Robust Reinforcement Learning Differential Game Guidance in Low-Thrust, Multi-Body Dynamical Environments

A Zero-Sum Reinforcement Learning Approach in Three-Body Dynamics

Ali Baniasad

Supervisor: Dr. Nobahari

Department of Aerospace Engineering

Sharif University of Technology

Outline

Research Motivation

- **Space missions** increasingly require on-board autonomous guidance systems
- **Low-thrust spacecraft** operate in complex gravitational environments
- **Three-body dynamics** (Earth-Moon CRTBP) present inherent instabilities
- **Classical control methods** struggle with:
  - Model uncertainties
  - Environmental disturbances
  - Fuel efficiency requirements
- **Need for robust, adaptive guidance** without precise dynamic models

Central Question

How can we achieve robust spacecraft guidance in uncertain environments?

Problem Statement

Research Objective

Design a robust guidance framework for low-thrust spacecraft operating in Earth-Moon three-body dynamics under uncertainties.

System Characteristics:

- State: $\mathbf{x} = [x, y, \dot{x}, \dot{y}]^T$
- Control: $|\mathbf{u}| \preceq u_{\max}$
- Dynamics: $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$

Mission Environment:

- Earth-Moon CRTBP
- Lyapunov orbit transfer
- Low-thrust propulsion

CRTBP Model and Lagrangian Points

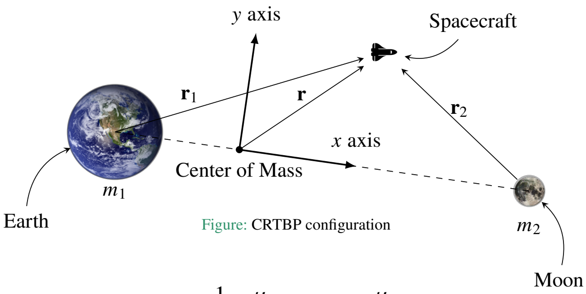


Figure: CRTBP configuration

$$\ddot{x} - 2\dot{y} = x - \frac{1-\mu}{r_1^3}(x+\mu) - \frac{\mu}{r_2^3}(x-1+\mu),$$

$$\ddot{y} + 2\dot{x} = y - \frac{1-\mu}{r_1^3}y - \frac{\mu}{r_2^3}y.$$

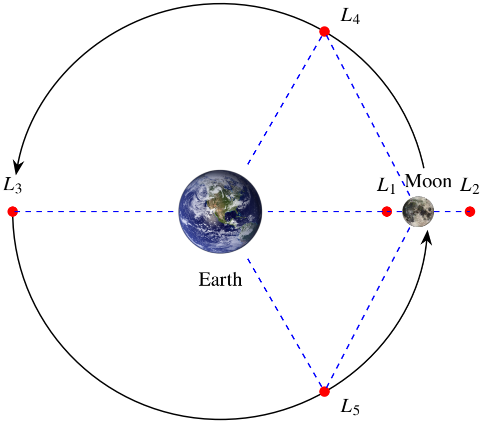


Figure: Lagrangian points

Agent Simulation in CRTBP Model

State Representation:

- Position and velocity: $\boldsymbol{s}_t = (\delta x, \delta y, \delta \dot{x}, \delta \dot{y})$
- Relative to target orbit/Lagrangian point

Action Space:

- Continuous control: $\boldsymbol{a}_t = (u_x, u_y)$
- Bounded thrust: $u_x, u_y \in [a_{Low}, a_{High}]$

Reward Function:

$$r(s,a) = r_{\text{thrust}}(a) + r_{\text{reference}}(s) + r_{\text{terminal}}(s)$$

$$r_{\text{thrust}}(a) = -k_1 \cdot |a|$$

$$r_{\text{reference}}(s) = -k_2 \cdot d(s, s_{\text{reference}})$$

$$r_{\text{terminal}}(s) = \begin{cases} +R_{\text{goal}} & \text{if } s \in S_{\text{goal}} \\ -R_{\text{fail}} & \text{if } d(s, s_{\text{ref}}) > \epsilon \\ 0 & \text{otherwise} \end{cases}$$

Table: Nondimensionalized spacecraft thrust capabilities

Abbrv.	**Spacecraft**	$f_{\mathbf{max}}$, nondim	$F_{\mathbf{max}}$
DS1	Deep Space 1	$6.94 \cdot 10^{-2}$	92.0 mN
Psyche	Psyche	$4.16 \cdot 10^{-2}$	279.3 mN
Dawn	Dawn	$2.74 \cdot 10^{-2}$	91.0 mN
LIC	Lunar IceCube	$3.28 \cdot 10^{-2}$	1.25 mN
H1	Hayabusa 1	$1.64 \cdot 10^{-2}$	22.8 mN
H2	Hayabusa 2	$1.63 \cdot 10^{-2}$	27.0 mN
s/c	Sample spacecraft	$4 \cdot 10^{-2}$	n/a

Reinforcement Learning Overview

- **Definition:** A type of machine learning where an agent learns to make decisions by taking actions in an environment to maximize cumulative reward.

- **Key Components:**
  - **Agent:** The learner or decision maker.
  - **Environment:** The external system with which the agent interacts.
  - **Actions:** Choices made by the agent to influence the environment.
  - **Rewards:** Feedback from the environment based on the agent's actions.

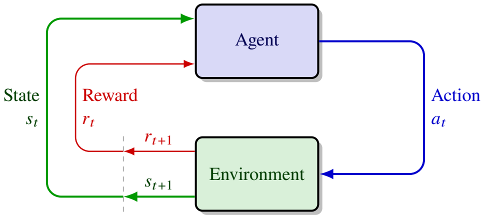


Figure: Agent-Environment Interaction Loop

State, Observations, and Actions

- **State (s):** Complete description of the environment at a given time
  - Encodes all variables needed to predict future dynamics
  - Typically hidden from the agent in real-world problems
- **Observation (o):** Information perceived by the agent
  - May be noisy or incomplete (partial observability)
  - In fully observable environments: $s = o$
  - In partially observable settings: agent must infer hidden aspects of s
- **Action Space ($\mathcal{A}$):** Set of all possible actions an agent can take
  - **Discrete:** Finite set of actions (e.g., `up, down, left, right`)
  - **Continuous:** Actions represented by real values (e.g., steering angle, force applied)

Trajectory and Reward

Definitions:

- Trajectory: sequence of states and actions the agent experiences over time.
- Reward: scalar feedback provided by the environment after taking an action.
- Return: accumulated reward over a trajectory (finite or discounted horizon).

Equations:

$$\tau = (s_0, a_0, s_1, a_1, \dots)$$

$$r_t = R(s_t, a_t, s_{t+1}) \quad \text{or} \quad r_t = R(s_t, a_t)$$

$$R(\tau) = r_1 + r_2 + \cdots + r_T = \sum_{t=0}^{T} r_t \text{ (finite horizon)}$$

$$R(\tau) = r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots = \sum_{t=0}^{\infty} \gamma^t r_t \text{ (discounted)}$$

Policy

- **Policy:** Rules that an agent uses to decide which actions to take
  - **Types:**
    - **Deterministic:** $a_t = \mu(s_t) \longrightarrow$ DDPG, TD3
    - **Stochastic:** $a_t \sim \pi(\cdot|s_t) \longrightarrow$ PPO, SAC
  - **Parameterized Policy:** Output is a function of policy parameters (neural network weights)
    - $a_t = \mu_\theta(s_t)$ or $a_t \sim \pi_\theta(\cdot|s_t)$
    - Parameters θ are optimized during learning

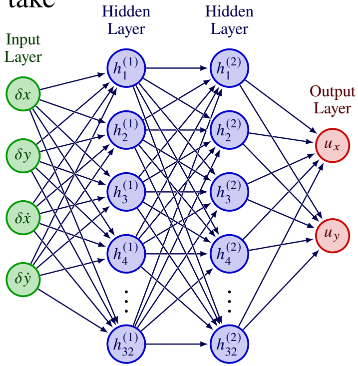


Figure: Policy Neural Network Structure

Value and Action-Value Functions

- **Value Function:** Expected return when following a policy

Value Function:

$$V^{\pi}(s) = \mathop{\mathbb{E}}_{\tau \sim \pi} [R(\tau) | s_0 = s]$$

Action-Value Function:

$$Q^{\pi}(s, a) = \mathop{\mathbb{E}}_{\tau \sim \pi} [R(\tau) | s_0 = s, a_0 = a]$$

Advantage Function:

$$A^{\pi}(s, a) = Q^{\pi}(s, a) - V^{\pi}(s)$$

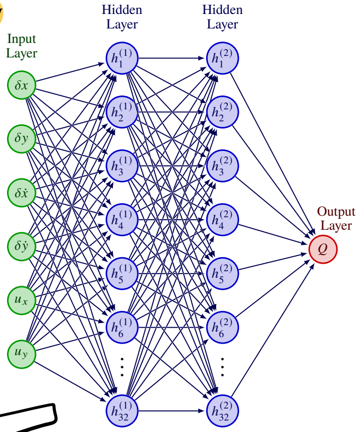


Figure: Action-Value Function Neural Network

Value Computation and Bellman Equations

Value Computation

How can we calculate the value of a state $V(s)$ and a state-action pair $Q(s,a)$?

For Policy Value Functions:

$$V^\pi(s) = \mathop{\mathrm{E}}_{\substack{a\sim\pi \\ s'\sim P}} \left[r(s,a) + \gamma V^\pi(s') \right]$$

$$Q^\pi(s,a) = r(s,a) + \gamma \mathop{\mathrm{E}}_{s'\sim P} \left[\mathop{\mathrm{E}}_{a'\sim\pi} \left[Q^\pi(s',a') \right] \right]$$

For Optimal Value Functions:

$$V^*(s) = \max_\pi V^\pi(s) \rightarrow V^*(s) = \max_a \mathop{\mathrm{E}}_{s'\sim P} \left[r(s,a) + \gamma V^*(s') \right]$$

$$Q^*(s,a) = \max_\pi Q^\pi(s,a) \rightarrow Q^*(s,a) = r(s,a) + \gamma \mathop{\mathrm{E}}_{s'\sim P} \left[\max_{a'} Q^*(s',a') \right]$$

DDPG Algorithm

Initialize: policy θ, Q-function ϕ, targets $\theta_{\text{targ}}, \phi_{\text{targ}}$, replay buffer $\mathcal{D}$
repeat
  Collect experience: $a = \text{clip}(\mu_\theta(s) + \text{noise})$, observe (s', r, d), store in $\mathcal{D}$
  Sample batch B from $\mathcal{D}$
  Compute targets: $y = r + \gamma(1-d)Q_{\phi_{\text{targ}}}(s', \mu_{\theta_{\text{targ}}}(s'))$
  Update critic: minimize $(Q_\phi(s,a) - y)^2$
  Update actor: maximize $Q_\phi(s, \mu_\theta(s))$
  Update targets: $\phi_{\text{targ}} \leftarrow \rho\phi_{\text{targ}} + (1-\rho)\phi$, same for θ
until convergence

Twin Delayed DDPG (TD3) Algorithm

```
Initialize: policy θ, Q-functions φ_1, φ_2, targets θ_targ, φ_targ,1, φ_targ,2, buffer 𝒟
repeat
    Collect experience: a = clip(μ_θ(s) + noise, a_Low, a_High)
    Store transition (s, a, r, s', d) in 𝒟
    if time to update then
        Sample batch B from 𝒟
        Compute target actions with noise: a'(s') = clip(μ_θtarg(s') + noise, a_Low, a_High)
        Compute targets: y = r + γ(1 − d) min_{i=1,2} Q_φtarg,i(s', a'(s'))
        Update Q-functions: minimize (Q_φi(s, a) − y)^2 for i = 1, 2
        Update policy: maximize Q_φ1(s, μ_θ(s))
        Update targets: φ_targ,i ← ρφ_targ,i + (1 − ρ)φ_i for i = 1, 2
        Update target policy: θ_targ ← ρθ_targ + (1 − ρ)θ
    end if
until convergence
```

Soft Actor-Critic (SAC) Algorithm

Initialize: policy θ, Q-functions ϕ_1, ϕ_2, targets $\phi_{\text{targ},1}$, $\phi_{\text{targ},2}$, buffer $\mathcal{D}$
repeat
    Collect experience: $a \sim \pi_\theta(\cdot|s)$, observe (s', r, d), store in $\mathcal{D}$
    if time to update **then**
        Sample batch B from $\mathcal{D}$
        Sample actions from policy: $\tilde{a}' \sim \pi_\theta(\cdot|s')$
        Compute targets: $y = r + \gamma(1 - d)\left(\min_{i=1,2} Q_{\phi_{\text{targ},i}}(s', \tilde{a}') - \alpha \log \pi_\theta(\tilde{a}'|s')\right)$
        Update Q-functions: minimize $(Q_{\phi_i}(s, a) - y)^2$ for $i = 1, 2$
        Sample actions using reparameterization trick: $\tilde{a}_\theta(s) \sim \pi_\theta(\cdot|s)$
        Update policy: maximize $\min_{i=1,2} Q_{\phi_i}(s, \tilde{a}_\theta(s)) - \alpha \log \pi_\theta(\tilde{a}_\theta(s)|s)$
        Update targets: $\phi_{\text{targ},i} \leftarrow \rho\phi_{\text{targ},i} + (1 - \rho)\phi_i$ for $i = 1, 2$
    end if
until convergence

Proximal Policy Optimization (PPO) Algorithm

Initialize: policy θ_0, value function ϕ_0

for $k = 0, 1, 2, \ldots$ **do**

  Collect trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in environment

  Compute rewards-to-go $\hat{R}_t$

  Compute advantage estimates $\hat{A}_t$ based on current value function V_{ϕ_k}

  Update policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|} \sum_{\tau,t} \min\Big(r_t(\theta)\hat{A}_t,\ \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\Big)$$

  where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}$ is the probability ratio

  Fit value function by minimizing:

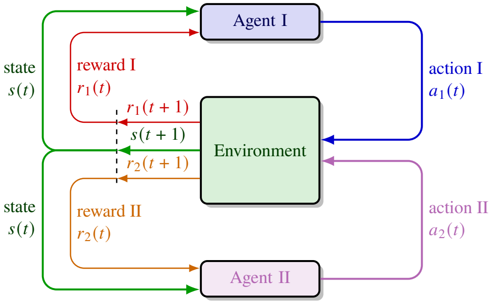
$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k|} \sum_{\tau,t} \left(V_\phi(s_t) - \hat{R}_t\right)^2$$

end for

Key Components & Definitions

Agents: Independent decision makers sharing an environment.
Policy $\pi_i(a_i|s)$: Action distribution of agent i.
Utility / Return: $V_i^{\pi}(s) = \mathbb{E}_{\pi}[\sum_t \gamma^t r_i]$.

- Single-agent RL is a special case ($n = 1$)
- Interaction types: cooperative, competitive, mixed
- Game-theoretic view clarifies stability / equilibria
- Shared state, distinct rewards and policies
- Centralized training, decentralized execution (CTDE)



Zero-Sum Games

Player 1 reward:

$$r_1(s, a_1, a_2) = -k_1|a_1| - k_2|a_2| - k_3\, d_1(s, s_{\text{ref},1}) + r_{\text{terminal},1}(s)$$

$$r_{\text{terminal},1}(s) = \begin{cases} +R_{\text{goal},1}, & s \in S_{\text{goal},1} \\ -R_{\text{fail},1}, & d_1(s, s_{\text{ref},1}) > \epsilon_1 \\ 0, & \text{otherwise} \end{cases}$$

Zero-sum property:

$$r_2(s, a_1, a_2) = -r_1(s, a_1, a_2), \qquad V_1^{(\pi_1,\pi_2)} = -V_2^{(\pi_1,\pi_2)}, \quad Q_1 = -Q_2$$

Minimax optimality:

$$V_1^*(s) = \max_{\pi_1} \min_{\pi_2} V_1^{(\pi_1,\pi_2)}(s) = \min_{\pi_2} \max_{\pi_1} V_1^{(\pi_1,\pi_2)}(s)$$

From Single-Agent to Zero-Sum Robustness

- Lift environment: $(s, a) \to (s, a_1, a_2)$
- Critic learns $Q_1(s, a_1, a_2)$; $Q_2 = -Q_1$
- Policy updates:

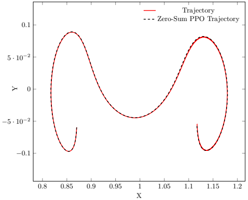
$$\max_{\theta_1} \mathbb{E}[Q_1], \quad \max_{\theta_2} \mathbb{E}[-Q_1]$$

- Stabilization: target networks, entropy (SAC), delay (TD3), clipping (PPO)
- Outcome: robust guidance via adversarial curriculum

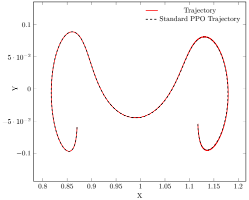
Single-Agent vs. Zero-Sum MARL

Comparison:

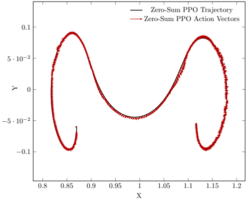
- Both single-agent RL and zero-sum MARL achieve the task.
- Single-agent remains effective but less robust to disturbances.
- With or without an adversary, policies remain robust.



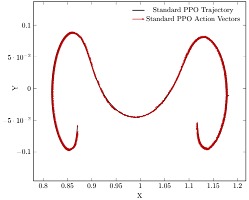
(a) Zero-Sum MARL Trajectory



(b) Single-Agent Trajectory



(c) Zero-Sum MARL Control

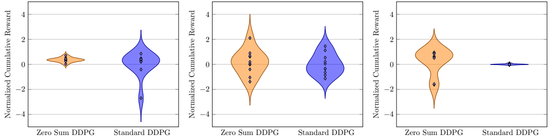


(d) Single-Agent Control

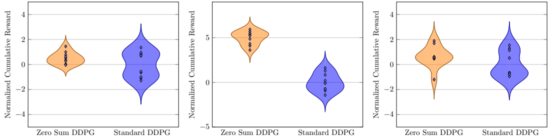
Robustness Scenario Specification

- **Random Init:** $x_0 \leftarrow x_0 + \mathcal{N}(0, 0.1^2)$
- **Actuator Disturbance:** $u_t \leftarrow u_t + \mathcal{N}(0, 0.05^2)$; (sensor additive) $y_t \leftarrow y_t + \mathcal{N}(0, 0.02^2)$
- **Model Mismatch:** $\theta \leftarrow \theta + \mathcal{N}(0, 0.05^2)$
- **Partial Observability:** mask 50% $\rightarrow m_t^{(i)} \sim \mathrm{Bern}(0.5)$, $y_t \leftarrow y_t \circ m_t$
- **Sensor Noise (multiplicative):** $y_t \leftarrow y_t \circ \left(1 + \mathcal{N}(0, 0.05^2)\right)$
- **Time Delay:** buffer length 10, z $u_t^{\text{applied}} \leftarrow u_{t-10} + \mathcal{N}(0, 0.05^2)$
- **Notes:**
  - All scenarios evaluated independently.
  - Zero-sum agents trained jointly once.

Robustness Evaluation: DDPG vs. MA-DDPG



(a) Random Initial Conditions (b) Actuator Disturbance (c) Model Mismatch

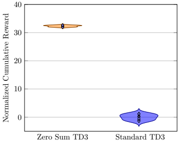


(d) Partial Observation (e) Sensor Noise (f) Time Delay

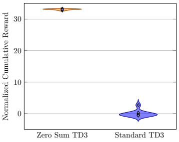
Scenario	Cumulative Reward		Path Error Sum	
	DDPG	MA-DDPG	DDPG	MA-DDPG
Random Initial Conditions	-4.17	-3.63	0.40	0.63
Actuator Disturbance	-1.93	-1.96	7.56	7.94
Model Mismatch	-3.24	-2.70	0.70	0.76
Partial Observation	-3.28	-2.89	0.68	0.75
Sensor Noise	-1.07	-0.47	0.10	0.15
Time Delay	-3.20	-1.91	1.74	2.43

Scenario	Control Effort Sum		Failure Probability	
	DDPG	MA-DDPG	DDPG	MA-DDPG
Random Initial Conditions	5.52	5.60	1.00	1.00
Actuator Disturbance	5.60	5.59	0.90	0.30
Model Mismatch	5.29	5.57	1.00	1.00
Partial Observation	5.57	5.57	0.60	0.80
Sensor Noise	5.51	5.54	0.00	0.00
Time Delay	5.61	5.61	0.70	0.70

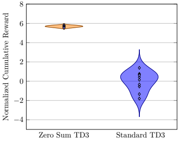
Robustness Evaluation: TD3 vs. MA-TD3



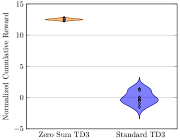
(a) Random Initial Conditions



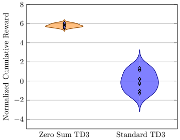
(b) Actuator Disturbance



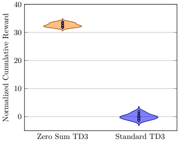
(c) Model Mismatch



(d) Partial Observation



(e) Sensor Noise

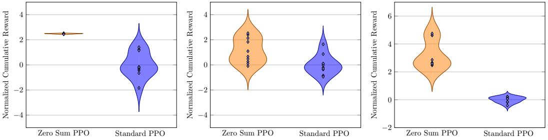


(f) Time Delay

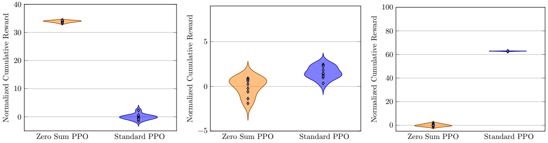
Scenario	Cumulative Reward		Path Error Sum	
	TD3	MA-TD3	TD3	MA-TD3
Random Initial Conditions	-2.95	-0.26	0.39	0.14
Actuator Disturbance	0.56	0.73	0.02	0.00
Model Mismatch	-4.73	-3.30	0.47	0.73
Partial Observation	0.21	0.71	0.02	0.01
Sensor Noise	-0.08	-2.93	0.11	3.19
Time Delay	0.55	0.67	0.01	0.01

Scenario	Control Effort Sum		Failure Probability	
	TD3	MA-TD3	TD3	MA-TD3
Random Initial Conditions	5.05	4.57	1.00	0.30
Actuator Disturbance	3.06	2.66	0.00	0.00
Model Mismatch	5.53	5.41	1.00	1.00
Partial Observation	4.09	3.18	0.00	0.00
Sensor Noise	5.46	5.50	0.00	1.00
Time Delay	4.57	4.57	0.00	0.00

Robustness Evaluation: PPO vs. MA-PPO



(a) Random Initial Conditions (b) Actuator Disturbance (c) Model Mismatch

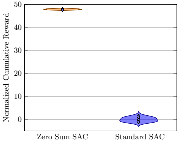


(d) Partial Observation (e) Sensor Noise (f) Time Delay

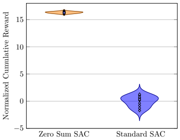
Scenario	Cumulative Reward		Path Error Sum	
	PPO	MA-PPO	PPO	MA-PPO
Random Initial Conditions	-1.85	0.46	0.22	0.14
Actuator Disturbance	-1.97	-1.91	8.33	7.50
Model Mismatch	0.46	0.30	0.07	0.08
Partial Observation	-3.60	-1.81	2.34	2.06
Sensor Noise	0.52	0.48	0.13	0.15
Time Delay	0.58	-2.44	0.03	2.49

Scenario	Control Effort Sum		Failure Probability	
	PPO	MA-PPO	PPO	MA-PPO
Random Initial Conditions	1.55	1.98	0.70	0.00
Actuator Disturbance	2.59	3.42	1.00	1.00
Model Mismatch	0.90	1.13	0.00	0.00
Partial Observation	1.06	2.15	1.00	1.00
Sensor Noise	1.22	2.08	0.00	0.00
Time Delay	2.43	2.56	0.00	1.00

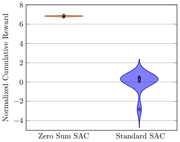
Robustness Evaluation: SAC vs. MA-SAC



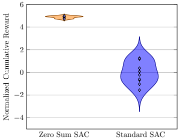
(a) Random Initial Conditions



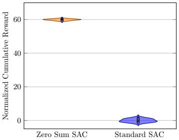
(b) Actuator Disturbance



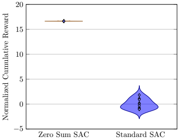
(c) Model Mismatch



(d) Partial Observation



(e) Sensor Noise

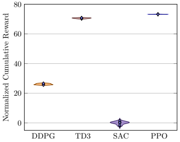


(f) Time Delay

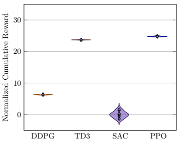
Scenario	Cumulative Reward		Path Error Sum	
	SAC	MA-SAC	SAC	MA-SAC
Random Initial Conditions	-4.69	-2.98	0.29	0.26
Actuator Disturbance	-1.95	-1.93	8.02	7.72
Model Mismatch	-4.89	-4.35	0.38	0.26
Partial Observation	-3.63	-0.44	1.95	0.07
Sensor Noise	-0.89	0.12	0.12	0.12
Time Delay	-4.14	-0.05	1.87	0.01

Scenario	Control Effort Sum		Failure Probability	
	SAC	MA-SAC	SAC	MA-SAC
Random Initial Conditions	2.15	1.37	1.00	1.00
Actuator Disturbance	3.26	3.09	1.00	1.00
Model Mismatch	1.99	1.16	1.00	1.00
Partial Observation	2.32	1.99	1.00	0.00
Sensor Noise	2.10	1.86	0.00	0.00
Time Delay	2.22	1.25	1.00	0.00

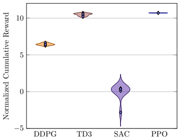
Single-Agent RL: Return and Error Distributions



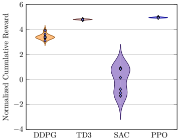
(a) Random Initial Conditions



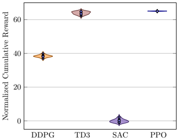
(b) Actuator Disturbance



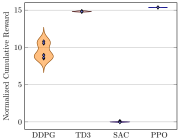
(c) Model Mismatch



(d) Partial Observation



(e) Sensor Noise

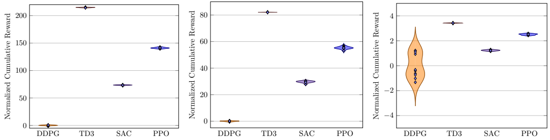


(f) Time Delay

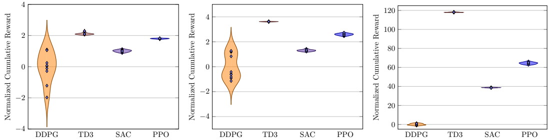
Scenario	Cumulative Return				Path Error Sum			
	DDPG	PPO	SAC	TD3	DDPG	PPO	SAC	TD3
Random Initial Conditions	-0.27	**0.61**	-0.76	0.56	3.30	2.56	8.06	**0.72**
Actuator Disturbance	-0.38	**0.61**	-0.72	0.55	3.74	2.58	7.91	**0.77**
Model Mismatch	-0.84	**0.58**	-2.98	0.51	10.87	3.06	17.12	**1.09**
Partial Observation	-0.88	**0.36**	-3.65	0.23	8.18	3.34	15.47	**1.77**
Sensor Noise	-0.85	**0.58**	-2.90	0.52	11.04	3.08	16.81	**1.02**
Time Delay	-0.76	**0.61**	-2.98	0.48	8.95	2.27	15.70	**0.81**

Scenario	Control Effort Sum				Failure Probability			
	DDPG	PPO	SAC	TD3	DDPG	PPO	SAC	TD3
Random Initial Conditions	5.11	**0.77**	1.76	3.31	**0.00**	**0.00**	0.00	**0.00**
Actuator Disturbance	4.89	**0.77**	1.71	3.07	**0.00**	**0.00**	0.00	**0.00**
Model Mismatch	5.48	**0.86**	2.37	4.32	**0.00**	**0.00**	1.00	**0.00**
Partial Observation	5.37	**1.03**	2.33	4.10	**0.00**	**0.00**	1.00	**0.00**
Sensor Noise	5.48	**0.86**	2.37	4.30	**0.00**	**0.00**	1.00	**0.00**
Time Delay	5.51	**0.76**	2.11	5.12	**0.00**	**0.00**	1.00	**0.00**

Zero-Sum MARL: Return and Error Distributions



(a) Random Initial Conditions (b) Actuator Disturbance (c) Model Mismatch



(d) Partial Observation (e) Sensor Noise (f) Time Delay

Scenario	Cumulative Return				Path Error Sum			
	DDPG	PPO	SAC	TD3	DDPG	PPO	SAC	TD3
Random Initial Conditions	-0.41	0.34	-0.02	**0.74**	4.42	4.30	4.02	**1.22**
Actuator Disturbance	-0.44	0.35	-0.02	**0.73**	4.39	4.38	4.01	**1.26**
Model Mismatch	-0.63	0.38	-0.13	**0.75**	8.85	3.57	4.78	**1.25**
Partial Observation	-1.52	0.40	-0.44	**0.71**	9.65	2.44	5.17	**1.09**
Sensor Noise	-0.60	0.37	-0.12	**0.75**	9.12	3.58	4.66	**1.25**
Time Delay	-1.19	0.17	-0.05	**0.67**	6.73	4.53	4.12	**1.21**

Scenario	Control Effort Sum				Failure Probability			
	DDPG	PPO	SAC	TD3	DDPG	PPO	SAC	TD3
Random Initial Conditions	5.40	**1.15**	1.34	2.76	**0.00**	**0.00**	0.00	**0.00**
Actuator Disturbance	5.08	**1.11**	1.23	2.66	**0.00**	**0.00**	0.00	**0.00**
Model Mismatch	5.55	**1.51**	2.09	3.38	**0.00**	**0.00**	1.00	**0.00**
Partial Observation	5.46	**1.50**	2.00	3.20	**0.00**	**0.00**	1.00	**0.00**
Sensor Noise	5.54	**1.52**	2.08	3.38	**0.00**	**0.00**	1.00	**0.00**
Time Delay	5.48	**1.25**	**1.25**	4.57	**0.00**	**0.00**	1.00	**0.00**

Summary of Principal Findings

- Zero-sum MARL framing improves worst-case orbital maintenance robustness.
- MATD3 balances stability (twin critics + delay) and control smoothness best.
- Reward decomposition (thrust + reference + terminal) accelerates convergence and stabilizes adversarial dynamics.
- Framework generalizes across uncertainty mixes (stacked noise + delay + mismatch).

Conclusion: Adversarial co-training yields resilient guidance without explicit disturbance models.

Thanks for your attention

DDPG Parameters

Steps / epoch	30k	Epochs	100
Buffer size	10^6	Discount γ	0.99
Polyak τ	0.995	Actor LR	1×10^{-3}
Critic LR	1×10^{-3}	Batch size	1024
Start policy steps	5k	Update start	1k
Update interval	2k	Action noise	0.1
Max episode len	6k	Device	CUDA
Nets (A/C)	(32,32)	Activation	ReLU

Table: DDPG hyperparameters

A/C = Actor/Critic; LR = learning rate; len = length.

TD3 Parameters

Steps / epoch	30k	Epochs	100
Buffer size	10^6	Discount γ	0.99
Polyak τ	0.995	Actor LR	1×10^{-3}
Critic LR	1×10^{-3}	Batch size	1024
Start policy steps	5k	Update start	1k
Update interval	2k	Target noise	0.2
Noise clip	0.5	Policy delay	2
Max episode len	30k	Nets (A/C)	(32,32)

Table: TD3 hyperparameters

A/C = Actor/Critic; LR = learning rate; len = length.

SAC Parameters

Steps / epoch	30k	Epochs	100
Buffer size	10^6	Discount γ	0.99
Polyak τ	0.995	LR (all)	1×10^{-3}
Alpha init	0.2	Batch size	1024
Start policy steps	5k	Update start	1k
Updates / step	10	Update interval	2k
Test episodes	10	Max episode len	30k
Nets (A/C)	(32,32)	Activation	ReLU

Table: SAC hyperparameters

A/C = Actor/Critic; LR = learning rate; len = length.

PPO Parameters

Steps / epoch	30k	Epochs	100
Discount γ	0.99	Clip ratio	0.2
Policy LR	3×10^{-4}	Value LR	1×10^{-3}
Policy iters	80	Value iters	80
Nets (Actor)	(32,32)	Nets (Critic)	(32,32)
Activation	ReLU	Batch (mini)	(derived)

Table: PPO hyperparameters

A/C = Actor/Critic; LR = learning rate; len = length.

Training Procedure (Summary)

1. Collect initial random experience (fill replay / buffer).
2. Loop: act, store (s, a, r, s', d), update (per algo rules).
3. Target networks: Polyak averaging (τ).
4. TD3: twin critics + delayed policy + target smoothing.
5. SAC: entropy term, adaptive temperature (if enabled).
6. PPO: clipped surrogate objective, on-policy batches.
7. Stability: normalization, gradient clipping (if needed), fixed seeds.

Nash Equilibrium

A policy profile $\pi^* = (\pi_1^*, \ldots, \pi_n^*)$ is Nash if:

$$V_i^{(\pi_i^*, \pi_{-i}^*)}(s) \;\geq\; V_i^{(\pi_i, \pi_{-i}^*)}(s) \quad \forall \pi_i,\ \forall i$$

Implications:

- No unilateral profitable deviation
- In zero-sum 2-player games value is unique
- Solution concepts guide stable MARL training