IFAC-PapersOnLine, Volume 52, Issue 12, 2019, pp.13-18
DOI:10.1016/j.ifacol.2019.11.062

Computational Guidance Using Sparse Gauss-Hermite Quadrature Differential Dynamic Programming

Shaoming He, Hyo-Sang Shin and Antonios Tsourdos

School of Aerospace, Transport and Manufacturing, Cranfield University, Cranfield MK43 0AL, UK (email: shaoming.he;h.shin;a.tsourdos@cranfield.ac.uk)

Abstract: This paper proposes a new computational guidance algorithm using differential dynamic programming and sparse Gauss-Hermite quadrature rule. By the application of sparse Gauss-Hermite quadrature rule, numerical differentiation in the calculation of Hessian matrices and gradients in differential dynamic programming is avoided. Based on the new differential dynamic programming approach developed, a three-dimensional computational algorithm is proposed to control the impact angle and impact time for an air-to-surface interceptor. Extensive numerical simulations are performed to show the effectiveness of the proposed approach.

Keywords: Computational guidance, Differential dynamic programming, Sparse Gauss-Hermite quadrature, Impact angle, Impact time

1. INTRODUCTION

Optimal guidance law and its variants have been widely used in aerospace guidance due to their systematically well-posed design process and ability to satisfy certain terminal constraints (Ryoo et al., 2005, 2006; He and Lee, 2018; He et al., 2019). Optimal guidance laws bring in the philosophy of trajectory shaping by optimizing a meaningful cost function and meeting certain boundary conditions. With the increasing complexity of application scenarios, however, real-world guidance problems in autonomous aerospace systems will be characterized by numerous practical constraints and highly time-varying, nonlinear dynamics. Therefore, classical closed-form optimal guidance laws, that rely on approximated models with linearization, unrealistic assumptions, and an offline design fashion, are no longer appealing to solve future real-world guidance problems.

Thanks to the rapid development on embedded computational capability, there has been an increasing attention on the development of computational guidance algorithms in recent years (Lu, 2017; Tsiotras and Mesbahi, 2017). Unlike classical optimal guidance laws, computational guidance algorithms generate the guidance command relies extensively on onboard computation and therefore dose not require analytic solution of specific guidance laws. Generally, computational guidance can be classified into two main categories: (1) model-based ; and (2) data-based. This paper mainly focuses on the model-based approach as it requires no training data in the implementation.

Dwivedi et al. (2011) proposed a model-based three-dimensional computational guidance algorithm with terminal flight path angle constraints using model predictive static programming (MPSP) (Padhi and Kothari, 2009). This basic idea behind MPSP is that it converts a dynamic programming problem into a static programming problem and therefore is computationally efficient. Due to this property, MPSP algorithm was later used in many practical guidance problems, e.g., impact-angle control guidance (Oza and Padhi, 2012), reentry guidance (Halbe et al., 2013), lunar landing (Banerjee et al., 2015). However, the major limitation of MPSP-based computational guidance algorithms is that they require a good initial solution guess to guarantee the convergence (Pan et al., 2019).

Computational guidance has also been studied in the perspective of differential dynamic programming (DDP) (Tassa et al., 2012; Manchester and Kuindersma, 2016; Gandhi and Theodorou, 2016; Sun et al., 2018). DDP is known as a second-order approximate dynamic programming algorithm and finds a locally optimal control command through an iterative way. Similar to Newton's method, DDP is an approximate second-order numerical algorithm that achieves quadratic convergence speed (Manchester and Kuindersma, 2016). The key of implementing DDP is to calculate the Hessian and gradient of the value function. Typical treatment of problem is to use numerical finite difference, which normally, however, has poor convergence property.

This paper proposes a new variant of DDP using sparse Gauss-Hermite quadrature (SGHQ) rule, thus termed as SGHQ-DDP. This is motivated by the recent application of unscented transformation in unscented dynamic programming (UDP) (Manchester and Kuindersma, 2016). Unlike UDP, we propose to use SGHQ to approximate the Hessian and gradient of the value function for the implementation of DDP. The SGHQ-DDP algorithm is then applied to the development of a new three-dimensional computational guidance approach to control the impact angle and impact time of an air-to-surface missile. Simulation results reveal that the proposed computational guidance algorithm has comparable complexity as UDP but with significantly improved numerical convergence property.

2. DIFFERENTIAL DYNAMIC PROGRAMMING

Consider the following general nonlinear discrete-time dynamic system

$$x_{k+1} = f_k(x_k, u_k) \tag{1}$$

where $x_k \in \mathbb{R}^n$ denotes the system states; $u_k \in \mathbb{R}^m$ refers to the control input; and $k = \{1, 2, \ldots, N\}$ stands for the discrete time index. Note that continuous-time systems can be discretized by using the well-known 4th order Runge-Kutta method.

The general performance index, or the so-called cost function, is given as

$$J = \ell_f(x_N) + \sum_{k=1}^{N-1} \ell(x_k, u_k) \tag{2}$$

where $\ell_f(x_N)$ is the final cost to cater for the terminal constraint and $\ell(x_k, u_k)$ denotes the running cost.

Given the initial state x_0, the optimal control input sequence $U^* = \{u_0^*, u_1^*, \ldots, u_{N-1}^*\}$ is the one that minimizes the cost function J as

$$U^* \triangleq \min_U J \tag{3}$$

Let $U_k = \{u_k, u_{k+1}, \ldots, u_{N-1}\}$ be the partial control sequence and define the cost-to-go $J(t_k)$, accumulated from time step k to time step N, as

$$J(t_k) = \ell_f(x_N) + \sum_{i=k}^{N-1} \ell(x_i, u_i) \tag{4}$$

The optimal cost-to-go, or the so-called value function, $V(x_k)$ is then obtained by substituting the optimal control sequence into Eq. (4) as

$$\begin{aligned} V(x_k) &= \min_{U_k} J(x_k, t_k) \\ &= \ell_f(x_N) + \sum_{i=k}^{N-1} \ell(x_i, u_i^*) \end{aligned} \tag{5}$$

According to Bellman equation, the value function satisfies a recursive form as

$$\begin{aligned} V(x_k) &= \min_{u_k} Q(x_k, u_k) \\ &= \min_{u_k} \left[\ell(x_k, u_k) + V(x_{k+1}, t_{k+1})\right] \end{aligned} \tag{6}$$

where $Q(x_k, u_k)$ is known as the action-value function.

Starting from $V(x_N) = \ell_f(x_N)$, dynamic programming finds the optimal control sequence by proceeding Eq. (5) backwards in time as an update. However, the value function $V(x_k)$ is in general nonlinear even with quadratic function due to the nonlinearity of dynamic system (1). This will result in the so-called curse-of-dimensionality and therefore finding the optimal solution is mathematically intractable in general. DDP addresses this problem by approximating the value function $V(x_k)$ around the nominal trajectory (x_k, u_k) using second-order Taylor series as

$$V(x_k + \delta x_k) \approx V(x_k) + V_x(t_k)^T \delta x_k + \frac{1}{2} \delta x_k^T V_{xx}(t_k) \delta x_k \tag{7}$$

where the notation $(\cdot)_x(t_k)$ refers to the partial derivative of $(\cdot)$ with respect to x evaluated at time t_k, i.e., $\frac{\partial(\cdot)}{\partial x}|_{x=x_k}$.

To approximate the value function using Eq. (7), consider the following second-order approximation of the perturbed action-value function $Q(x_k + \delta x_k, u_k + \delta u_k)$

$$\begin{aligned} Q(x_k + \delta x_k, u_k + \delta u_k) &\approx Q(x_k, u_k) + Q_x \delta x_k \\ &+ Q_u \delta u_k + \frac{1}{2} \begin{bmatrix} \delta x_k \\ \delta u_k \end{bmatrix}^T \begin{bmatrix} Q_{xx} & Q_{xu} \\ Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} \delta x_k \\ \delta u_k \end{bmatrix} \end{aligned} \tag{8}$$

where the Hessian matrices and gradient vectors of the action value function $Q(x_k, u_k)$ can be easily obtained by the definition of $Q(x_k, u_k)$ as

$$\begin{aligned} Q_x &= \ell_x(t_k) + f_x(t_k)^T V_x(t_{k+1}) \\ Q_u &= \ell_u(t_k) + f_u(t_k)^T V_x(t_{k+1}) \\ Q_{xx} &= \ell_{xx}(t_k) + f_x(t_k)^T V_{xx}(t_{k+1}) f_x(t_k) \\ &\quad + V_x(t_{k+1}) f_{xx}(t_k) \\ Q_{uu} &= \ell_{uu}(t_k) + f_u(t_k)^T V_{xx}(t_{k+1}) f_u(t_k) \\ &\quad + V_x(t_{k+1}) f_{uu}(t_k) \\ Q_{ux} &= \ell_{ux}(t_k) + f_u(t_k)^T V_{xx}(t_{k+1}) f_x(t_k) \\ &\quad + V_x(t_k) f_{ux}(t_k) \end{aligned} \tag{9}$$

Note that the second-order partial derivatives of the system dynamics, i.e., $f_{xx}(t_k)$, $f_{uu}(t_k)$, $f_{ux}(t_k)$, are rank-three tensors. Calculating these high-order tensors is relatively computationally expensive and therefore is usually ignored in the implementation of DDP, resulting the following approximated Hessian matrices

$$\begin{aligned} Q_{xx} &\approx \ell_{xx}(t_k) + f_x(t_k)^T V_{xx}(t_{k+1}) f_x(t_k) \\ Q_{uu} &\approx \ell_{uu}(t_k) + f_u(t_k)^T V_{xx}(t_{k+1}) f_u(t_k) \\ Q_{ux} &\approx \ell_{ux}(t_k) + f_u(t_k)^T V_{xx}(t_{k+1}) f_x(t_k) \end{aligned} \tag{10}$$

Substituting Eq. (8) to Bellman optimality condition gives

$$\begin{aligned} V(x_k + \delta x_k) &= \min_{\delta u_k} Q(x_k + \delta x_k, u_k + \delta u_k) \\ &\approx \min_{\delta u_k} \Big\{ Q(x_k, u_k) + Q_x \delta x_k + Q_u \delta u_k \\ &\quad + \frac{1}{2} \begin{bmatrix} \delta x_k \\ \delta u_k \end{bmatrix}^T \begin{bmatrix} Q_{xx} & Q_{xu} \\ Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} \delta x_k \\ \delta u_k \end{bmatrix} \Big\} \end{aligned} \tag{11}$$

To find the optimal control correction δu_k such that the second-order approximation of the action-value function is minimized, taking the gradient of Eq. (8) with respect to δu_k and setting it as zero yields the following control correction

$$\delta u_k = -Q_{uu}^{-1} Q_{ux} \delta x_k - Q_{uu}^{-1} Q_u \tag{12}$$

Substituting Eq. (12) back into Eq. (11) and grouping the zero-order, first-order and second-order terms of δx_k gives

$$\begin{aligned} V(x_k + \delta x_k) &= V(x_k) - \frac{1}{2} Q_u Q_{uu}^{-1} Q_u \\ V_x(t_k) &= Q_x - Q_u Q_{uu}^{-1} Q_{ux} \\ V_{xx}(t_k) &= Q_{xx} - Q_{xu} Q_{uu}^{-1} Q_{ux} \end{aligned} \tag{13}$$

Given an initial control sequence guess U_0, DDP and related algorithms utilize the recursion rule (13) backward from $k = N$ until the initial time step. At that point, a forward pass process is triggered by applying the corrected or updated control input to find a new nominal trajectory as

$$\begin{aligned}\hat{x}_0 &= x_0 \\ \hat{u}_k &= u_k + \delta u_k \\ &= u_k - Q_{uu}^{-1} Q_{ux} \left(\hat{x}_k - x_k\right) - Q_{uu}^{-1} Q_u \\ \hat{x}_{k+1} &= f\left(\hat{x}_k, \hat{u}_k\right)\end{aligned} \tag{14}$$

where $\hat{x}_k$ and $\hat{u}_k$ denote the updated system states and control input, respectively.

These backward and forward procedures are implemented iteratively until a locally optimal control sequence is obtained by DDP.

3. SPARSE GAUSS-HERMITE QUADRATURE DIFFERENTIAL DYNAMIC PROGRAMMING

As noted from Eqs. (9) and (10), the key of successful implementation of DDP is to accurately calculate the Hessian matrices and gradients of the action value function $Q\left(x_k, u_k\right)$. Typical treatment of this problem is to utilize numerical differentiation or unscented transformation as shown in the iLQR and UDP algorithms. In this section, we propose an alternative way by utilizing the recently proposed SGHQ approach to implement the DDP algorithm, thus termed as SGHQ-DDP. This newly developed algorithm is shown to have better numerical stability and comparable computational complexity in our simulations, as compared to UDP algorithm.

3.1 Calculating the Hessian Matrix of $Q\left(x_k, u_k\right)$

To begin with, define the following G matrix

$$G = \begin{bmatrix} G_1 & G_{12} \\ G_{12}^T & G_2 \end{bmatrix} \tag{15}$$

where

$$\begin{aligned} G_1 &= f_x\left(t_k\right)^T V_{xx}\left(t_{k+1}\right) f_x\left(t_k\right) \\ G_{12} &= f_x\left(t_k\right)^T V_{xx}\left(t_{k+1}\right) f_u\left(t_k\right) \\ G_2 &= \ell_{uu}\left(t_k\right) + f_u\left(t_k\right)^T V_{xx}\left(t_{k+1}\right) f_u\left(t_k\right) \end{aligned} \tag{16}$$

Notice that the Hessian matrix of $Q\left(x_k, u_k\right)$ can be formulated as

$$\begin{bmatrix} Q_{xx} & Q_{xu} \\ Q_{ux} & Q_{uu} \end{bmatrix} = G + \begin{bmatrix} \ell_{xx} & \ell_{xu} \\ \ell_{ux} & 0_m \end{bmatrix} \tag{17}$$

Then, calculating the Hessian matrix of $Q\left(x_k, u_k\right)$ reduces to the calculation of matrix G. Now, let us define augmented system state $\bar{x}_k = \left[x_k^T, u_k^T\right]^T$, augmented system dynamics $\bar{x}_{k+1} = \bar{f}_k\left(\bar{x}_k\right) = \left[f_k^T\left(x_k, u_k\right), u_k^T\right]^T$. Since

$$\begin{aligned} G &= \begin{bmatrix} f_x\left(t_k\right) & f_u\left(t_k\right) \\ 0_{m\times n} & I_m \end{bmatrix}^T \begin{bmatrix} V_{xx}\left(t_{k+1}\right) & 0_{n\times m} \\ 0_{m\times n} & \ell_{uu}\left(t_k\right) \end{bmatrix} \cdot \\ &\quad \times \begin{bmatrix} f_x\left(t_k\right) & f_u\left(t_k\right) \\ 0_{m\times n} & I_m \end{bmatrix} \\ &= \bar{f}_x\left(t_k\right)^T \underbrace{\begin{bmatrix} V_{xx}\left(t_{k+1}\right) & 0_{n\times m} \\ 0_{m\times n} & \ell_{uu}\left(t_k\right) \end{bmatrix}}_{P} \bar{f}_x\left(t_k\right) \end{aligned} \tag{18}$$

we have

$$G^{-1} = \left(\bar{f}_x\left(t_k\right)\right)^{-1} P^{-1} \left(\bar{f}_x\left(t_k\right)\right)^{-T} \tag{19}$$

Consider the following Cholesky decompositions

$$P^{-1} = \Gamma\Gamma^T, \quad G^{-1} = \Lambda\Lambda^T \tag{20}$$

Then, Eq. (19) can be rewritten as

$$\Lambda\Lambda^T = \left(\bar{f}_x\left(t_k\right)\right)^{-1} \Gamma\Gamma^T \left(\bar{f}_x\left(t_k\right)\right)^{-T} \tag{21}$$

which subsequently gives the following transformation

$$\Lambda = \left(\bar{f}_x\left(t_k\right)\right)^{-1} \Gamma \tag{22}$$

The preceding equation can be viewed as a backward propagation of Γ through dynamics $\bar{f}_k^-$. This provides us the opportunity to utilize the well-established transformation techniques developed for nonlinear filtering to approximate Λ. Similar to extended Kalman filter, one basic way to address this issue is to utilize the well-known first-order linearization technique to approximate the nonlinear dynamics. However, this approach only works well for weakly nonlinear dynamics and therefore might become unstable in applications. Alternative improvements over first-order linearization in filtering technique is unscented transformation (Julier and Uhlmann, 1997), and cubature rule (Arasaratnam and Haykin, 2009), which generate several sigma points to approximate the nonlinear transformations based on some specific rules. In a recent noteworthy contribution (Jia et al., 2011), a new nonlinear transformation method, called SGHQ, was developed and was proved to be computationally much more efficient than the conventional GHQ. This newly developed approach is also shown to be more flexible as it can achieve higher approximation accuracy than unscented transformation and cubature rule with only a moderate increase in the computational complexity. Motivated by this observation, this paper aims to utilize the SGHQ approach to approximate G.

With specified accuracy level L and increase manner m_L, we can get M points χ_l, $l \in \{1, 2, \cdots, M\}$ with associated weights ω_l using SGHQ approach. The relationships between M and L, m_L and the detailed sampling algorithm can be found in Jia et al. (2011). After getting χ_l, the corresponding samples can be generated as

$$\lambda_l = \bar{x}_{k+1} + \Gamma\chi_l \tag{23}$$

These M samples are then backward propagated through dynamics as

$$\lambda_l^- = \bar{f}_k^-\left(\lambda_l\right) \tag{24}$$

According to the SGHQ method, the matrix G can then be approximated using samples λ_l^- as

$$G^{-1} = \sum_{l=1}^{M} \omega_l \left(\lambda_l^- - \lambda_l\right)\left(\lambda_l^- - \lambda_l\right)^T \tag{25}$$

Substitution of Eq. (25) into Eq. (17) gives the approximation of the Hessian matrix of Q_i in terms of the backward propagated samples.

3.2 Calculating the Gradient of $Q\left(x_k, u_k\right)$

Except for the Hessian matrices of $Q\left(x_k, u_k\right)$, the gradients of $Q\left(x_k, u_k\right)$ with respect to x_k and u_k are also required to implement the DDP algorithm. This subsection utilizes the basic finite-difference approach to approximate these two gradients using the generated SGHQ samples. To begin with, define the means of samples before and after backpropagation as

$$\bar{\lambda} = \sum_{l=1}^{M} \omega_l \lambda_l, \quad \bar{\lambda}^- = \sum_{l=1}^{M} \omega_l \lambda_l^- \tag{26}$$

For simplicity, we define

$$\begin{aligned}\zeta_l &= V_x\left(t_{k+1}\right)^T \lambda_l\left(1:n\right)\\ \bar{\zeta} &= V_x\left(t_{k+1}\right)^T \bar{\lambda}\left(1:n\right)\end{aligned} \tag{27}$$

where $\lambda_l\left(1:n\right)$ refers to the first n elements of λ_l.

Then, according the principle of finite-difference, we have

$$\begin{bmatrix} \left(\lambda_1^- - \bar{\lambda}^-\right)^T \\ \left(\lambda_2^- - \bar{\lambda}^-\right)^T \\ \vdots \\ \left(\lambda_M^- - \bar{\lambda}^-\right)^T \end{bmatrix} \begin{bmatrix} f_x\left(t_k\right)^T V_x\left(t_{k+1}\right) \\ f_u\left(t_k\right)^T V_x\left(t_{k+1}\right) \end{bmatrix} = \begin{bmatrix} \zeta_1 - \bar{\zeta} \\ \zeta_2 - \bar{\zeta} \\ \vdots \\ \zeta_M - \bar{\zeta} \end{bmatrix} \tag{28}$$

Solving the preceding algebraic equation and substituting the solutions back to Eq. (9) we can get the approximations of the gradients of Q_i with respect to x and u. Using the results of previous two subsections, we can now implement the DDP algorithm using SGHQ approach that avoids the utilization of numerical differentiations.

4. ANGLE AND TIME CONSTRAINED COMPUTATIONAL GUIDANCE

In this section, a new three-dimensional computational guidance algorithm to control the impact time and the impact angle for an air-to-surface missile to intercept a stationary target is proposed by using the SGHQ-DDP algorithm developed. Assuming the interceptor is represented by an ideal mass model, i.e., with no autopilot delay, the three-dimensional kinematics can be described by the following differential equations

$$\begin{aligned}\dot{x}_m &= V_m \cos\gamma_m \cos\psi_m\\ \dot{y}_m &= V_m \cos\gamma_m \sin\psi_m\\ \dot{z}_m &= V_m \sin\gamma_m\\ \dot{V}_m &= -\frac{D}{m} - g\sin\gamma_m\\ \dot{\gamma}_m &= \frac{-a_z - g\cos\gamma_m}{V_m}\\ \dot{\psi}_m &= \frac{a_y}{V_m\cos\gamma_m}\end{aligned} \tag{29}$$

where $[x_m, y_m, z_m]^T$ denotes the inertial position of the missile. The notations V_m, γ_m and ψ_m stand for the velocity magnitude, the flight path angle and the azimuth angle, respectively. The variable m represents the mass of the vehicle and g is the gravitational acceleration. a_z and a_y are the lateral acceleration commands to be designed. The aerodynamic drag force D is modeled by

$$D = \frac{1}{2}\rho V_m^2 S C_D \tag{30}$$

where S is the reference area; C_D refers to the drag force coefficient; and ρ is the air density.

In order to intercept a stationary target at $[x_t, y_t, z_t]^T$ with zero miss distance, the following terminal constraints should be satisfied

$$\begin{aligned}x_m\left(t_f\right) &= x_t\\ y_m\left(t_f\right) &= y_t\\ z_m\left(t_f\right) &= z_t\end{aligned} \tag{31}$$

where t_f denotes the final impact time. Except for the miss distance constraint, the missile is required to hit the target at desired angles γ_d and ψ_d to increase the warhead kill effect. For this reason, we also enforce the following terminal impact angle constraints

$$\begin{aligned}\gamma_m\left(t_f\right) &= \gamma_d\\ \psi_m\left(t_f\right) &= \psi_d\end{aligned} \tag{32}$$

Additionally, the impact time constraint is also considered here to make the proposed guidance algorithm feasible in a salvo attack mission. This constraint can be expressed as

$$t_f = t_d \tag{33}$$

where t_d is the desired impact time.

In summary, the objective here is to design energy-optimal guidance commands a_z and a_y to satisfy terminal constraints (31)-(33), subject to (29). Define $x = [x_m, y_m, z_m, V_m, \gamma_m, \psi_m]^T$ and $u = [a_y, a_z]^T$ as the system state vector and control input vector, respectively. Then, the proposed SGHQ-DDP algorithm can be utilized to design a computational guidance algorithm by solving the following equivalent problem:

Find

$$\begin{aligned}u^* = \min_u \frac{1}{2}&\left[x\left(t_d\right) - x_d\right]^T S_f \left[x\left(t_d\right) - x_d\right]\\ &+ \frac{1}{2}\int_0^{t_d} \left(x - x_d\right)^T S\left(x - x_d\right) + u^T R u dt\end{aligned} \tag{34}$$

subject to (29) with $x_d = [x_t, y_t, z_t, \text{free}, \gamma_d, \psi_d]^T$. Notice that although the quadratic performance index is leveraged here, the value function is also nonlinear since the kinematics (29) is nonlinear.

5. NUMERICAL SIMULATIONS

In this section, the effectiveness of the proposed SGHQ-DDP computational guidance algorithm is validated by comparing with the recent UDP algorithm. The required aerodynamic model parameters and initial conditions are summarized in Table 1. The air density model is taken from Kee et al. (1998).

Table 1. Aerodynamic model parameters and initial conditions.

Parameter	Value
Missile's initial position	$[5000m, 5000m, 5000m]^T$
Missile's initial velocity	$300m/s$
Target's position	$[0m, 0m, 0m]^T$
Reference area	$0.0324m^2$
Drag coefficient	0.0169

To show the robustness of the proposed algorithm against the initial solution guess, we use the uncontrolled trajectory, i.e., $a_y = a_z = 0$, as the initialization for both SGHQ-DDP and UDP algorithms. The discretization time is set as $0.1s$ for both algorithms. The accuracy level L and increase manner m_L of the SGHQ rule are set as $L = 2$ and $m_L = 2$ in the implementation. The weighting matrices S_f, S and R are selected as $S_f = diag\left(10I_3, 0.01, 10^8 I_2\right)$, $S = 10^{-4}I_6$, $R = I_2$ without further tunings for all test scenarios.

To show the advantages of the proposed algorithm, four different scenarios, shown in Table 2, are considered to perform comparison simulations. The simulation results

Table 2. Conditions of different scenarios.

Parameter	Scenario 1	Scenario 2	Scenario 3	Scenario 4
$\gamma_m\,(0)$	$0°$	$0°$	$0°$	$0°$
$\psi_m\,(0)$	$250°$	$-100°$	$250°$	$250°$
γ_d	$-90°$	$-60°$	$-90°$	$-90°$
ψ_d	$200°$	$0°$	$200°$	$200°$
t_d	$30s$	$30s$	$35s$	$25s$

obtained from both UDP and the proposed SGHQ-DDP algorithms for four different scenarios are presented in Figs. 1-4. In all simulation scenarios, the miss distance and impact angle error of the proposed computational guidance algorithm are less than $2m$ and $2°$, respectively. This enables the air-to-surface missile to accurately intercept the ground target with desired impact angle. Compared to the proposed SGHQ-DDP algorithm, the UDP-based computational guidance approach results in mission failure for scenarios 2 and 3, and generates large miss distance for scenario 4. Even the missile guided by UDP-based algorithm successfully intercepts the target with desired impact angle in scenario 1, the commanded acceleration shows sudden change phenomenon when the missile is near the target. This fact is, obviously, not desirable for the stability of onboard control system. In contrast, the commanded acceleration generated by the proposed SGHQ-DDP computational guidance algorithm is very smooth and easy for implementation.

6. CONCLUSION

This paper has proposed a new computational guidance algorithm using differential dynamic programming. The implementation of differential dynamic programming is supported by the sparse Gauss-Hermite quadrature rule. This enables the calculation of Hessian matrices without using direct differentiation. Simulation results of a three-dimensional guidance problem reveal that the proposed computational guidance algorithm is robust against the initial solution guess and generates better performance, compared to the existing algorithm.

REFERENCES

Arasaratnam, I. and Haykin, S. (2009). Cubature kalman filters. *IEEE Transactions on Automatic Control*, 54(6), 1254–1269.

Banerjee, A., Padhi, R., and Vatsal, V. (2015). Optimal guidance for accurate lunar soft landing with minimum fuel consumption using model predictive static programming. In *American Control Conference (ACC), 2015*, 1861–1866. IEEE.

Dwivedi, P.N., Bhattacharya, A., and Padhi, R. (2011). Suboptimal midcourse guidance of interceptors for high-speed targets with alignment angle constraint. *Journal of Guidance, Control, and Dynamics*, 34(3), 860–877.

Gandhi, M. and Theodorou, E. (2016). A comparison between trajectory optimization methods: Differential dynamic programming and pseudospectral optimal control. In *AIAA Guidance, Navigation, and Control Conference*, 0385.

Halbe, O., Raja, R.G., and Padhi, R. (2013). Robust reentry guidance of a reusable launch vehicle using model predictive static programming. *Journal of Guidance, Control, and Dynamics*, 37(1), 134–148.

He, S. and Lee, C.H. (2018). Optimality of error dynamics in missile guidance problems. *Journal of Guidance, Control, and Dynamics*, 41(7), 1624–1633.

He, S., Lee, C.H., Shin, H.S., and Tsourdos, A. (2019). Minimum-effort waypoint-following guidance. *Journal of Guidance, Control, and Dynamics*.

Jia, B., Xin, M., and Cheng, Y. (2011). Sparse gauss-hermite quadrature filter with application to spacecraft attitude estimation. *Journal of Guidance, Control, and Dynamics*, 34(2), 367–379.

Julier, S.J. and Uhlmann, J.K. (1997). New extension of the kalman filter to nonlinear systems. In *Signal processing, sensor fusion, and target recognition VI*, volume 3068, 182–194. International Society for Optics and Photonics.

Kee, P., Dong, L., and Siong, C. (1998). Near optimal midcourse guidance law for flight vehicle. In *36th AIAA Aerospace Sciences Meeting and Exhibit*, 583.

Lu, P. (2017). Introducing computational guidance and control. *Journal of Guidance, Control, and Dynamics*, 40(2), 193–193.

Manchester, Z. and Kuindersma, S. (2016). Derivative-free trajectory optimization with unscented dynamic programming. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, 3642–3647. IEEE.

Oza, H.B. and Padhi, R. (2012). Impact-angle-constrained suboptimal model predictive static programming guidance of air-to-ground missiles. *Journal of Guidance, Control, and Dynamics*, 35(1), 153–164.

Padhi, R. and Kothari, M. (2009). Model predictive static programming: a computationally efficient technique for suboptimal control design. *International Journal of Innovative Computing, Information and Control*, 5(2), 399–411.

Pan, B., Ma, Y., and Yan, R. (2019). Newton-type methods in computational guidance. *Journal of Guidance, Control, and Dynamics*, 42(2), 377–383.

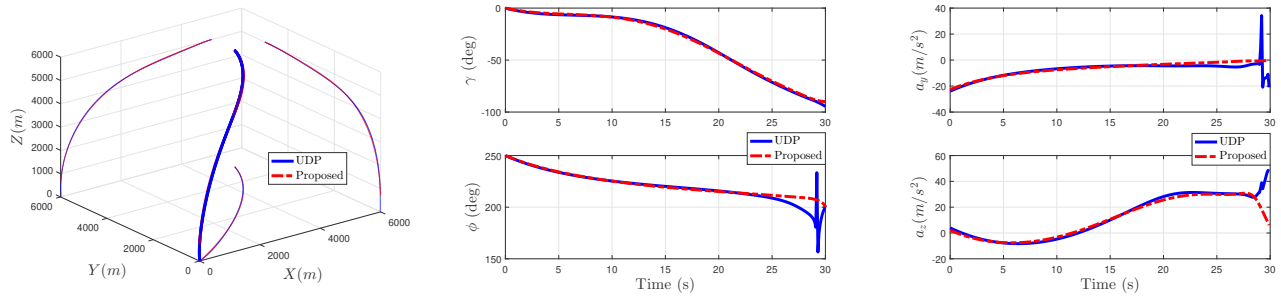
Ryoo, C.K., Cho, H., and Tahk, M.J. (2005). Optimal guidance laws with terminal impact angle constraint. *Journal of Guidance, Control, and Dynamics*, 28(4), 724–732.

Ryoo, C.K., Cho, H., and Tahk, M.J. (2006). Time-to-go weighted optimal guidance with impact angle constraints. *IEEE Transactions on Control Systems Technology*, 14(3), 483–492.

Sun, W., Pan, Y., Lim, J., Theodorou, E.A., and Tsiotras, P. (2018). Min-max differential dynamic programming: Continuous and discrete time formulations. *Journal of Guidance, Control, and Dynamics*, 41(12), 2568–2580.

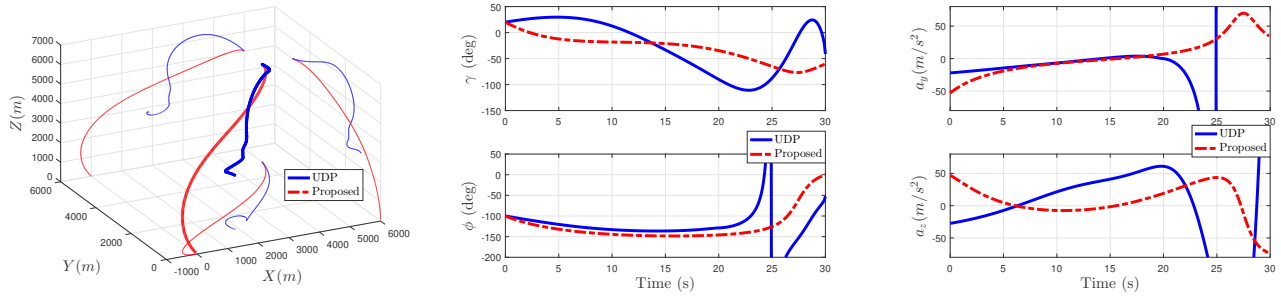
Tassa, Y., Erez, T., and Todorov, E. (2012). Synthesis and stabilization of complex behaviors through online trajectory optimization. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 4906–4913. IEEE.

Tsiotras, P. and Mesbahi, M. (2017). Toward an algorithmic control theory. *Journal of Guidance, Control, and Dynamics*, 40(2), 194–196.



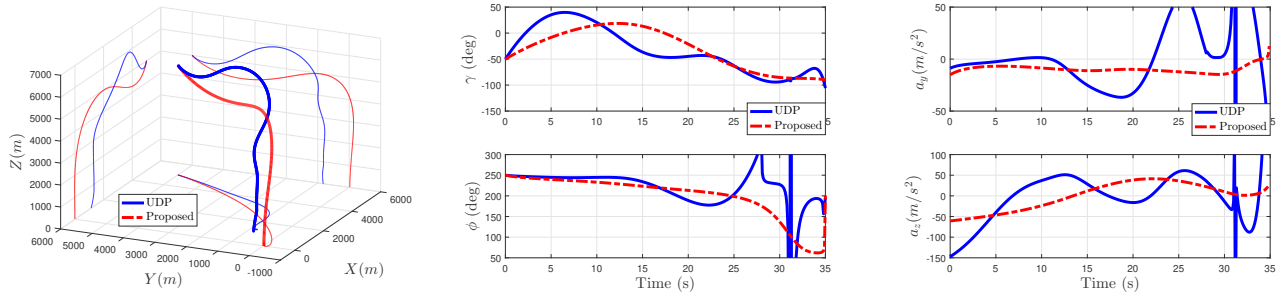
(a) Flight trajectory (b) Angle history (c) Commanded acceleration

Fig. 1. Comparisons results for scenario 1.



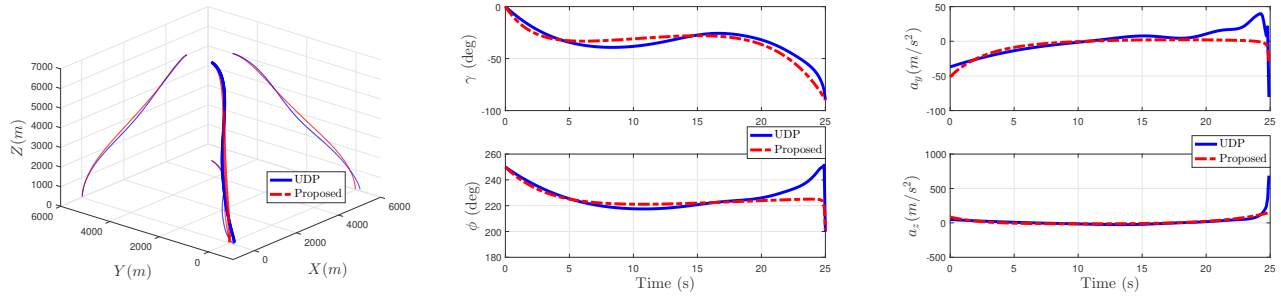
(a) Flight trajectory (b) Angle history (c) Commanded acceleration

Fig. 2. Comparisons results for scenario 2.



(a) Flight trajectory (b) Angle history (c) Commanded acceleration

Fig. 3. Comparisons results for scenario 3.



(a) Flight trajectory (b) Angle history (c) Commanded acceleration

Fig. 4. Comparisons results for scenario 4.