Incremental Generalized Policy Iteration for Adaptive Attitude Tracking Control of a Spacecraft

Li, Yifei; Kampen, Erik Jan van

DOI
10.23919/ECC57647.2023.10178221

Publication date
2023

Document Version
Final published version

Published in
2023 European Control Conference, ECC 2023

2023 European Control Conference (ECC)
June 13-16, 2023. Bucharest, Romania

Incremental Generalized Policy Iteration for Adaptive Attitude Tracking Control of a Spacecraft

Yifei Li[1] and Erik-Jan van Kampen[2]

Abstract— This paper proposes a novel dynamic programming algorithm for nonlinear system optimal control problem, namely Incremental Generalized Policy Iteration (IGPI). The proposed IGPI algorithm combines the advantages of Incremental Control(IC) and Generalized Policy Iteration(GPI). Incremental control can handle the nonlinearity and uncertainty in nonlinear systems without knowing the nonlinear system information, GPI can learn an optimal control law for dynamical systems. Based on the proposed IGPI algorithm, a data-driven adaptive attitude controller is designed for a spacecraft with sloshing liquid fuel. Simulation results demonstrate the effectiveness of the spacecraft attitude controller.

I. INTRODUCTION

Reinforcement learning (RL) is an effective method to solve the Hamilton-Jacob-Bellman (HJB) equation for discrete-time infinite horizon optimal control problems for nonlinear systems[1]. Most RL algorithms, such as Heuristic Dynamic Programming[2], Dual Heuristic Dynamic Programming[3], Soft Actor-Critic[4], follow the same basic idea, namely Generalized Policy Iteration (GPI)[5]. GPI is a computational method that conducts the policy evaluation and policy improvement steps recursively to acquire the optimal control policy.

Compared to the well-known policy iteration (PI), GPI does not precisely approximate the cost function in the policy evaluation step, such that the computation load is reduced. Theoretical results of GPI are mostly focusing on the admissible initial control[5], convergence and stability[6], and cost function approximation with adaptive iteration[7]. However, GPI requires a system model to predict the states in next time step, which could be inaccurate in practice for physical systems with high nonlinearity and model uncertainties.

Incremental control method can handle system nonlinearities and uncertainties by linearizing the nonlinear system[8], [9], [10]. Recent works have combined incremental control and reinforcement learning algorithms to develop model-free optimal controllers[8], [9], [10]. In [8], the incremental control is combined with approximate linear dynamic programming for air vehicle angle of attack tracking with output feedback. In [9], the partial observability problem is considered and the developed incremental approximate dynamic programming algorithm is applied for satellite attitude tracking problem with sloshing liquid fuel. In [10], an Incremental Global Dual Heuristic Programming (IGDHP) algorithm is developed for F-16 aircraft flight control.

Following the idea of combining incremental control and reinforcement learning algorithms, this paper develops a new model-free RL algorithm, namely Incremental Generalized Policy Iteration (IGPI). IGPI adopts Recursive Least Square (RLS) identification to learn an incremental model of physical systems, the identified incremental model is then used to predict model states such that generalized policy iteration can be conducted.

The contributions of this paper are summarized as follows.

- A new algorithm is developed to solve the HJB equation, through combining system identification and generalized policy iteration, namely incremental generalized policy iteration.
- Incremental control and RLS identification algorithm are combined to achieve state prediction without knowing the nonlinear system model perfectly. The resulting IGPI algorithm has the improved online performance because it identifies a linearized model.
- The IGPI algorithm is applied to develop an adaptive optimal attitude controller for a spacecraft, to deal with the effect of unknown internal dynamics caused by sloshing liquid.

The remainder of this paper is structured as follows. Section II presents the HJB equation for the discrete-time infinite horizon optimal control problem, and the fundamental idea of the GPI algorithm. Section III develops the IGPI algorithm by combining incremental control and GPI. Section IV provides the continuous-time longitudinal model of a spacecraft with sloshing fuel, and discretizes it into an incremental model. Section V provides the simulation results of IGPI algorithm in adaptive attitude tracking problem of a spacecraft. The conclusion of this paper is provided in Section VI.

[1]Yifei Li is with Faculty of Aerospace Engineering, Delft University of Technology, 2629HS Delft, The Netherlands Y.Li-34@tudelft.nl

[2]Erik-Jan van Kampen is with Faculty of Aerospace Engineering, Delft University of Technology, Delft, 2629HS Delft, the Netherlands E.vanKampen@tudelft.nl

II. PROBLEM FORMULATION

A. Discrete-time Hamilton-Jacob-Bellman Equation

The control-affine nonlinear dynamical system in discrete-time form is presented as[11]

$$x_{k+1} = f(x_k) + g(x_k)u(x_k) \tag{1}$$

where $x_k \in \mathbb{R}^n, f(x_k) \in \mathbb{R}^n, g(x_k) \in \mathbb{R}^{n\times m}$, and the input $u_k \in \mathbb{R}^m$. Assume that the system (1) is stabilizable on a prescribed compact set $\Omega \in \mathbb{R}^n$.

Definition 1: Stabilizable System. A nonlinear dynamical system is defined to be stabilizable on a compact set $\Omega \in \mathbb{R}^n$ if there exists a control input $u_k \in \mathbb{R}^m$ such that, for all initial conditions $x_0 \in \Omega$, the state $x_k \to 0$ as $k \to \infty$.

It is desired to find the control action u_k which minimizes the infinite-horizon cost function given as

$$V(x_k) = \sum_{n=k}^{\infty} \left[Q(x_n) + u_n^T R u_n\right], \text{for all } \mathrm{x_n} \tag{2}$$

where $Q(x_n) > 0$ and $R > 0 \in \mathbb{R}^{m\times m}$. The controllers need to be stable and also to guarantee that (2) is finite, i.e., the control must be admissible. The definition of admissible control is given as[11]

Definition 2: Admissible Control. A control $u(x_k)$ is defined to be admissible with respect to cost function (2) on Ω if $u(x_k)$ is continuous on a compact set $\Omega \in \mathbb{R}^n$, $u(0) = 0$, u stabilizes dynamical system (1) on Ω, and $\forall x_0 \in \Omega$, $V(x_0)$ is finite.

Equation (2) can be rewritten as

$$\begin{aligned} V(x_k) =& Q(x_k) + u_k^T R u_k + \sum_{n=k+1}^{\infty} \left(Q(x_n) + u_n^T R u_n\right) \\ =& Q(x_k) + u_k^T R u_k + V(x_{k+1}), \text{for all } \mathrm{x_n} \end{aligned} \tag{3}$$

According to Bellman's optimality principle[12], in discrete-time infinite-horizon optimal control problem, the optimal cost function $V^*(x_k)$ is a constant which makes the following HJB equation hold:

$$V^*(x_k) = \min_{u_k}(x_k^T Q(x_k) x_k + u_k^T R u_k + V^*(x_{k+1})) \tag{4}$$

The optimal control u_k^* for $V^*(x_k)$ meets the following partial differential equation:

$$\frac{\partial(x_k^T Q(x_k) x_k + u_k^T R u_k)}{\partial u_k} + \left(\frac{\partial x_{k+1}}{\partial u_k}\right)^T \frac{\partial V^*(x_{k+1})}{\partial x_{k+1}} = 0 \tag{5}$$

Solving (5) yields

$$u^*(x_k) = -\frac{1}{2} R^{-1} g(x_k)^T \frac{\partial V^*(x_{k+1})}{\partial x_{k+1}} \tag{6}$$

Substitute (6) into (4)

$$\begin{aligned} V^*(x_k) = & x_k^T Q(x_k) x_k + \frac{1}{4} \frac{\partial V^{*T}(x_{k+1})}{\partial x_{k+1}} g(x_k) R^{-1} g(x_k)^T \frac{\partial V^*(x_{k+1})}{\partial x_{k+1}} \\ & + V^*(x_{k+1}) \end{aligned} \tag{7}$$

Eq.(7) is the discrete-time HJB equation which describes the trajectory of optimal cost function $V^*(x_k)(k = 0,1,2,\cdots)$. When the system is linear and the cost function is in a quadratic form, Eq.(7) is solvable[5]. For a nonlinear system, it cannot be solved analytically, which motivates the approximated numerical solution.

B. Generalized Policy Iteration Algorithm

In this subsection, the GPI algorithm will be introduced to solve the HJB equation in Eq.(7) numerically. The GPI algorithm includes two steps[5]: (1) policy evaluation; (2) policy improvement.

Policy evaluation.

$$V_{i,j_i+1}(x_k) = Q(x_k) + v_i(x_k)^T R v_i(x_k) + V_{i,j_i}[F(x_k, v_i(x_k))] \tag{8}$$

Policy improvement.

$$\begin{aligned} v_i(x_k) = & \arg\min_{v_i(x_k)} \left\{Q(x_k) + v_i(x_k)^T R v_i(x_k) \right. \\ & \left. + V_{i-1}\left[F(x_k, v_i(x_k))\right]\right\} \end{aligned} \tag{9}$$

where the iteration index i denotes the ith policy iteration, j_i is the jth value iteration for fixed policy v_i, $0 < i < N_i, 0 < j_i < N_{j_i}$. $F(x_k, v_i(x_k)) = f(x_k) + g(x_k)v_i(x_k)$. The initial value function $V_{i,0}(x_k)$ for fixed v_i is calculated using the results of $(i-1)$th iteration:

$$\begin{aligned} V_{i,0}(x_k) =& \min_{v_k} \left\{Q(x_k) + v_i(x_k)^T R v_i(x_k) + V_{i-1}\left[F(x_k, v_i(x_k))\right]\right\} \\ =& Q(x_k) + v_i(x_k)^T R v_i(x_k) + V_{i-1}\left[F(x_k, v_i(x_k))\right] \end{aligned} \tag{10}$$

Remark 1. The value of N_{j_i} determines two special cases: (1)when $N_{j_i} = 1$, GPI algorithm reduces to a value iteration (VI) algorithm; (2)when $N_{j_i} \to \infty$, the GPI algorithm turns to be a PI algorithm.

Algorithm 1: Generalized Policy Iteration (GPI) Algorithm[5]

Initialization:

Choose randomly states x_k in Ω_x, i.e., $X_k = (x_k^{(1)}, x_k^{(2)}, ..., x_k^{(p)})$, where p is a large positive integer;

Choose a computation precision ε;

Construct a sequence N_i, where $N_i \geq 0, i = 1,2,...$ is an arbitrary non-negative integer.

Iteration:

1: Let the iteration index $i = 0$. Obtain $V_{1,0}(x_k)$ and $v_1(x_k)$;

2: Let j_1 increase from 0 to N_1. For all $x_k \in \Omega_x$, update the iterative value function by

$$V_{1,j_1+1}(x_k) = U(x_k, v_1(x_k)) + V_{1,j_1}(F(x_k, v_1(x_k)))$$
$$U(x_k, v_1(x_k)) = Q(x_k) + v_1(x_k)^T R v_1(x_k)$$

3: Let $i = i+1$. For all $x_k \in \Omega_x$, do Policy Improvement

$$v_i(x_k) = \underset{u_k}{argmin} U(x_k, u_k) + V_{i-1}(F(x_k, u_k));$$

4: Let j_i increase from 0 to N_i. Do Policy Evaluation

$$V_{i,j_i+1}(x_k) = U(x_k, v_i(x_k)) + V_{i,j_i}(F(x_k, v_i(x_k)));$$

5: Let $V_i(x_k) = v_{i,N_i(x_k)}$;

6: For all $x_k \in \Omega_x$, if $V_{i-1}(x_k) - V_i(x_k) < \varepsilon$, the approximate optimal cost function and the control law are obtained. Go to Step 7. Else go to Step 3;

7: return $v_i(x_k)$ and $V_{i,j_i}(x_k)$.

end for

III. Incremental Generalized Policy Iteration

Despite the fact that GPI can solve the HJB function, it requires the full knowledge of system transition function. When the system transition function is not perfectly known, GPI will not work. This section firstly introduces an incremental approach as a linearization method for nonlinear control problems. The second part explicates the implementation of a recursive least squares algorithm. Finally, the incremental model-based GPI algorithm is developed to solve the HJB function when the system model is unknown.

A. Incremental Control Approach

The main ideal of incremental control approach is to approximate a nonlinear system with a time-varying linear model. This incremental model is then identified using system identification methods at each local discrete time step[13]. This technique has been used to control unknown nonlinear systems in [14], [15].

Considering the discrete-time nonlinear system (1), taking the Taylor expansion yields

$$x_{k+1} = x_k + F_{k-1}(x_k - x_{k-1}) + G_{k-1}(u_k - u_{k-1}) + O(\Delta x_k^2, \Delta u_k^2) \tag{11}$$

where $F_{k-1} = \partial[f(x) + g(x)u(x)]/\partial x|_{x_{k-1},u_{k-1}} \in \mathbb{R}^{n\times n}$ is the system transition matrix, and $G_{k-1} = \partial f(x,u)/\partial u|_{x_{k-1},u_{k-1}} \in \mathbb{R}^{n\times n}$ is the input distribution matrix at time step $k-1$ for discrete systems. $O(\Delta x_k^2, \Delta u_k^2)$ is the higher-order term of Taylor series.

This discrete-time system can be written in an incremental form as

$$\Delta x_{k+1} = F_{k-1}(\Delta x_k) + G_{k-1}(\Delta u_k) + O(\Delta x_k^2, \Delta u_k^2) \tag{12}$$

B. Recursive Least Squares Identification

Define the augmented state and the augmented system matrix as

$$\begin{cases} X_k = \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} \\ \hat{\Theta}_{k-1} = \begin{bmatrix} \hat{F}_{k-1} & \hat{G}_{k-1} \end{bmatrix}^T \end{cases} \tag{13}$$

where $\hat{F}_{k-1}, \hat{G}_{k-1}$ are approximations of F_{k-1}, G_{k-1}.

The one-step prediction of Δx_{k+1}^T is calculated as

$$\Delta \hat{x}_{k+1}^T = X_k^T \hat{\Theta}_{k-1} \tag{14}$$

The error between Δx_{k+1}^T and $\Delta \hat{x}_{k+1}^T$ is defined as

$$\varepsilon_k = \Delta x_{k+1}^T - \Delta \hat{x}_{k+1}^T \tag{15}$$

The estimate of the augmented system matrix $\hat{\Theta}_{k-1}$ is updated as

$$\hat{\Theta}_k = \hat{\Theta}_{k-1} + \frac{\Lambda_{k-1} X_k}{\kappa + X_k^T \Lambda_{k-1} X_k} \varepsilon_k \tag{16}$$

where Λ_{k-1} is the equal weighted estimation of the covariance matrix $\mathrm{Cov}(\hat{\Theta}_k - \hat{\Theta}_{k-1})$, which describes the confidence of the estimated $\hat{\Theta}_k$. $\kappa \in (0,1)$ is the forgetting factor. Λ_{k-1} is updated by

$$\Lambda_k = \frac{1}{\kappa}\left[\Lambda_{k-1} - \frac{\Lambda_{k-1} X_k X_k^T \Lambda_{k-1}}{\kappa + X_k^T \Lambda_{k-1} X_k}\right] \tag{17}$$

Remark 2. The value of κ provides a balance between noise rejection and time-varying parameter estimation. When $\kappa \to 1$, the RLS algorithm becomes equally weighted and behaves better at noise rejection; when $\kappa \to 0$, the RLS algorithm shows more adaptation to new measurements, and thus adapts better to time-varying parameters. For a satisfying performance in practice, κ is suggested in [16] to be assigned as $0.9 < \kappa < 0.995$.

C. Incremental Generalized Policy Iteration Algorithm

This subsection develops the Incremental Generalized Policy Iteration (IGPI) algorithm, which combines the incremental control and generalized policy iteration. The IGPI algorithm firstly estimates a linear system model using RLS system identification method, and then iteratively calculates an optimal control policy. The IGPI algorithm is seen in Algorithm 2.

The developed IGPI algorithm assumes that cost function $V_{i,j_i}(x_k)$ and control policy $v_i(x_k)$ in each iteration step are known. However, the cost function is the sum of infinite utility functions and difficult to calculate. Meanwhile, the mathematical expression of control policy is not known. In these cases, parameterized approximators (neural networks in our work) can be used to approximate the cost function and control policy.

Algorithm 2: Incremental Generalized Policy Iteration Algorithm

Initialization:

Choose randomly states x_k in Ω_x, i.e., $X_k = (x_k^{(1)}, x_k^{(2)}, ..., x_k^{(p)})$, where p is a large positive integer;

Choose a computation precision ε;

Construct a sequence $N_i \geq 0 (i = 1, 2, ...)$, N_i is an arbitrary non-negative integer.

Choose initial value $\hat{\Theta}_0 = [\hat{F}_0, \hat{G}_0]^T, \Lambda_0$

RLS Identification:

1: $\Delta \hat{x}_{k+1}^T = X_k^T \hat{\Theta}_{k-1}$

2: $\Delta x_{k+1} = x_{k+1} - x_k$

3: $\boldsymbol{\varepsilon}_k = \Delta x_{k+1}^T - \Delta \hat{x}_{k+1}^T$

4: $\hat{\Theta}_k = \hat{\Theta}_{k-1} + \frac{\Lambda_{k-1} X_k}{\kappa + X_k^T \Lambda_{k-1} X_k} \varepsilon_k$

5: $\Lambda_k = \frac{1}{\kappa} \left[\Lambda_{k-1} - \frac{\Lambda_{k-1} X_k X_k^T \Lambda_{k-1}}{\kappa + X_k^T \Lambda_{k-1} X_k} \right]$

Iteration:

6: Let the iteration index $i = 0$. Obtain $V_{1,0}(x_k)$ and $v_1(x_k)$;

7: Define $F(x_k, u_k) = x_k + \hat{F}_{k-1}(x_k - x_{k-1}) + \hat{G}_{k-1}(u_k - u_{k-1})$

9: Let j_1 increase from 0 to N_1. For all $x_k \in \Omega_x$, update the iterative value function by

$$V_{1,j_1+1}(x_k) = U(x_k, v_1(x_k)) + V_{1,j_1}[F(x_k, v_1(x_k))]$$

10: Let $i = i + 1$. For all $x_k \in \Omega_x$, do Policy Improvement

$$v_i(x_k) = \arg\min_{v_i(x_k)} \{U(x_k, v_i(x_k)) + V_{i-1}[F(x_k, v_i(x_k))]\};$$

11: Let j_i increase from 0 to N_i.

For all $x_k \in \Omega_x$, do Policy Evaluation

$$V_{i,j_i+1}(x_k) = U(x_k, v_i(x_k)) + V_{i,j_i}[F(x_k, v_i(x_k))];$$

12: Let $V_i(x_k) = v_{i,N_i(x_k)}$;

13: For all $x_k \in \Omega_x$, if $V_{i-1}(x_k) - V_i(x_k) < \varepsilon$, the approximate optimal cost function and the approximate optimal control law are obtained. Go to Step 7.

Else go to Step 3;

14: return $v_i(x_k)$ and $V_{i,j_i}(x_k)$.

end for

IV. Spacecraft Dynamical Model

This section provides the longitudinal dynamical model of a spacecraft with sloshing liquid. Firstly, a schematic of the spacecraft is introduced with basic definitions of state variables and coordinate systems. Secondly, the dynamical equations of rigid-body spacecraft and sloshing liquid are introduced to mathematically describe their motions. Finally, the rotational motions of spacecraft and liquid fuel are derived, which are used to design attitude controllers.

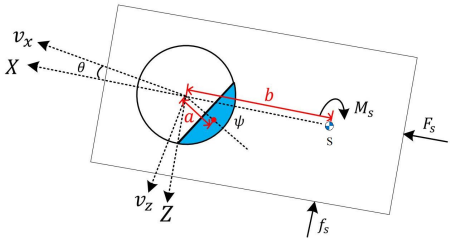

Fig. 1. A schematic of spacecraft model with sloshing liquid fuel. The coordinate system (X, Z) is fixed on spacecraft body, X is the central axis of spacecraft body and points to the front, Z is perpendicular to X and points down. The v_x axis in velocity coordinate system (v_x, v_z) is defined as the direction of the projection of spacecraft velocity in longitudinal plane, v_z is perpendicular to v_x and points down. S is the center of mass of rigid-body spacecraft

Figure 1 provides a schematic of the spacecraft with sloshing liquid fuel. The effect of sloshing liquid is caused by the oscillating movement of the liquid in a compartment. From the oscillation mode side, the movement of the sloshing liquid generates the wave in all oscillation modes, in which the first and the second modes cause the largest disruption in the system of a rigid-body spacecraft. In terms of the other oscillation modes, they show less aggressive behaviors than those of the first and the second modes. To avoid establishing a complex fluid model of the sloshing liquid, one can consider using an equivalent mechanical pendulum model to approximate the liquid dynamics[17]. The transitional and rotational models of spacecraft longitudinal plane are derived by using Lagrange equations, as in Refs.[9], [17], [18]. As a result, the rigid-body spacecraft and liquid fuel transitional motions are described as

$$\begin{aligned}&(m_s + m_p)(\dot{v}_x + v_z\dot{\theta}) + m_s b\ddot{\theta} + m_p a(\ddot{\psi} + \ddot{\theta})\sin(\ddot{\psi}) \\ &+ m_p a(\ddot{\psi} + \dot{\theta})^2 \cos(\psi) = F_s\end{aligned} \tag{18}$$

$$\begin{aligned}&(m_s + m_p)(\dot{v}_z - v_x\dot{\theta}) + m_s b\ddot{\theta} + m_p a(\ddot{\psi} + \ddot{\theta})\cos(\psi) \\ &- m_p a(\ddot{\psi} + \dot{\theta})^2 \sin(\psi) = f_s\end{aligned} \tag{19}$$

where v_x, v_z are velocities of spacecraft center of mass along x, z axis of velocity coordinate system (v_x, v_z). m_s, I_s, θ are the mass, moment of inertia, pitch angle of spacecraft. θ is defined as the angle between axis v_z in (v_x, v_z) and axis X in (X, Z). m_p, I_p, ψ are the fuel mass, moment of inertia, angle with respect to the longitudinal axis of a spacecraft. F_s is the constant thrust. f_s is the transverse force. a represents the overall length of the pendulum. b represents the relative distance between the center of mass of satellite and the connected point of pendulum. The values of spacecraft and liquid fuel can be seen in Table I.

The rigid-body spacecraft and liquid fuel rotational motions are modelled as a second-order dynamical nonlinear systems

$$m_s b(\dot{v}_z - v_x\dot{\theta}) + (I_s + m_s b^2)\ddot{\theta} - \kappa\ddot{\psi} = M_s + b f_s \tag{20}$$

$$\begin{aligned}&(m_p a^2 + I_p)(\ddot{\psi} + \ddot{\theta}) + m_p a[(\dot{v}_x + v_z\dot{\theta})\sin(\psi) \\ &+ (\dot{v}_z - v_x\dot{\theta})\cos(\psi)] + \kappa\ddot{\psi} = 0\end{aligned} \tag{21}$$

Rewrite (18), (19) as

$$\dot{v}_x = \frac{F_s - m_b b\dot{\theta} - m_p a(\ddot{\psi} + \ddot{\theta})\sin(\psi) - m_p a(\ddot{\psi} + \dot{\theta})^2 \cos(\psi)}{m_s + m_p} - v_z\dot{\theta} \tag{22}$$

$$\dot{v}_z = \frac{f_s - m_s b\ddot{\theta} - m_p a(\ddot{\psi} + \ddot{\theta})\cos(\psi) + m_p a(\ddot{\psi} + \dot{\theta})^2 \sin(\psi)}{m_s + m_p} + v_x\dot{\theta} \tag{23}$$

Substitute (22),(23) into (20),(21)

$$\ddot{\theta} = \frac{(m_s+m_p)(M_s+bf_s)+(m_s+m_p)\kappa\dot{\psi}}{(m_s+m_p)(I_s+m_sb^2)-m_s^2b^2-m_sm_pab\cos(\psi)} - \frac{m_sb[f_s-m_pa\ddot{\psi}\cos(\psi)+m_pa(\dot{\psi}+\dot{\theta})^2\sin(\psi)]}{(m_s+m_p)(I_s+m_sb^2)-m_s^2b^2-m_sm_pab\cos(\psi)} \tag{24}$$

$$\ddot{\psi} = -\frac{m_pa\sin(\psi)[F_s-m_sb\dot{\theta}-m_pa\ddot{\theta}\sin(\psi)-m_pa(\dot{\psi}+\dot{\theta})^2\cos(\psi)]}{(m_s+m_p)(m_pa^2+I_p)-m_pa\sin(\psi)m_pa\sin(\psi)-m_pa\cos^2(\psi)} - \frac{m_pa\cos(\psi)[f_s-m_sb\ddot{\theta}-m_pa\ddot{\theta}\cos(\psi)+m_pa(\dot{\psi}+\dot{\theta})^2\sin(\psi)]}{(m_s+m_p)(m_pa^2+I_p)-m_pa\sin(\psi)m_pa\sin(\psi)-m_pa\cos^2(\psi)} - \frac{(m_s+m_p)(m_pa^2+I_p)\ddot{\theta}-(m_s+m_p)\kappa\dot{\psi}}{(m_s+m_p)(m_pa^2+I_p)-m_pa\sin(\psi)m_pa\sin(\psi)-m_pa\cos^2(\psi)} \tag{25}$$

It is worth noting that the rotational motions of spacecraft and liquid fuel described as in (24),(25) are not affected by translational variables v_x, v_z, so that it can be simulated independently.

V. Simulation Results

This section provides the simulation results of a spacecraft attitude tracking problem using IGPI algorithm. The controller tracks the square-wave reference pitch angle θ_{ref}. The amplitude and frequency of θ_{ref} are 17.2° and $\frac{1}{300}Hz$, respectively. To implement the IGPI algorithm, two neural networks, namely critic and actor, are used. The updates of hyperparameters in critic and actor networks use gradient descent method[19]. The simulation environment is MATLAB R2021b with Intel(R) Core(TM) i5-9300H CPU @2.40GHz and 16GB RAM. The physical coefficients of the spacecraft are provided in Table I.

TABLE I
Spacecraft Physical Coefficients

Parameter	Value
Mass of spacecraft m_s	600kg
Mass of liquid fuel m_p	100kg
Spacecraft moment of inertia I_s	720kg/m^2
Fuel moment of inertia I_p	90kg/m^2
Thrust F_s	500N
Pendulum coefficient κ	0.19 (kg$\cdot$ m^2)/s
Length of pendulum a	0.3m
Distance from spacecraft center of mass to the pendulum connected point b	0.3m

A. Pitch Angle θ Set-point Tracking

Figure 2 shows that the spacecraft pitch angle θ follows the reference θ_{ref} in 60s. When the sign of θ_{ref} turns, θ tracks θ_{ref} in less than 50s. The pitch rate $\dot{\theta}$ converges to 0 after each change of reference θ_{ref}. The pitching moment M_s, the transverse force f_s are two control inputs of dynamical system consists of rigid-body spacecraft and liquid fuel. The results in Figure 3 show that M_s and f_s are capable of accomplishing tracking task of spacecraft pitch angle in the ranges $[-5,+5]$ and $[-3,+3]$, respectively. The stabilization of fuel attitude angle ψ is required for spacecraft attitude control. The motion of ψ is affected through interacting with spacecraft dynamics, as shown in Eq.(22),(23). In Figure 4, the results show that $\dot{\psi}$ is driven close to 0°/s, ψ oscillates near 0°.

B. Update of Critic and Actor Network Weights

Figure 5 provides the results of critic and actor networks weight matrices. The weight matrices of critic and actor networks $Y_{critic}, W_{critic}, Y_{actor}, W_{actor}$ start from their random initial values with 0 expectations. As a result, the elements of the critic network weight matrices converge in 100s and remain stable. The elements of the actor network weight matrices converge in less than 20s. As a result, the control command outputted by actor network can drive the pitch angle θ to its reference value θ_{ref}, while keeping the attitude angle ψ stable.

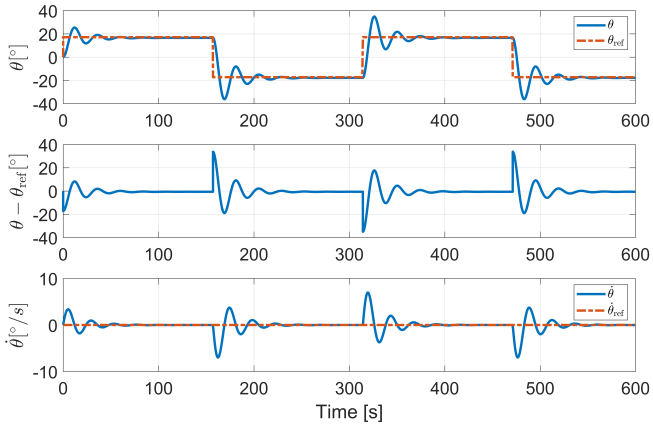

Fig. 2. Pitch Angle θ and pitch rate $\dot{\theta}$.

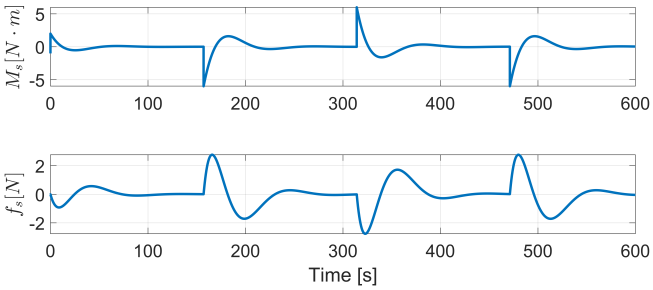

Fig. 3. Input moment M_s and input force f_s.

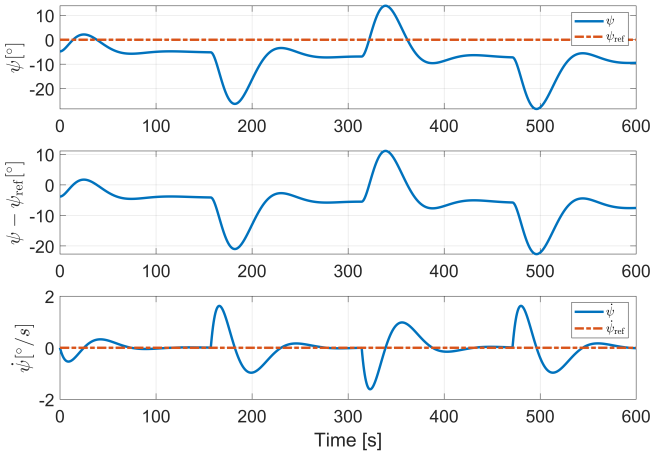

Fig. 4. Liquid attitude angle ψ and angle rate $\dot{\psi}$.

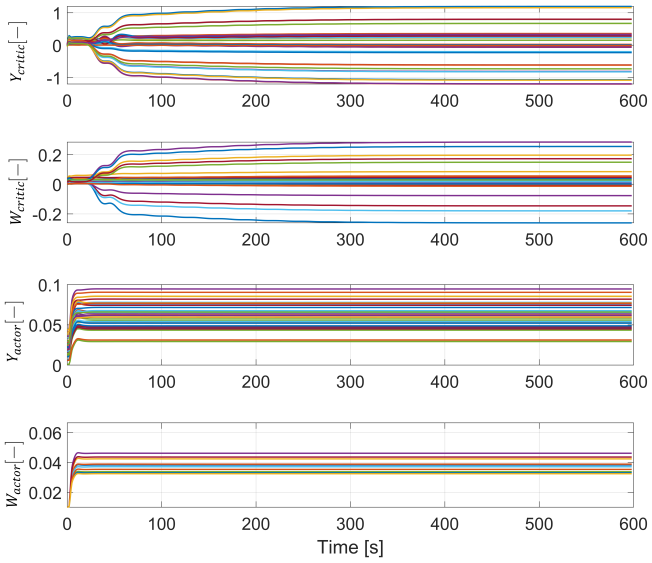

Fig. 5. Critic and actor network weight matrices.

VI. CONCLUSIONS

A reinforcement learning algorithm, namely incremental generalized policy iteration, is introduced to solve the discrete-time HJB function for infinite-time horizon optimal control problems. The uncertainties and nonlinearities of the system are dealt with by using RLS identification. As a result, the IGPI algorithm can be model-free. Simulation results of IGPI applied to a spacecraft attitude tracking problem demonstrate that it can deal with the uncertainty of the dynamical system caused by sloshing liquid fuel.

References

[1] B.Kiumarsi, K.G.Vamvoudakis, H.Modares, F.L.Lewis. Optimal and autonomous control using reinforcement learning: a survey. IEEE Transactions on Neural Networks and Learning Systems, 29(6), 2042-2062, 2017.

[2] P.J.Werbos. Advanced forceasting methods for global crisis warning and models of intelligence. General System Yearbook, 22: 25-28, 1977.

[3] D.V.Prokhorov, D.C.Wunsch. Adaptive critic designs. IEEE Transactions on Neural Networks, 8(5): 997-1077, 1997.

[4] T.Haarnoja, A.Zhou, P.Abbeel, et.al. Soft actot-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. International Conference on Machine Learning, 2018.

[5] D.Liu, Q.Wei, P.Yan. Generalized policy iteration adaptive dynamic programming for discrete-time nonlinear systems. IEEE Transactions on Systems, Man and Cybernetics: Systems, 45(12), 1577-1591, 2015.

[6] K.Vamvoudakis, D.Vrabie, F.L.Lewis. Online policy iteration based algorithms to solve the continuous-time infinite horizon optimal control problem. IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning, March, Nashville, USA, 2009.

[7] I.Puncochaf, J.Skach, M.Simandl. Adaptive generalized policy iteration in active fault detection and control 9th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Process: Paris, 2-4, September 2015.

[8] Y.Zhou, E.van Kampen, Q.Chu. Nonlinear adaptive flight control using incremental approximate dynamic programming and output feedback. Journal of Guidance, Control and Dynamics, 40(2), 493-500, 2017.

[9] Y.Zhou, E.van Kampen, Q.Chu. Incremental approximate dynamic programming for nonlinear adaptive tracking control with partial observability. Journal of Guidance, Control and Dynamics, 41(12), 2554-2567, 2018.

[10] B.Sun, E.van Kampen. Incremental model-based global dual heuristic programming with explicit analytical calculations applied to flight control. Engineering Applications of Artificial Intelligence, 89, 1-11, 2020.

[11] A.Al-Tamimi, F.L.Lewis, and M.Abu-Khalaf. Discrete-time nonlinear HJB solution using approximate dynamic programming: convergencec proof. IEEE Transactions on System, Man and Cybernetics, 38(4), 943-949, 2008.

[12] R.S.Sutton, A.G.Barto. Reinforcement learning: An Introduction, Cambridge, Massachussetts, USA: The MIT Press, 1998.

[13] S.Sieberling, Q.P.Chu, J.A.Mulder. Robust flight control using incremental nonlinear dynamic inversion and angular acceleration prediction. Journal of Guidance, Control and Dynamics, 33(6), 1732-1742, 2010.

[14] P.Acquatella, E.van Kampen, Q.P.Chu. Incremental backstepping for robust nonlinear flight control. Proceedings of the EuroGNC 2013:2rd CEAS Specialist Conference on Guidance, Navigation and Control, 4623, 2013.

[15] Y.Zhou, E.van Kampen, Q.P.Chu. An incremental approximate dynamic programming flight controller based on output feedback, AIAA Guidance, Navigation and Control Conference, 2016-0360, 2016.

[16] R.Isermann, M.Mnchhof. Identification of dynamic systems-an introduction with applications, Springer Heidelberg Dordrecht, New York, USA, 2011.

[17] L.C.de Souza, A.G.Zouza. Satellite attitude control system design considering fuel slosh dynamics. Shock and Vibration, 2014: 1-8, 2014.

[18] H.Zhang, Z.Wang. Attitude control and sloshing suppression for liquid-filled spacecraft in the presence of sinusoidal disturbance. Journal of Sound and Vibration, 383: 64-75, 2016.

[19] S.Sun, Z.Cao, H.Zhu, J.Zhao. A survey of optimization methods from a machine learning perspective. IEEE Transactions on Cybernetics, 50(8): 3668-3681, 2020.