ARL-TN-1288 ● Dec 2025

Hamilton–Jacobi–Bellman Enhanced Reinforcement Learning Control for a Linear System

by Bradley T. Burchett

NOTICES

Disclaimers

ARL-TN-1288 ● DEC 2025

Hamilton–Jacobi–Bellman Enhanced Reinforcement Learning Control for a Linear System

Bradley T. Burchett
DEVCOM Army Research Laboratory

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED START DATE	END DATE
December 2025	Technical Note	1 July 2025	30 September 2025

4. TITLE AND SUBTITLE
Hamilton–Jacobi–Bellman Enhanced Reinforcement Learning Control for a Linear System

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**

6. AUTHOR(S)
Bradley T. Burchett

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
DEVCOM Army Research Laboratory ATTN: FCDD-RLA-WD Aberdeen Proving Ground, MD 21005	ARL-TN-1288

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES
ORCID: Bradley T. Burchett, 0000-0002-1934-0537

14. ABSTRACT
Deep reinforcement learning (RL) has recently come to the attention of many researchers in various disciplines. In this note, we explore the application of Deep RL with a supervised learning enhancement from optimal control theory enabled by a physics informed neural network to the control of a linear state space model. Performance is found to be worse than solutions from optimal control theory.

15. SUBJECT TERMS
Weapons Sciences, deep learning, reinforcement learning, physics informed neural network, optimal control

16. SECURITY CLASSIFICATION OF: a. REPORT	b. ABSTRACT	C. THIS PAGE	17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UU	20

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
Bradley T. Burchett	(520) 691-5110

STANDARD FORM 298 (REV. 5/2020)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures **iv**

1. Introduction **1**

2. Benchmark Problem **2**

2.1 Proximal Policy Optimization 2

2.2 HJB Loss Term 3

2.3 The Plant Model 4

2.4 Results 5

3. Conclusion **10**

4. References **11**

List of Symbols, Abbreviations, and Acronyms **12**

Distribution List **14**

List of Figures

Figure 1. Closed-loop response with unstable plant.. 6
Figure 2. Closed-loop response with stable plant.. 7
Figure 3. Closed-loop response for 10 plant models, various initial conditions. . 8
Figure 4. Performance with varying penalty on pitch rate. 9

1. Introduction

Deep reinforcement learning (RL) is a very active field of research. Deep RL is based upon the decades-old idea of actor–critic neural networks where two networks are paired to provide feedback control (actor), critique the performance of the control (critic), and provide a training signal to tune the actor (critic). In deep RL, the two networks are used onboard an "agent," which must perform a task in an "environment." The environment provides "rewards" for progress toward a goal, and/or the achievement of a goal. The critic network tries to predict the cumulative future rewards given the current state, previous state, and previous action.

A large community of researchers, engineers, and programmers now contribute to open-source deep RL repositories. These repositories provide a starting point for newcomers to explore deep RL by using turnkey algorithms within the OpenAI Gymnasium environment.[1] Gymnasium, or Gym, is a standard architecture that defines an environment usually consisting of a system with dynamics. A Gym environment must contain function definitions for initialization, time step, reset, get observation, get info, and close. Additional subroutines can be defined for graphical displays and motion rendering.

Physics-informed neural networks (PINNs) are networks that use physical and constitutive relationships to guide training to a subset of solutions that faithfully represent the underlying physics of a system. Recently, new feedback control methods have been contrived through applying a PINN to solve partial differential equations (PDEs) that define the necessary conditions of optimality. Hamilton–Jacobi–Bellman (HJB) is one such PDE.[2–4] As such, PINNs are a supervised learning approach because the residual from the optimal solution can be computed exactly and gradients from this residual guide the search directly toward the local optimum. Continuous time deep RL is reinforcement learning in the sense that the HJB is approximated and the agent must explore the environment and update the HJB approximation based upon whether or not rewards improve in the search direction. By combining a value loss function based upon the HJB PINN, deep RL training can be accelerated by providing a supervised learning direction to the critic network.

In this report, we explore the application of an open-source proximal policy optimization (PPO) algorithm with HJB search enhancement. The HJB enhancement is done through a PINN with application of constrained expressions (CEs) to force satisfaction of the value function Dirichlet boundary condition. Performance is demonstrated on stable and unstable linear plant models. The controllers from deep RL are found to be suboptimal and not very robust.

2. Benchmark Problem

The algorithm was suggested by Scorsoglio et al.[5] who applied it to a nonlinear marginally stable plant. We previously demonstrated optimal control of that system[4] using a PINN with HJB residual as the sole loss function. Mukherjee and Liu[6] used the algorithm to control a 1-D PDE model of fluid-cooled battery packs. We adapted their code to control a system described by a linear ordinary differential equation with all states available for measurement and feedback.

2.1 Proximal Policy Optimization

PPO is a "model free" policy gradient method and part of the large family of actor–critic methods. As such, it consists of two neural networks—an actor and a critic. The actor network takes the current state vector as its input and computes an action vector. The critic network takes the state vector as input and returns the "value." The value function is tuned by gradient search to match the "advantage." Advantage is a function that approximates the rewards given the current state and action. The reward is computed and returned by the environment and is defined for our application in Section 2.2.

Policy gradient methods train the networks by estimating gradients of the loss function and using gradient ascent to move the network closer to an optimum. A basic gradient estimate has the form

$$\hat{g} = \mathrm{E}_t\left[\nabla_\theta \log \pi_\theta \,(a_t |\, s_t)\hat{A}_t\right] \tag{1}$$

Applying policy gradient directly can lead to oscillation and instability in training due to large update steps. Trust region methods were developed to limit the update step sizes. Instead of using the log expectation of the policy advantage estimate directly, a probability ratio is formed[7]

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)} \tag{2}$$

PPO uses a clipped probability ratio as the surrogate objective, specifically

$$L(\theta) = \mathrm{E}_t[\min(r_t(\theta)A_t, \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)] \tag{3}$$

where ϵ is a hyperparameter, typically $\epsilon = 0.2$, and the advantage is computed as[6]

$$A_t = \sum_{n=t}^{T-1} (\gamma\lambda)^{n-t}\delta_n \tag{4}$$

where γ is the discount factor, λ is the generalized advantage estimation (GAE) parameter, and Mukherjee and Liu[6]

$$\delta_t = R_t + \gamma V(\varphi, s_{t+1}) - V(\varphi, s_t) \tag{5}$$

$V(\varphi, s_t)$ is the critic network estimate of advantage given the network parameters φ and current state s_t.

2.2 HJB Loss Term

The HJB loss term, $MSE_f = \mathcal{R}^2$, is defined by the HJB PDE[4]

$$\mathcal{R} = -\mathbf{x}^T\mathbf{Q}\mathbf{x} + V_x^T\mathbf{f}(\mathbf{x}) - \frac{1}{4}V_x\mathbf{g}(\mathbf{x})\mathbf{R}^{-1}\mathbf{g}^T(\mathbf{x})V_x^T = 0 \tag{6}$$

where $\mathbf{f}(\mathbf{x})$ and $\mathbf{g}(\mathbf{x})$ are defined by the plant state dynamics.

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u} \tag{7}$$

V is the output of the value (critic) network, $V_x = \nabla_x V$, a row vector found by automatic differentiation, and $V(\mathbf{0}) = 0$ is the Dirichlet boundary condition. **Q** and **R** are positive definite matrices used to shape the quadratic cost function, which defines the optimal control problem.

$$\max_{\mathbf{u}\in U} J(\mathbf{x}, \mathbf{u}) = \int_{\mathbf{0}}^{\infty} -(\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{u}^{\mathrm{T}}\mathbf{R}\mathbf{u})\, dt \tag{8}$$

subject to Equation 7 and $\mathbf{x}(0) = \mathbf{x_0}$. The reward function models progress toward, or achievement of, a goal within the environment. Physically speaking, it is defined by the environment rather than the agent. As users, we can define that reward to incentivize behavior that matches optimal control. For this application, we define that reward as

$$R_t = 4(10^{-4})x_{1t}^2 + x_{2t}^2 + x_{3t}^2 + 10^{-1}x_{4t}^2 + 2u_{t-1}^2 \tag{9}$$

The **Q** and **R** matrices must accurately reflect the instantaneous reward function defined in the system Gym environment. Thus, $\mathbf{Q} = \mathrm{diag}([4(10^{-4}) \quad 1 \quad 1 \quad 10^{-1}])$ and **R** = 2.0. The design of R_t, and {**Q**, **R**} thus appears to be a *chicken and egg* paradox; however, once one is chosen, the other follows immediately.

Rather than treat $\boldsymbol{x} = \mathbf{0}$ and $V(\mathbf{0}) = 0$ as a separate training pair, we exercise a trick from CE[8] and set

$$
\begin{aligned}
V(\boldsymbol{x}) &= V_{NN}(\boldsymbol{x}) - V_{NN}(\boldsymbol{0}) \\
V_x(\boldsymbol{x}) &= \nabla_x V_{NN}(\boldsymbol{x}) - \nabla_x V_{NN}(\boldsymbol{0})
\end{aligned}
\tag{10}
$$

where V_{NN} is the critic network output, and $\nabla_x V_{NN}$ is the gradient with respect to the state found through automatic differentiation.

2.3 The Plant Model

For this report, we seek to control a subset of the linear models presented by Burchett and Bryson.[9] These models capture the dynamics of a fin-stabilized, tail-flap-controlled projectile flying between Mach 1 and Mach 3 with angle of attack between 0 and 10°. We specifically chose the four-state version of the linear models[9] because it captures both the slow phugoid and fast short period modes of the dynamics. Over the state flight envelope, both modes can be stable or unstable. The phugoid mode has a minimum open-loop settling time of 61.5 s. The short period mode is either unstable or oscillatory with a maximum damping ratio of 0.323, minimum settling time of 4.67 s, and minimum damping ratio of 0.0378.

The system is given as[9]

$$
\begin{bmatrix} \dot{u} \\ \dot{\theta} \\ \dot{\alpha} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} X_u & -gc_{\theta_0} & X_\alpha & -w_0 \\ 0 & 0 & 0 & 1 \\ Z_u/u_0 & -gs_{\theta_0}/u_0 & Z_\alpha/u_0 & 1 \\ M_u & M_\theta & M_\alpha + M_{\dot{\alpha}} Z_\alpha/u_0 & M_q + M_{\dot{\alpha}} \end{bmatrix} \begin{bmatrix} u \\ \theta \\ \alpha \\ q \end{bmatrix} + \begin{bmatrix} X_{\delta q} \\ 0 \\ Z_{\delta q}/u_0 \\ M_{\delta_q} \end{bmatrix} [\delta_q] \tag{11}
$$

where the elements are mostly dimensionalized sensitivities of aerodynamic coefficients. The state vector consists of the forward velocity u in m/s, the pitch θ and angle of attack α in rad, and the pitch rate q in rad/s. The control δ_q is given in degrees. For this application, we had greater success training the system without applying any kind of scaling, but rather choosing the cost function, optimal control weighting, and initial conditions wisely. Elements of the **Q** matrix are chosen to penalize non-zero states at each time step. The specific values are chosen based upon the relative scaling of each state shown in Equation 11. Pitch and angle of attack are roughly equal and measured in radians, so their magnitude is on the order of unity. Forward velocity u is on the order of 10^2 so u^{-2} is on the order of 10^{-4}. Pitch rate is on the order of 10^1 so q^{-2} is on the order of 10^{-2}.

For a typical flight condition, Equation 11 has the values

$$
\begin{bmatrix} \dot{u} \\ \dot{\theta} \\ \dot{\alpha} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} 0.0779 & -9.7172 & 18.261 & -140.12 \\ 0 & 0 & 0 & 1 \\ 2.6(10^{-5}) & -0.001 & -3.15 & 1 \\ -0.180 & -900.0 & -900 & -4.24 \end{bmatrix} \begin{bmatrix} u \\ \theta \\ \alpha \\ q \end{bmatrix} + \begin{bmatrix} -2.26 \\ 0 \\ 0.007 \\ 33.7 \end{bmatrix} [\delta_q] \tag{12}
$$

Note that δ_q has a large influence on $\dot{q}$, and less so on $\dot{u}$. More importantly, note that $\dot{\theta}$ is driven solely by q and $\dot{\alpha}$ is driven by q and a damping term with very weak coupling to the other states. Thus, θ and α will follow approximately the same trajectory. In other words, although all states are theoretically *reachable* for a *controllable* linear system, states with $\mathrm{sign}(\alpha) \neq \mathrm{sign}(\theta)$ are not reachable within a short time horizon—neither can they be recovered from within a short time. Considering that, we limit the initial conditions used in training to ones where $\theta(0) = \alpha(0)$.[10,11]

2.4 Results

We used a PPO actor–critic agent to control the system. The actor and critic networks each have two hidden layers with 64 neurons each and hyperbolic tangent activation. Each output layer is linear. The training algorithm is shown as Algorithm 1 in Mukherjee and Liu.[6] We adapted the code from Amartyamukherjee[12] to control a linear state space plant. We coded the plant model as a Gym environment with time step 10^{-3}, maximum control deflection = 10°, reward as shown in Equation 9, and dynamics as shown in Equation 11. Time stepping is done using a two-step Adams–Bashforth method.

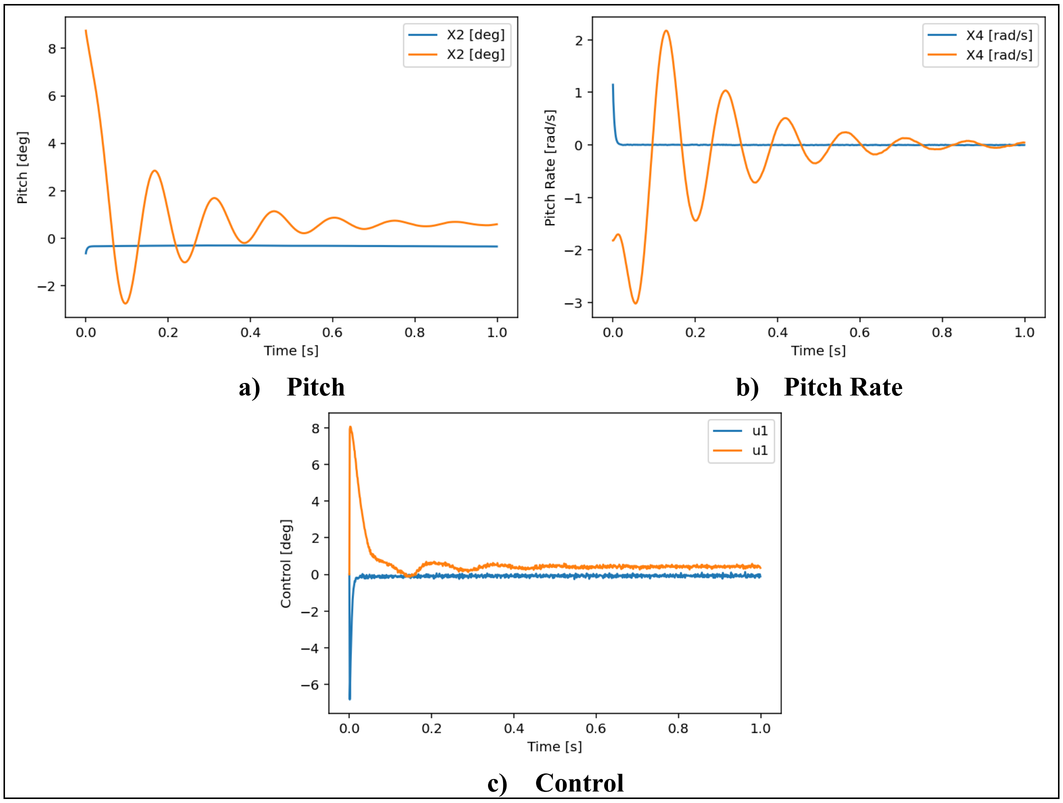
$$\mathbf{x}_{n+2} = \mathbf{x}_{n+1} + \frac{3}{2}h\dot{\mathbf{x}}(t_{n+1}, \mathbf{x}_{n+1}) - \frac{1}{2}h\dot{\mathbf{x}}(t_n, \mathbf{x}_n) \tag{13}$$

which simply requires the environment to remember the previous value of the state derivative.

The 100 linear models from Burchett and Bryson[9] may be categorized as 1) those with two stable modes, 2) those with an unstable phugoid mode, and 3) those with an unstable short period mode. We found that a single HJBPPO trained agent was only able to control plants within one of the categories. We also found that HJBPPO training was too slow and would oscillate or diverge unless the actor network was first trained to match a baseline stabilizing controller.

We trained three agents—one for each category of plant—using the hyperparameters $\epsilon = 0.2$, $\gamma = 0.99$, $\alpha_\pi = 3(10^{-4})$, and $\alpha_\varphi = 10^{-3}$. Max episode length was set to 1000 time steps or 1 s. Each agent was trained for about $1.5(10^6)$ total time steps.

Sample closed-loop responses are shown in Figures 1–3. Figure 1 shows the closed-loop response for two unstable plants with randomly chosen initial conditions. The agent reduces the settling time to about 1 s. Due to instability, the open-loop settling time is undefined.

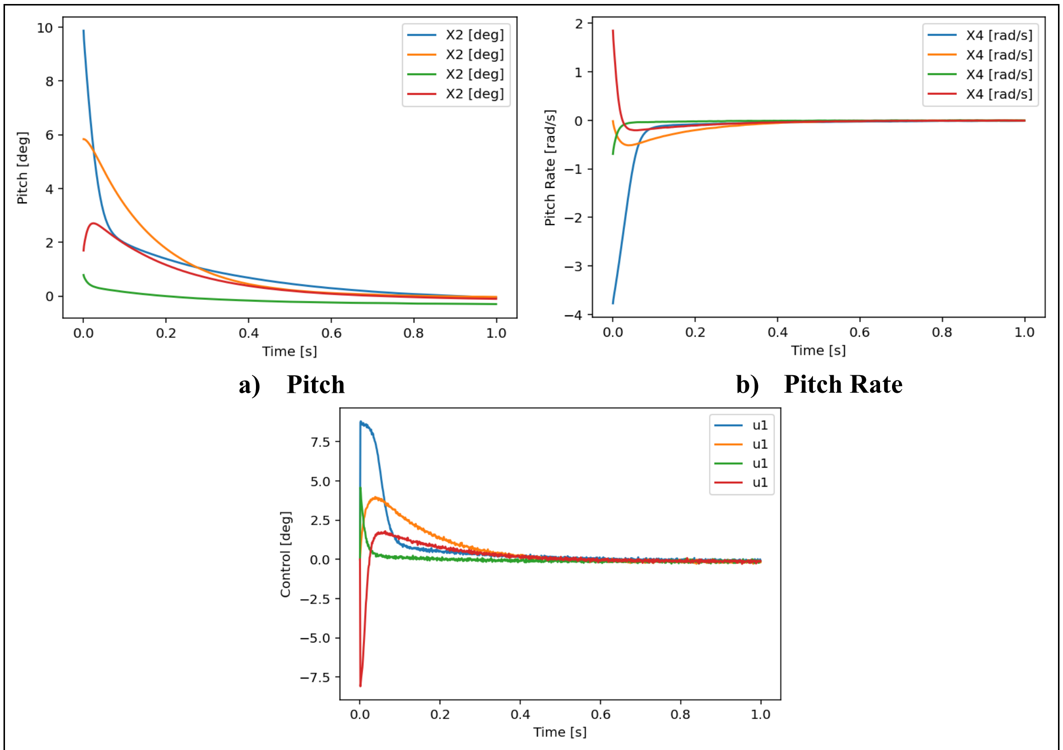


a) Pitch **b) Pitch Rate**

c) Control

Figure 1. Closed-loop response with unstable plant.

Figure 2 shows the closed-loop response using a stable plant and randomly chosen initial conditions. The agent nearly eliminates oscillation in these responses and achieves a settling time around 0.6 s.

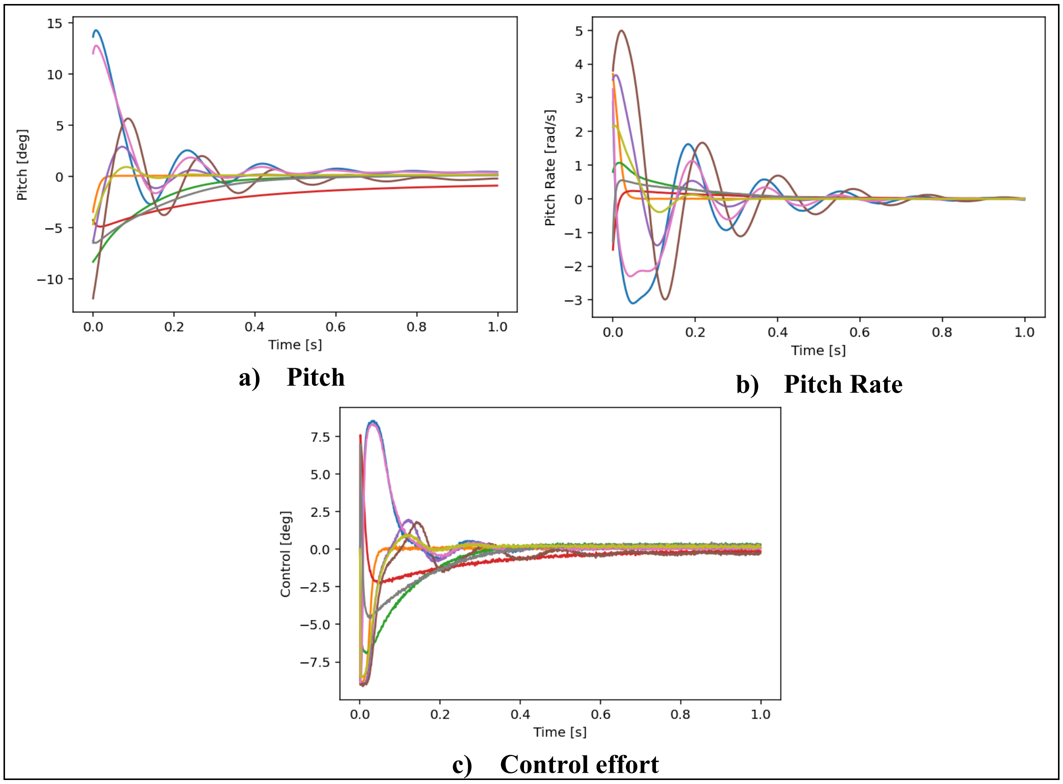


a) Pitch **b) Pitch Rate**

c) Control

Figure 2. Closed-loop response with stable plant.

Figure 3 shows closed-loop responses for 10 stable plant models. Oscillations are more pronounced when initial pitch rate exceeds 2 rad/s.

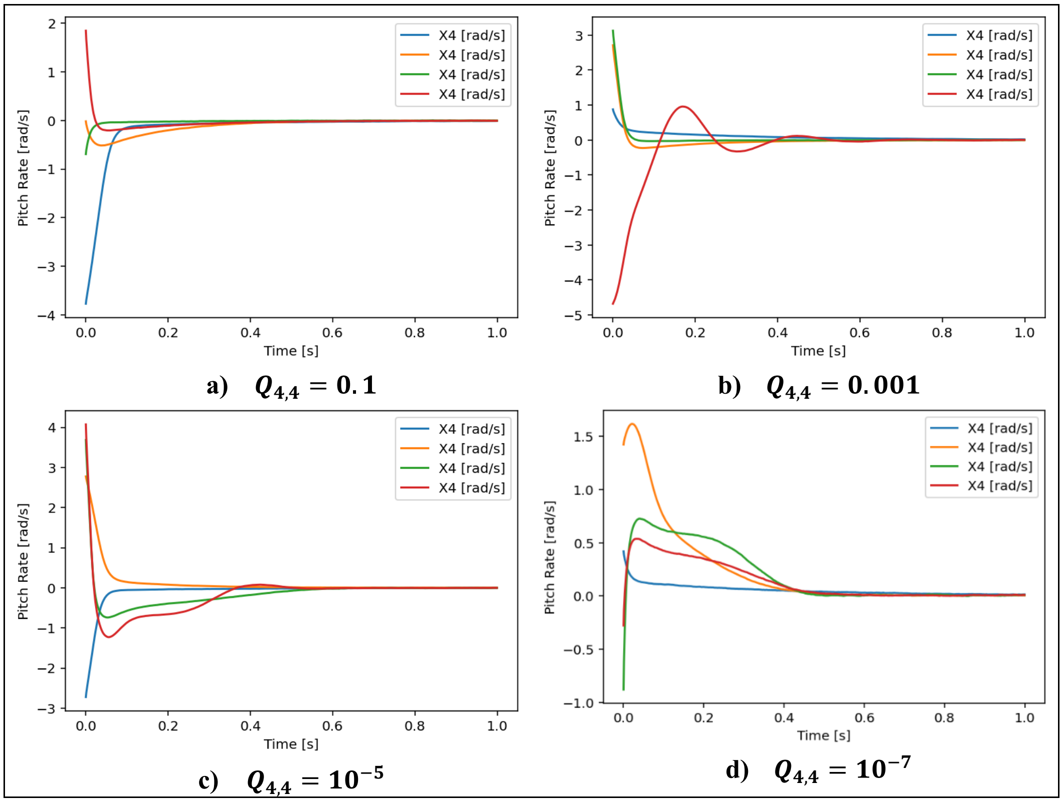


a) **Pitch** b) **Pitch Rate**

c) **Control effort**

Figure 3. Closed-loop response for 10 plant models, various initial conditions.

Next, we investigated the role of the reward function and hence $\mathbf{Q}$ and $\mathbf{R}$ in determining performance. In Figure 4, we show a trade study of closed-loop overshoot and settling time as a function of $\boldsymbol{Q_{4,4}}$, the penalty term on pitch rate. Figure 4a is a copy of Figure 2b showing the response of three stable models controlled by an agent designed with the default values. Maximum overshoot is around 25%, but the system creeps toward steady state after the first peak for a settling time exceeding 0.4 s. In Figure 4b, we show the result after reducing $\boldsymbol{Q_{4,4}}$ by 2 orders of magnitude. Pitch (x_3) and pitch rate (x_3) are both depicted. Overshoot remains around 25%, but the pitch rate quickly settles after the first peak for a settling time around 0.5 s. In Figure 4c, we repeat the design with $\boldsymbol{Q_{4,4}}$ reduced by two additional orders of magnitude. Overshoot increases to 35% and settling time extends to nearly 0.6 s. Finally, in Figure 4d, we show the result for $\boldsymbol{Q_{4,4} = 10^{-7}}$. Responses are shown for four stable plants. Overshoot is difficult to measure in the responses that appear nonlinear; however, the largest overshoot is around 75%.



a) $Q_{4,4} = 0.1$ b) $Q_{4,4} = 0.001$ c) $Q_{4,4} = 10^{-5}$ d) $Q_{4,4} = 10^{-7}$

Figure 4. Performance with varying penalty on pitch rate.

For a basis of comparison, we formed four linear-quadratic regulator (LQR) control designs using the same range of values for **Q** and **R**. With the default values of **Q**, the LQR closed-loop poles have a damping ratio of 0.0633 or an overshoot of 82%. Reducing the value of $Q_{4,4}$ to 0.001, 10^{-5}, and 10^{-7}, resulted in damping ratios of 0.0397, 0.0394, and 0.0394, respectively—all with an overshoot of about 88%. The open loop damping ratio for the design plant is 0.0378. The large penalty on control (**R** = 2) hinders LQR from moving the poles to more desirable locations.

The RL controller clearly provides much more damping than an LQR design with identical design parameters. The RL agent is, by definition, a nonlinear controller, and thus can exceed the expected performance of a linear controller such as LQR. Also, it seems that the penalties on pitch and angle of attack (both unity) provide enough incentive for the RL controller to find a solution with fast settling time and small oscillation. The hard limit on control effort ($u \leq 10°$) may have a stronger effect on the solution than the quadratic penalty of **R** = 2. No hard limit is set on u by the LQR solution.

3. Conclusion

Overall, we found the PPO algorithm computationally burdensome, and overly sensitive to the choice of hyperparameters. With the simple enhancements of seeding the actor network with a stabilizing gain, and adding an HJB residual to the cost function, a suboptimal solution could be found in a couple of hours on a quad-core laptop.

I am not impressed with RL for control of dynamical systems. Although the method is "model free" in theory, the user still needs a dynamical model to use RL. Extensive offline training in a virtual environment is necessary before deployment to any physical system. Therefore, one must write a Gym environment that includes the plant dynamics. If a user must go to the effort of forming a dynamical model of the system, then he would be well advised to save time, effort, and frustration by designing a controller using the wealth of existing control theory rather than relying on RL. A previous publication showed that similar control problems can be tackled by solving the HJB PDE with a PINN with only 50 total hidden units or about 750 total weights.[4] The attempt at RL in this report used two networks with 128 hidden units each or nearly 9000 total weights and resulted in marginal performance.

4. References

1. Gymnasium. Gymnasium documentation. 2025 Farama Foundation; c2025 [accessed 2025 Aug 22]. https://gymnasium.farama.org/index.html

2. Furfaro R, D'Ambrosio A. Increasing autonomy of aerospace systems via PINN-based solutions of HJB equation. AIAA SCITECH 2024 Forum; 2024. p. 1786.

3. D'Ambrosio A, Schiassi E, Curti F, Furfaro R. Pontryagin neural networks with functional interpolation for optimal intercept problems. Mathematics. 2021;9(9):996.

4. Burchett BT. Pontryagin neural network solutions for Hamilton–Jacobi–Bellman optimality of high-order systems. DEVCOM Army Research Laboratory (US); 2025 June. Report No.: ARL-MR-1128.

5. Scorsoglio A, D'Ambrosio A, Furfaro R. Stability certification of reinforcement learning-based control for aerospace applications. 2025 AAS/AIAA Space Flight Mechanics Meeting; 2025 Jan 19–23; Kaua'I, HI.

6. Mukherjee A, Liu J. Bridging physics-informed neural networks with reinforcement learning: Hamilton–Jacobi–Bellman proximal policy optimization (HJBPPO). arXiv; 2023. arXiv:2302.00237. https://doi.org/10.48550/arXiv.2302.00237

7. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. arXiv; 2017. arXiv:1707.06347. https://doi.org/10.48550/arXiv.1707.06347

9. Burchett BT, Bryson JT. Automatic linear model generation using Gauss process regression surrogate models for aerodynamic sensitivities. Proceedings of the AIAA SciTech 2025; 2025 6–10 Jan; Orlando, FL.

10. Åström KJ, Murray RM. Feedback systems: an introduction for scientists and engineers. Princeton University Press; 2008.

11. Franklin GF, Powell JD, Emami-Naeini A. Feedback control of dynamic systems. 3rd ed. Addison-Wesley; 1994.

12. Amartyamukherjee. PPO-PackCooling. Nikhil Barhate; c2018 [accessed 2025 Aug 22]. https://github.com/amartyamukherjee/PPO-PackCooling

8. Leake C, Mortari D. Deep theory of functional connections: a new method for estimating the solutions of partial differential equations. Mach Learn Knowl Extr. 2020;2(1):37–55.

List of Symbols, Abbreviations, and Acronyms

1-D	one-dimensional
ARL	Army Research Laboratory
CE	constrained expression
DEVCOM	U.S. Army Combat Capabilities Development Command
GAE	generalized advantage estimation
HJB	Hamilton–Jacobi–Bellman
HJBPPO	Hamilton–Jacobi–Bellman proximal policy optimization
LQR	linear-quadratic regulator
PDE	partial differential equation
PINN	physics-informed neural networks
PPO	proximal policy optimization
RL	reinforcement learning

Nomenclature

$\boldsymbol{A}$	linear state space dynamics matrix
$\boldsymbol{B}$	linear state space control matrix
L	network loss function
a_t	action at time step t
s_t	state at time step t
r_t	probability ratio
A_t	advantage at time step t
R_t	reward at time step t
π	policy (actor) network
V	value (critic) network
θ	policy network parameters
φ	value network parameters
γ	discount factor

λ	generalized advantage estimation parameter
α_i	learning rates
ϵ	clipping parameter

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
1 (PDF)	DEVCOM ARL FCDD RLB CI TECH LIB