target in a short time and hence we also penalize the flight time in the terminal constraint. With the preceding considerations, the terminal constraint function $\Phi(\boldsymbol{x}(t_f), t_f)$ for our guidance problem is defined as

$$\Phi(\boldsymbol{x}(t_f), t_f) = \frac{1}{2}\omega_{N_r} r^2(t_f) + \frac{1}{2}\omega_{N_\eta} e_\eta^2(t_f) + k_N t_f \tag{47}$$

where $\omega_{N_r} > 0$, $\omega_{N_\eta} > 0$ and $k_N > 0$ are the weighting factors that enforce the soft terminal constraints on the terminal miss distance, terminal impact angle error and flight time, respectively. Notice that these weighting factors should be tuned to consider the scale difference across all state variables. Define $\boldsymbol{x} = [x_I, y_I, x_T, y_T, V_I, V_T, \gamma_I, \gamma_T]^T$ as the system state vector and the desired terminal state $\boldsymbol{x}_d = [0, 0, 0, 0, 0, 0, \eta_d/2, \eta_d/2]^T$, Eq. (47) can be rewritten in a more compact form as

$$\Phi(\boldsymbol{x}(t_f), t_f) = \frac{1}{2}\left[\boldsymbol{x}(t_f) - \boldsymbol{x}_d\right]^T \mathbf{W}_N \left[\boldsymbol{x}(t_f) - \boldsymbol{x}_d\right] + k_N t_f \tag{48}$$

where $\mathbf{W}_N$ is a fixed weighting matrix and can be easily obtained from Eq. (47).

Since minimizing the quadratic energy consumption term is helpful to reduce the induced drag and is a worthy goal in endo-atmospheric trajectory optimization, we consider energy consumption of both the interceptor and the target to formulate the running cost $\mathcal{L}(\boldsymbol{x}(t), a_I(t), a_T(t))$ as

$$\mathcal{L}(\boldsymbol{x}(t), a_I(t), a_T(t)) = \frac{1}{2}\left[\frac{R_I}{t_{\text{go}}} a_I^2(t) - \gamma^2 R_T a_T^2(t)\right] \tag{49}$$

where two players are competing with each other in the running cost with positives $R_I/t_{\text{go}} > 0$ and $R_T > 0$ energy consumption weighting factors. The symbol $t_{go} = t_f - t$ denotes the remaining flight time, which penalizes more control energy at the initial period of the homing phase and gradually reduce the penalty when approaching the target.

This helps to improve the operational margins of the interceptor and hence can improve the impact performance in terms of the terminal interception errors. Notice that The positive constant $\gamma > 0$ measures the interceptor's maneuverability relative to the target and setting a big value is helpful to locally attenuate the effect of unknown target maneuver a_T [13].

Combing Eqs. (48) and (49), the objective function can be explicitly presented as

$$J\left(a_I, a_T\right) = \frac{1}{2}\left[\boldsymbol{x}(t_f) - \boldsymbol{x}_d\right]^T \mathbf{W}_N \left[\boldsymbol{x}(t_f) - \boldsymbol{x}_d\right] + k_N t_f + \int_{t_0}^{t_f} \frac{1}{2}\left[\frac{R_I}{t_{\text{go}}} a_I^2(t) - \gamma^2 R_T a_T^2(t)\right] \mathrm{d}t \tag{50}$$

Then, the differential game optimization problem can be formulated as: find an optimal guidance command profile $a_I^*(t)$ that minimizes objective function (50) subject to system dynamics (45) with a given initial condition $\boldsymbol{x}_0$.

C. Closed-Loop Guidance Command Generation

Notice that minimizing objective function (50) generates an open-loop command and hence is sensitive to external disturbances, e.g., the missile will miss the target once the target performs a sudden maneuver. To enable the generation of a closed-loop guidance command, we reformulate the optimization problem in a receding horizon manner, i.e., solving a differential game optimization problem at every time instant. To this end, we revise objective function (50) as

$$J_k = \frac{1}{2}\left[\boldsymbol{x}(t_f) - \boldsymbol{x}_d\right]^T \mathbf{W}_N \left[\boldsymbol{x}(t_f) - \boldsymbol{x}_d\right] + k_N t_f + \int_{t_k}^{t_f} \frac{1}{2}\left[\frac{R_I}{t_{\text{go}}} a_I^2(t) - \gamma^2 R_T a_T^2(t)\right] \mathrm{d}t \tag{51}$$

The optimal guidance command can be obtained by sequentially minimizing objective function (51) from the initial time t_0 to the final time t_f: at time instant t_k, we apply the proposed GT-FFT-LGRDDP algorithm to find an optimal control sequence $\{a_{I_k}^*, a_{I_{k+1}}^*, \cdots, a_{I_{N+k}}^*\}$, which contains N elements corresponding to N nodes that required by the algorithm. The first element of $\boldsymbol{U}_k^*$, i.e., $a_{I_k}^*$, is applied to the missile as the guidance command at time instant t_k. This means that the guidance command is solved in a receding horizon optimization manner with shrinking time horizon.

V. Numerical Simulations

The performance of the proposed algorithm derived in Sec. III is evaluated in the following subsection through numerical simulations. The initial conditions of the considered scenario and vehicles aerodynamics are summarized in Table 1. The atmospheric density model is $\rho(y) = 1.15579 - 1.058\times10^{-4}y + 3.725\times10^{-9}y^2 - 6.0\times10^{-14}y^3$ kg/m^3, $y \in [0\,\text{m}, 20\,000\,\text{m}]$. The constant penalty weightings in cost function (51) are chosen as $k_N = 1$, $R_I = 1$, $R_T = 1$, $\gamma = 18$. We use $N = 39$ LGR nodes to discretize the dynamic system and an unguided trajectory, i.e., with zero guidance command, and $t_N = 4.2s$ to initialize GT-FFT-LGRDDP algorithm in all simulations. Considering the initial condition of the scenario, the terminal time is bounded by $t_N \in [0.01s,\ 8s]$. The termination threshold is tuned as $\epsilon = 10^{-6}$ for each simulation run. Notice that the interceptor with additional impact angle constraint requires bigger lateral acceleration than the homing-only guidance. Hence, we limit the acceleration as $|a_I| \leq 400m/s^2$, which shares similar

level of maneuver demand as [13, 25, 26]. Since the aerodynamics of a hostile target is not available to the interceptor, we assume that the target is moving only with the influence by the lift acceleration and the gravity in running the optimization algorithm while the full motion model is considered in determining the trajectory of the target.

Table 1 Initial conditions and aerodynamics.

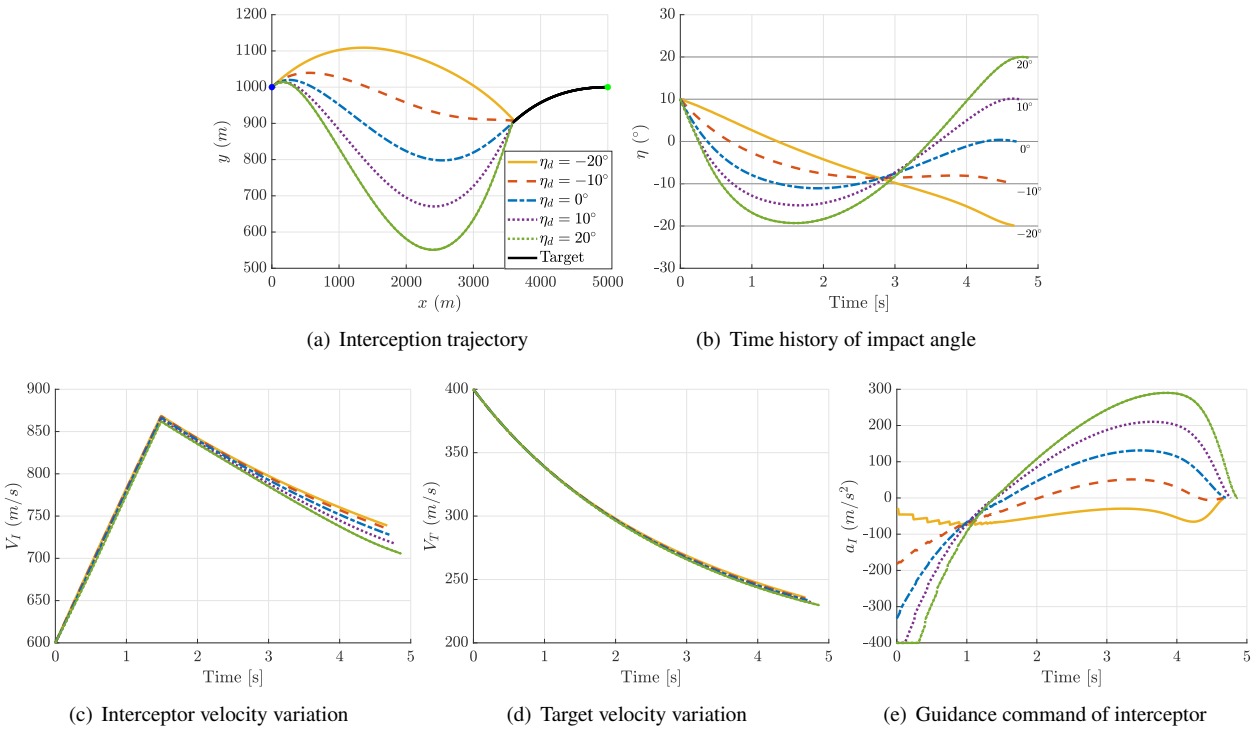
Parameters of Interceptor	Value	Parameters of Target	Value
Initial position $(x_I,\ y_I)$	$(0m,\ 1000m)$	Initial position $(x_T,\ y_T)$	$(5000m,\ 1000m)$
Initial flight path angle γ_I	10°	Initial flight path angle γ_T	0°
Initial speed V_I	$600m/s$	Initial speed V_T	$400m/s$
Zero-lift drag coefficient C_{D_I}	0.035	Zero-lift drag coefficient C_{D_T}	0.35
Induced drag coefficient K_{D_I}	0.32	Induced drag coefficient K_{D_T}	0.04
Reference area S_I	$1m^2$	Reference area S_T	$27.87m^2$
Thrust $T_I\ (kN)$	$\begin{cases} 35, & t \in [0, 1.5s] \\ 0, & t > 1.5s \end{cases}$	Thrust T_T	$131.22kN$
Mass $m_I\ (kg)$	$\begin{cases} 135 - 30t, & t \in [0, 1.5s] \\ 90, & t > 1.5s \end{cases}$	Mass m_T	$9207kg$

A. Characteristics of Proposed Algorithm

We first consider a gravity-compensated step maneuver target with $a_T = -10m/s^2 + g\cos\gamma_T$ to analyze the characteristics of the proposed GT-FFT-LGRDDP algorithm. The interceptor is supposed to intercept the target with various impact angles $\eta_d = -20^\circ, -10^\circ, 0^\circ, 10^\circ, 20^\circ$, which includes typical engagement scenarios like head-on and side attack. The tuning parameters ω_{N_r} is set as 400 for five cases and ω_{N_η} is 8×10^7 for all in this simulation. The simulation results, including the engagement trajectory, time history of impact angle, velocity of the interceptor and the target, and guidance command of the interceptor, are presented in Fig. 2. The results indicate that the proposed GT-FFT-LGRDDP algorithm with a receding horizon mechanism provides the capability to find the optimal guidance command that enables accurate target interception with desired impact angle constraint. The recorded miss distance and impact angle error in the simulations are less than $3m$ and 0.1°, respectively. We can also observe that the interceptor in the side attack scenario generates more curved flight trajectory than the head-on case, thus requiring bigger lateral acceleration. From Fig. 2(e), we can find the curve of the guidance command of the interceptor exhibits jagged changes. This can be attributed to the fact that the optimization horizon is shrinking while the number of discrete nodes is fixed. To eliminate or mitigate the jagged oscillation, we can increase the number of discrete nodes. This, however, will increase the computational burden.

B. Comparison with FFT-LGRDDP and GT-FFT-DDP

In this subsection, we compare the proposed algorithm with Flexible-Final-Time Legendre-Gauss-Radau DDP (FFT-LGRDDP) algorithm and Game-Theoretic Flexible-Final-Time DDP (GT-FFT-DDP) algorithm to demonstrate the



(a) Interception trajectory (b) Time history of impact angle

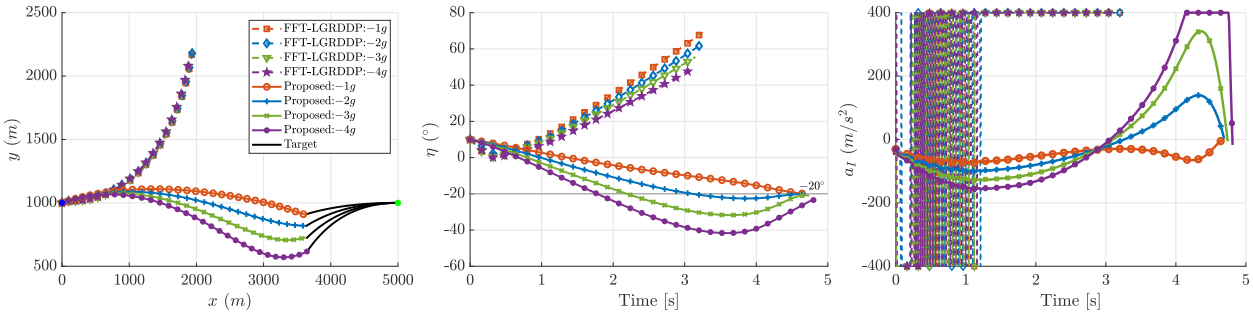
(c) Interceptor velocity variation (d) Target velocity variation (e) Guidance command of interceptor

Fig. 2 Engagement simulation of the proposed algorithm with step maneuver target.

advantages of the proposed algorithm. The FFT-LGRDDP algorithm replaces the Euler discretization in the FFT-DDP [18] and the GT-FFT-DDP algorithm modifies the original GT-DDP [16] by additionally optimizing the final time. All test algorithms are running in a shrinking receding horizon manner to generate a closed-loop guidance command and the tuning parameters ω_{N_r} and ω_{N_η} are set as 225 and 8×10^7 in all simulations for fair comparison.

We first compare the performance of the proposed algorithm with FFT-LGRDDP with different target maneuvers $a_T = 1g + g\cos\gamma_T, 2g + g\cos\gamma_T, 3g + g\cos\gamma_T, 4g + g\cos\gamma_T$. Since the future target maneuver is generally uncertain to the interceptor, we implement the FFT-LGRDDP with $a_T = g\cos\gamma_T$. The comparison results, including the engagement trajectory, time history of impact angle, and guidance command of the interceptor, with different target maneuvers under two methods, are presented in Fig. 3. The results indicate that the proposed algorithm can guide the missile to intercept the target with desired terminal conditions, while the FFT-LGRDDP algorithm becomes unstable. Therefore, it can be concluded that for unknown target maneuvers up to $4g$, the proposed algorithm is able to provide satisfactory results, while FFT-LGRDDP fails. This indicates that the proposed algorithm provides improved performance in dealing with the uncertain maneuver target.

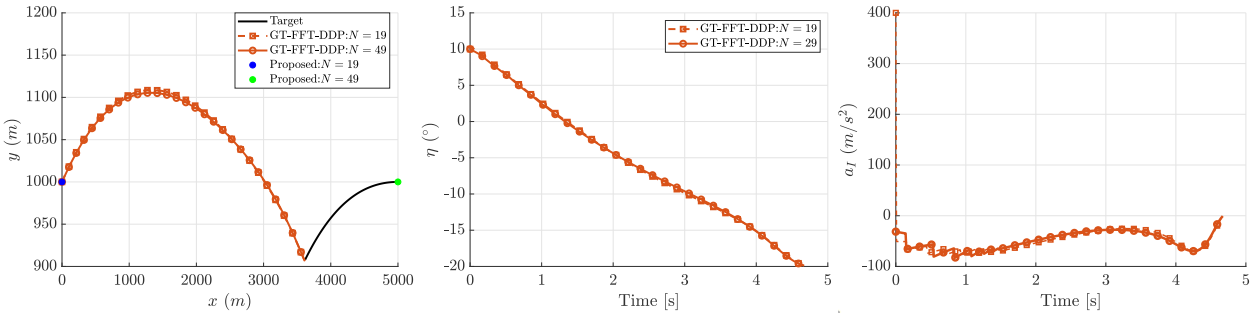
Now, let us evaluate the advantage of leveraging LGR discretization in DDP by comparing with GT-FFT-DDP, which utilizes the Euler discretization scheme. The target is supposed to execute a step maneuver with the magnitude of $1g$. To better understand the sensitivity of the algorithm to N, we perform comparative simulations with several different



(a) Interception trajectory (b) Time history of impact angle (c) Guidance command of interceptor

Fig. 3 Comparison simulation results against step maneuver target.

numbers of nodes $N = 19, 29, 39, 49, 59, 69$ and the detailed comparison results of terminal errors are summarized in Table 2. It follows from the results that the GT-FFT-DDP with $N = 19$ shows a divergence at early stage of the homing phase, as confirmed by Fig. 4. The results indicate that the performance of the proposed algorithm is robust against the variation of the number of nodes, while the GT-FFT-DDP algorithm is sensitive to this parameter. From Table 2, we can note that increasing the number of discretization nodes is helpful to improve the numerical accuracy for both algorithms. However, compared to GT-FFT-DDP, the proposed algorithm shows an improvement of around 6.5% in terms of terminal miss distance and 8.5% in terms of terminal impact angle error. Intuitively, the error between the two algorithms decreases as the number of discrete nodes increases. Notably, although increasing the number of discretization nodes can improve the numerical accuracy of the GT-FFT-DDP algorithm, this approach with $N = 19$ still becomes unstable at the early stage of the simulation, as shown in Fig. 4(c). Therefore, the proposed algorithm shows higher numerical stability and higher terminal accuracy than the GT-FFT-DDP algorithm.



(a) Interception trajectory (b) Time history of impact angle (c) Guidance command of interceptor

Fig. 4 Comparison simulation results against step maneuver target.

Table 2 Terminal conditions for GT-FFT-LGRDDP and GT-FFT-DDP.

Terminal Condition	Method	19	29	39	49	59	69
Terminal Miss Distance	GT-FFT-LGRDDP	2.229m	2.096m	2.006m	1.956m	1.925m	1.902m
	GT-FFT-DDP	\	2.239m	2.107m	2.033m	1.988m	1.956m
	Improvement	\	6.39%	4.79%	3.79%	3.17%	2.76%
Terminal Impact Angle Error	GT-FFT-LGRDDP	0.0949°	0.0892°	0.0867°	0.0854°	0.0521°	0.0515°
	GT-FFT-DDP	\	0.0973°	0.0925°	0.0899°	0.0557°	0.0546°
	Improvement	\	8.32%	6.27%	5.01%	6.46%	5.68%

VI. Conclusions

This Note proposes a new GT-FFT-LGRDDP algorithm to solve the nonlinear differential game problem. The proposed algorithm revisits the GT-DDP by adopting a LGR quadrature scheme and dynamically adjusting the final time. The key features of the algorithm developed are twofolds: the computational complexity is reduced with guaranteed numerical accuracy, and the terminal time is optimized instead heuristically set beforehand. A two-dimensional impact-angle-constrained missile guidance problem with a realistic interceptor model is considered to verify the effectiveness of the proposed method. The simulation results reveal that the proposed algorithm is robust against to the unpredictable target maneuver and the variation of the number of discretization nodes.

From the simulation results, it can be concluded that dynamically adjusting the final time in GT-DDP is helpful to cope with the unknown target maneuvers than pre-setting the flight time. In addition, the results also reveal that the Gaussian quadrature provides the capability of improving the numerical accuracy of the algorithm compared to conventional Euler discretization scheme. Therefore, the Gaussian integration method can be used in similar scenarios where the numerical accuracy of the algorithm needs to be improved. Future work includes theoretical convergence analysis and computational burden alleviation.

References

[1] Mastalli, C., Merkt, W., Marti-Saumell, J., Ferrolho, H., Solà, J., Mansard, N., and Vijayakumar, S., "A feasibility-driven approach to control-limited DDP," *Autonomous Robots*, 2022. https://doi.org/10.1007/s10514-022-10061-w.

[2] He, S., Shin, H.-S., and Tsourdos, A., "Trajectory optimization for target localization with bearing-only measurement," *IEEE Transactions on Robotics*, Vol. 35, No. 3, 2019, pp. 653–668. https://doi.org/10.1109/TRO.2019.2896436.

[3] He, S., Shin, H.-S., and Tsourdos, A., "Trajectory optimization for multitarget tracking using joint probabilistic data association filter," *Journal of Guidance, Control, and Dynamics*, Vol. 43, No. 1, 2020, pp. 170–178. https://doi.org/10.2514/1.G004249.

[4] Aziz, J. D., Scheeres, D. J., and Lantoine, G., "Hybrid differential dynamic programming in the circular restricted three-body problem," *Journal of Guidance, Control, and Dynamics*, Vol. 42, No. 5, 2019, pp. 963–975. https://doi.org/10.2514/1.G003617.

[5] Ozaki, N., Campagnola, S., Funase, R., and Yam, C. H., "Stochastic differential dynamic programming with unscented

transform for low-thrust trajectory design," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 2, 2018, pp. 377–387. https://doi.org/10.2514/1.G002367.

[6] He, S., Shin, H.-S., and Tsourdos, A., "Computational guidance using sparse Gauss-Hermite quadrature differential dynamic programming," *IFAC-PapersOnLine*, Vol. 52, No. 12, 2019, pp. 13–18. https://doi.org/10.1016/j.ifacol.2019.11.062.

[7] Mayne, D., "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems," *International Journal of Control*, Vol. 3, No. 1, 1966, pp. 85–95. https://doi.org/10.1080/00207176608921369.

[8] Pavlov, A., Shames, I., and Manzie, C., "Interior point differential dynamic programming," *IEEE Transactions on Control Systems Technology*, Vol. 29, No. 6, 2021, pp. 2720–2727. https://doi.org/10.1109/TCST.2021.3049416.

[9] Manchester, Z., and Kuindersma, S., "Derivative-free trajectory optimization with unscented dynamic programming," *2016 IEEE 55th Conference on Decision and Control (CDC)*, IEEE, 2016, pp. 3642–3647. https://doi.org/10.1109/CDC.2016.7798817.

[10] González-Arribas, D., Soler, M., and Sanjurjo-Rivo, M., "Robust aircraft trajectory planning under wind uncertainty using optimal control," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 3, 2018, pp. 673–688. https://doi.org/10.2514/1.G002928.

[11] Matsuno, Y., Tsuchiya, T., Wei, J., Hwang, I., and Matayoshi, N., "Stochastic optimal control for aircraft conflict resolution under wind uncertainty," *Aerospace Science and Technology*, Vol. 43, 2015, pp. 77–88. https://doi.org/10.1016/j.ast.2015.02.018.

[12] Hernández-Romero, E., Valenzuela, A., and Rivas, D., "Probabilistic multi-aircraft conflict detection and resolution considering wind forecast uncertainty," *Aerospace Science and Technology*, Vol. 105, 2020, p. 105973. https://doi.org/10.1016/j.ast.2020.105973.

[13] Bardhan, R., and Ghose, D., "Nonlinear differential games-based impact-angle-constrained guidance law," *Journal of Guidance, Control, and Dynamics*, Vol. 38, No. 3, 2015, pp. 384–402. https://doi.org/10.2514/1.G000940.

[14] Cho, D., Kim, H. J., and Tahk, M.-j., "Impact angle constrained sliding mode guidance against maneuvering target with unknown acceleration," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 51, No. 2, 2015, pp. 1310–1323. https://doi.org/10.1109/TAES.2015.140358.

[15] Cho, D., Kim, H. J., and Tahk, M.-J., "Fast adaptive guidance against highly maneuvering targets," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 52, No. 2, 2016, pp. 671–680. https://doi.org/10.1109/TAES.2015.140958.

[16] Sun, W., Pan, Y., Lim, J., Theodorou, E. A., and Tsiotras, P., "Min-max differential dynamic programming: Continuous and discrete time formulations," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 12, 2018, pp. 2568–2580. https://doi.org/10.2514/1.G003516.

[17] Sun, W., Theodorou, E. A., and Tsiotras, P., "Game theoretic continuous time differential dynamic programming," *2015 American control conference (ACC)*, IEEE, 2015, pp. 5593–5598. https://doi.org/10.1109/ACC.2015.7172215.

[18] Zheng, X., He, S., and Lin, D., "Constrained Trajectory Optimization with Flexible Final Time for Autonomous Vehicles," *IEEE Transactions on Aerospace and Electronic Systems*, 2021. https://doi.org/10.1109/TAES.2021.3121668.

[19] Gong, Q., Fahroo, F., and Ross, I. M., "Spectral algorithm for pseudospectral methods in optimal control," *Journal of Guidance, Control, and Dynamics*, Vol. 31, No. 3, 2008, pp. 460–471. https://doi.org/10.2514/1.32908.

[20] Morelli, A. C., Hofmann, C., and Topputo, F., "Robust Low-Thrust Trajectory Optimization Using Convex Programming and a Homotopic Approach," *IEEE Transactions on Aerospace and Electronic Systems*, 2021. https://doi.org/10.1109/TAES.2021.3128869.

[21] Mondal, S., and Padhi, R., "Constrained Quasi-Spectral MPSP With Application to High-Precision Missile Guidance With Path Constraints," *Journal of Dynamic Systems, Measurement, and Control*, Vol. 143, No. 3, 2021. https://doi.org/10.1115/1.4048488.

[22] Patterson, M. A., and Rao, A. V., "GPOPS-II: A MATLAB software for solving multiple-phase optimal control problems using hp-adaptive Gaussian quadrature collocation methods and sparse nonlinear programming," *ACM Transactions on Mathematical Software (TOMS)*, Vol. 41, No. 1, 2014, pp. 1–37. https://doi.org/10.1145/2558904.

[23] Huntington, G. T., "Advancement and analysis of a Gauss pseudospectral transcription for optimal control problems," Ph.D. thesis, MIT, 2007.

[24] He, S., Shin, H.-S., and Tsourdos, A., "Computational missile guidance: a deep reinforcement learning approach," *Journal of Aerospace Information Systems*, Vol. 18, No. 8, 2021, pp. 571–582. https://doi.org/10.2514/1.I010970.

[25] Kumar, S. R., Rao, S., and Ghose, D., "Nonsingular terminal sliding mode guidance with impact angle constraints," *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 4, 2014, pp. 1114–1130. https://doi.org/10.2514/1.62737.

[26] Cho, H., Ryoo, C.-K., Tsourdos, A., and White, B., "Optimal impact angle control guidance law based on linearization about collision triangle," *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 3, 2014, pp. 958–964. https://doi.org/10.2514/1.62910.

https://dspace.org/

Game-theoretic flexible-final-time differential dynamic programming using Gaussian quadrature

Zheng, Xiaobo; Guan, Haodong; Lin, Defu; He, Shaoming; Shin, Hyo-Sang

2023-04-01
American Institute of Aeronautics and Astronautics (AIAA)
https://doi.org/10.2514/1.G007113
Zheng X, Guan H, Lin D, et al., (2023) Game-theoretic flexible-final-time differential dynamic programming using Gaussian quadrature. Journal of Guidance, Control, and Dynamics, Volume 46, Issue 4, April 2023, pp. 742-751

Downloaded from a DSpace repository. https://dspace.org/