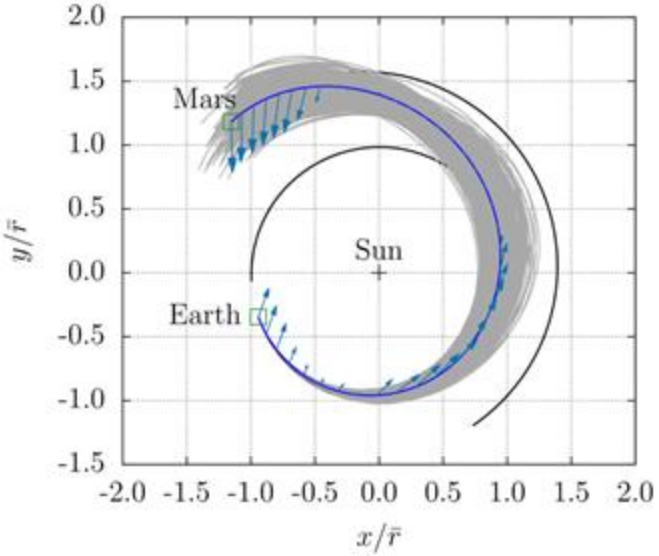
Reinforcement Learning for Robust Trajectory Design of Interplanetary Missions

By Alessandro Zavoli and Lorenzo Federici



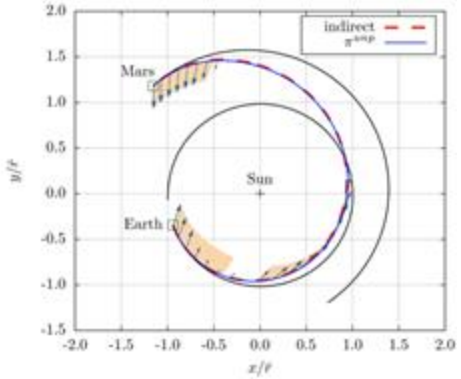
Or basically…

"Teaching a spacecraft to fly itself when things are not what they seem!"

Daniel Huterer Prats

September 4, 2025

Robust Interplanetary Trajectories



- **Why is this problem relevant?**
  - We want to conduct valuable science in the most resource efficient manner possible

    → choosing smaller satellites with cheaper equipment…
- **What does robust mean?**
  - Robust to uncertainties in position & velocity, incorrect maneuver execution or completely missed thrust events etc…

    → the stochastic nature of these uncertainties and errors makes planning very challenging
- **Other related trajectory research that inspired this work or was inspired by it**
  - Real-Time Optimal Control via Deep Neural Networks: Study on Landing Problems (Carlos Sánchez-Sánchez and Dario Izzo), cited 226 times
  - Six degree-of-freedom body-fixed hovering over unmapped asteroids via LIDAR altimetry and reinforcement meta-learning (B. Gaudet, R. Linares, R. Furfaro), cited 44 times
  - Optimization of multi-gravity assist trajectories using genetic algorithms (My Master Thesis)

Motivation & Context

- **We are now in a Micro-spacecraft era:**
  - PROCYON & MarCO showed deep-space CubeSats are feasible
  - but with low-TRL components → navigation errors, control execution errors, and missed thrust events (MTEs) are more frequent
- As low-cost mission are becoming more popular → design needs to account for less reliable equipment
- **Classical tools:**
  - indirect methods: based on Pontryagin's principle
  - direct methods: collocation
  - Model Predictive Control: convexification

→ **nonconvex constraints can hinder robustness when uncertainty bites.**

- **Can deep RL handle the 'ugly' parts?**

Traditional vs “G&CNet” approach for robustness

Traditionally

- Engineers check and improve the trajectory robustness to uncertainties after designing the trajectory
- Time-consuming iterative procedures → bring to suboptimal solutions and overconservative margins
- These large margins and hardware redundant design is no longer aligned with the design philosophy of cheap and fast micro-spacecraft missions

Now → Guidance and Control Network

- deep neural network running onboard that provides real-time guidance and control functionalities to the spacecraft

Main Implementations

Behavioral cloning: given a set of “expert” trajectories from an expert the network is trained to clone the observed “expert” behavior

Reinforcement learning: learning from experience rather than from expert demonstrations

Contributions of the paper

- Treat problem as MDP and train using PPO:
  - **maps uncertain observations → Δv**
- **Shows promising robustness for various types of uncertainties and errors** (tested through 1000 MC runs):
- **Handles several uncertainty types: state** (unmodeled accel), **observation** (nav noise), **control** (thrust direction/magnitude errors), and **random MTEs**
  - In the case of MTEs over several steps even the RL policy starts to fail often (SR ≈ 62.8% vs 28.2% using π^{unp})
- Trained **policy also performs well in comparison to traditional (optimal) methods in the deterministic case** without any uncertainties or error
- RL-training took long on the ground but in execution on-board it is very fast!

→ RL-trained policy present promising 'robust solutions' for in-space applications but more work needs to be done…

Problem Setup

- **A robust trajectory from Earth Mars is to be flown with some fixed terminal constraints**
- 3D-orbit (considering only Sun-Keplerian dynamics), **time-fixed Earth→Mars rendezvous**

Model Dynamics:

$$\begin{bmatrix} \boldsymbol{r}_{k+1} \\ \boldsymbol{v}_{k+1} \\ m_{k+1} \end{bmatrix} = f_{\text{det}}(\boldsymbol{s}_k, \boldsymbol{a}_k) = \begin{bmatrix} \hat{f}_k \boldsymbol{r}_k + \hat{g}_k(\boldsymbol{v}_k + \Delta\boldsymbol{v}_k) \\ \dot{\hat{f}}_k \boldsymbol{r}_k + \dot{\hat{g}}_k(\boldsymbol{v}_k + \Delta\boldsymbol{v}_k) \\ m_k \exp(-|\Delta\boldsymbol{v}_k|/u_{\text{eq}}) \end{bmatrix}$$

Adding state uncertainties

$$\begin{bmatrix} \boldsymbol{r}_{k+1} \\ \boldsymbol{v}_{k+1} \\ m_{k+1} \end{bmatrix} = f(\boldsymbol{s}_k, \boldsymbol{u}_k, \boldsymbol{\omega}_{s,k+1}) = f_{\text{det}}(\boldsymbol{s}_k, \boldsymbol{u}_k) + \begin{bmatrix} \delta\boldsymbol{r}_{k+1} \\ \delta\boldsymbol{v}_{k+1} \\ 0 \end{bmatrix}$$

Uncertainties Modeled:

- **State:** additive Gaussian on $[r, v]$
- **Observation (navigation):** additive Gaussian on measured $[r, v]$
- **Control:** small-angle rotation $(\delta\phi, \delta\theta, \delta\psi)$ + magnitude error δu
- **Missed Thrust Event:** single- or multi-step MTE

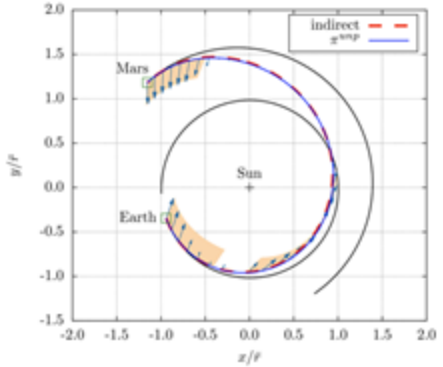
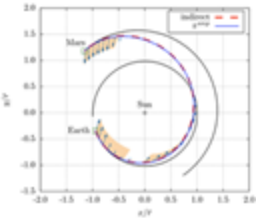


Table 1 Problem data

Variable	Value
t_f, days	358.79
m_0, kg	1,000
$T_{\max}$, N	0.50
u_{eq}, km/s	19.6133
$\mu_\odot$, km^3/s^2	132,712,440,018
$\boldsymbol{r}_\oplus$, km	$[-140{,}699{,}693, -51{,}614{,}428, 980]^T$
$\boldsymbol{v}_\oplus$, km/s	$[9.7746, -28.0783, 4.3377 \times 10^{-4}]^T$
$\boldsymbol{r}_\sigma$, km	$[-172{,}682{,}023, 176{,}959{,}469, 7{,}948{,}912]^T$
$\boldsymbol{v}_\sigma$, km/s	$[-16.4274, -14.8605, 9.2149 \times 10^{-2}]^T$

Approach



POMDP Setup:

$$s_{k+1} = f(s_k, u_k, \omega_{s,k+1}) \quad (1)$$

$$o_k = h(s_k, t_k, \omega_{o,k}) \quad (2)$$

$$a_k = \pi(o_k) \quad (3)$$

$$u_k = g(a_k, \omega_{a,k}) \quad (4)$$

Transition:
deterministic from gravity dynamics model

Discount Factor: gamma = 0.9999

- **Stochastic MDP:**
  Actions are **impulsive Δv** with N=40 steps (~9 days/step).
- **Reward (to maximize final mass under constraints):**

$$R_k = -\mu_k - \lambda_{e_u} e_{u,k-1} - \lambda_{e_s} \max\{0, e_{s,k} - \varepsilon\}$$

Fuel penalty

Violation of max Δv magnitude

Final position and velocity error

Approach

$$\theta^* = \arg\max_\theta J(\theta) = \arg\max_\theta \left(\mathop{\mathbb{E}}_{\tau\sim\pi_\theta} \left[\sum_{k=0}^{N-1} \gamma^k R_k \right] \right)$$

$$\nabla_\theta J(\theta) = \mathop{\mathbb{E}}_{\tau\sim\pi_\theta} \left[\sum_{k=0}^{N-1} \nabla_\theta \log \pi_\theta(\boldsymbol{a}_k|\boldsymbol{s}_k) \hat{A}_k \right]$$

Advantage function

$$A_k^{\pi_\theta}(\boldsymbol{s},\boldsymbol{a}) = Q_k^{\pi_\theta}(\boldsymbol{s},\boldsymbol{a}) - V_k^{\pi_\theta}(\boldsymbol{s})$$

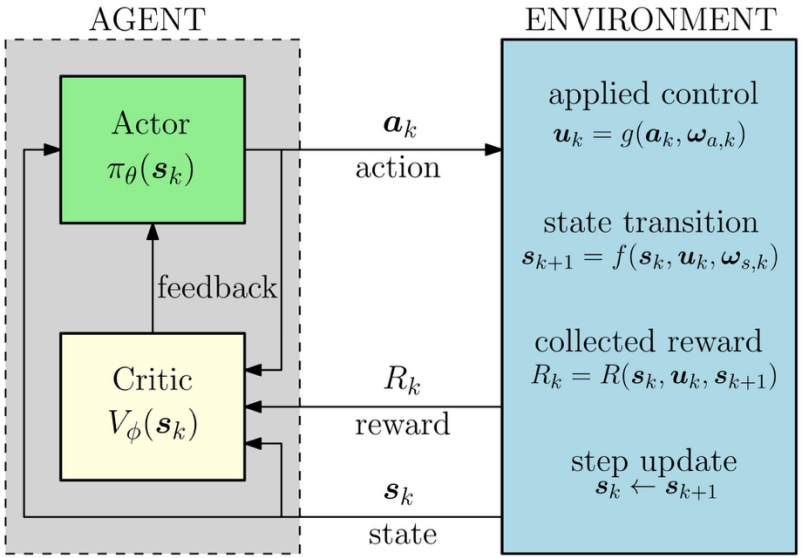
$$J^{\text{clip}}(\theta) = \mathop{\mathbb{E}}_{\tau\sim\pi_\theta} \left[\frac{1}{N} \sum_{k=0}^{N-1} \min(\tilde{r}_k(\theta)\hat{A}_k, \text{clip}(\tilde{r}_k(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_k) \right]$$

$$\tilde{r}_k(\theta) = \frac{\pi_\theta(\boldsymbol{a}_k|\boldsymbol{s}_k)}{\pi_{\theta_{\text{old}}}(\boldsymbol{a}_k|\boldsymbol{s}_k)}$$

Probability ratio

$$J^{\text{ppo}}(\theta) = J^{\text{clip}}(\theta) - c_1 H(\theta) + c_2 S(\theta)$$

Error term H Entropy term S



What made it work (implementation "knobs")

- **Nondimensionalization:**

$$\bar{r} = 149.6 \times 10^6 \text{ km}$$
$$\bar{v} = \sqrt{\mu_\odot / \bar{r}}$$
$$\bar{m} = m_0$$

- **Network: 3 hidden layers sized…interestingly**

$o_k = h(s_k, t_k, \omega_{o,k})$ → $h_1 = 10$ → $h_2 = \sqrt{h_1 h_3}$ → $h_3 = 10 * n_o$

- **Linear decay for α and clip:** $\alpha = \alpha_0(1 - t/T) \quad \epsilon = \epsilon_0(1 - t/T)$
- **Hyperparameter search** via "Tree-Structured Parzen Estimator"
  - Something similar to cross-entropy optimization from DMU
- **Constraint handling: similar to "curriculum learning"**

→ **make the environment harder as you train**

Table 2 PPO hyperparameters

Hyperparameter	Symbol	Value	Eligibility interval
Discount factor	γ	0.9999	$\{0.9, 0.95, 0.98, 0.99, 0.999, 0.9999, 1\}$
GAE factor	λ	0.99	$\{0.8, 0.9, 0.92, 0.95, 0.98, 0.99, 1.0\}$
Initial learning rate	α_0	2.5×10^{-4}	$[1 \times 10^{-5}, 1]$
Initial clip range	ϵ_0	0.3	$\{0.1, 0.2, 0.3, 0.4\}$
Value function coefficient	c_1	0.5	$\{0.5, 1.0, 5.0\}$
Entropy coefficient	c_2	4.75×10^{-8}	$[1 \times 10^{-8}, 0.1]$
Number of SGA epochs per update	n_{opt}	30	$\{1, 5, 10, 20, 30, 50\}$

$$R_k = -\mu_k - \lambda_{e_u} e_{u,k-1} - \lambda_{e_s} \max\{0, e_{s,k} - \varepsilon\}$$

$$\varepsilon(t) = \begin{cases} \varepsilon_0 & t \le T/2 \\ \varepsilon_\infty & t > T/2 \end{cases}$$

Final position and velocity error

This was a significant performance improver!

Their code can be found here:
https://github.com/LorenzoFederici/RobustTrajectoryDesignbyRL

Metrics & test scenarios

- **Training scenarios**:
  - Deterministic → Optimal Trajectory Design
  - Robust Trajectory Design (under uncertainties or control errors or MTE/s)
    - One policy for each scenario: π^{st}, π^{obs}, π^{ctr}, $\pi^{\text{mte,1}}$, and $\pi^{\text{mte,2}}$
- **Monte Carlo evaluation: 1,000 episodes** per scenario
  - success if terminal error ≤ ε; report mean/std of final mass and errors
  - report mean/std of final mass and errors

Table 3 Parameters of the different uncertainty models

Uncertainty	Parameter	Value
State	$\sigma_{r,s}$, km	1.0
	$\sigma_{v,s}$, km/s	0.05
Observation	$\sigma_{r,o}$, km	1.0
	$\sigma_{v,o}$, km/s	0.05
Control	σ_{ϕ}, deg	1.0
	σ_{ϑ}, deg	1.0
	σ_{ψ}, deg	1.0
	σ_{u}	0.05
Single-step MTE	p_{mte}	0
	n_{mte}	1
Multiple-step MTE	p_{mte}	0.1
	n_{mte}	3

Results - robust policies & what they trade

Final mass 606.34 kg vs 603.91 kg from an indirect (optimal) method

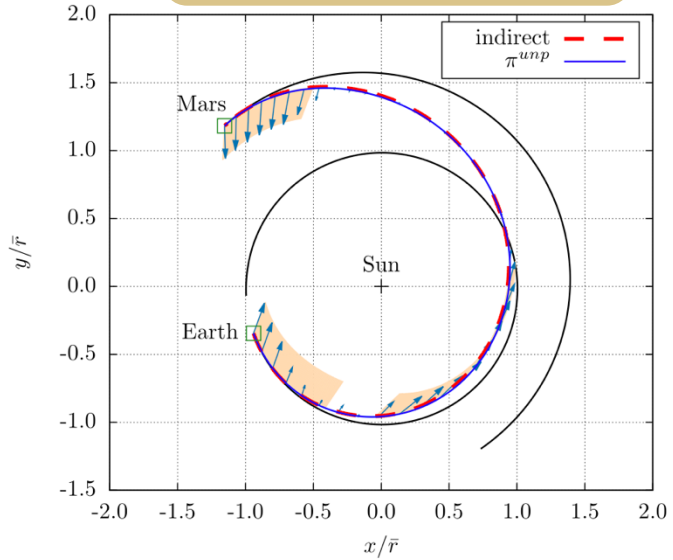


Fig. 2 Optimal Earth–Mars trajectories in the deterministic scenario.

Trained policies tested in the unperturbed environment

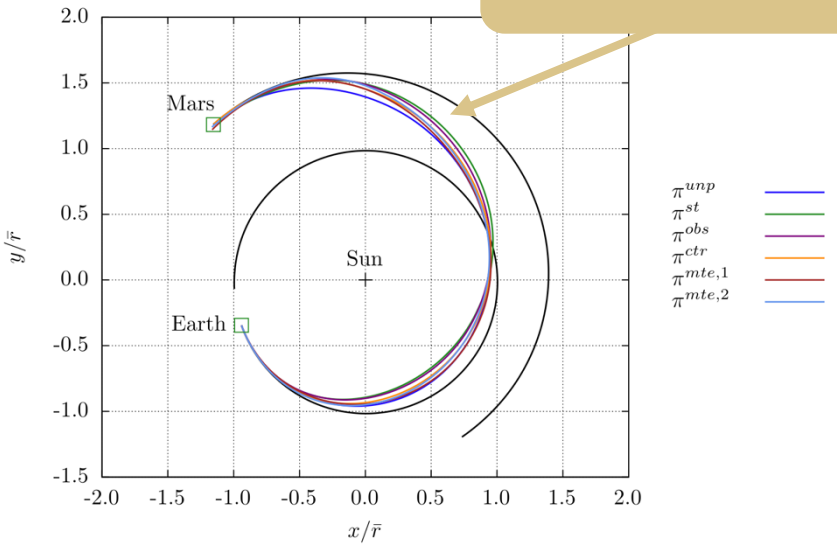


Fig. 3 Earth–Mars trajectories corresponding to different robust policies.

Results - robust policies & what they trade

"Bang-off-bang" pattern in unperturbed case

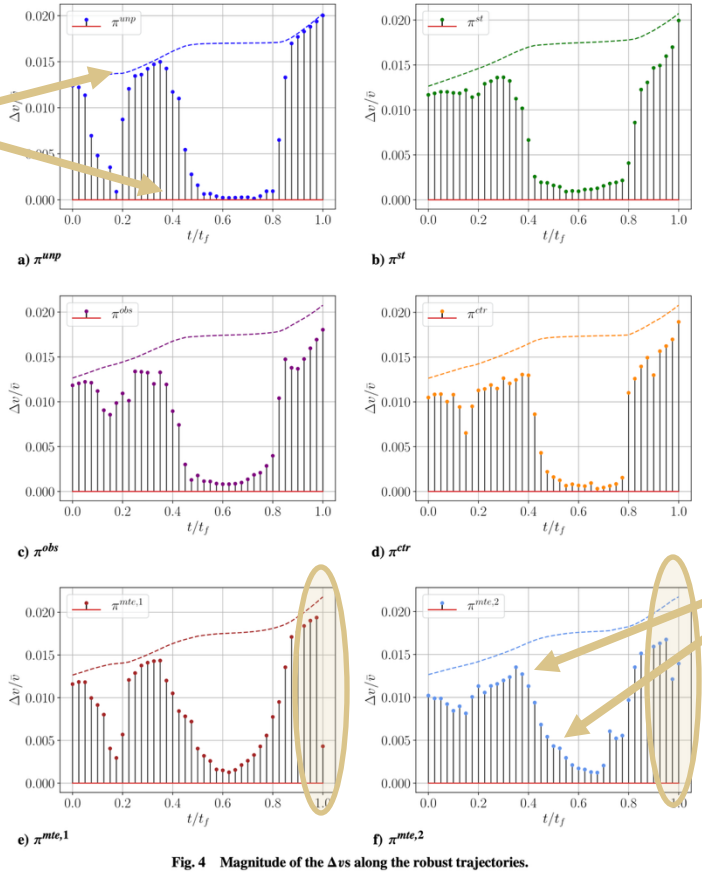


Fig. 4 Magnitude of the Δvs along the robust trajectories.

more conservative behaviour

Policy tries to achieve close to terminal conditions before final step (in anticipation of MTE)

Results - robust policies & what they trade

To the policy they are effectively the same except when meeting terminal constraints

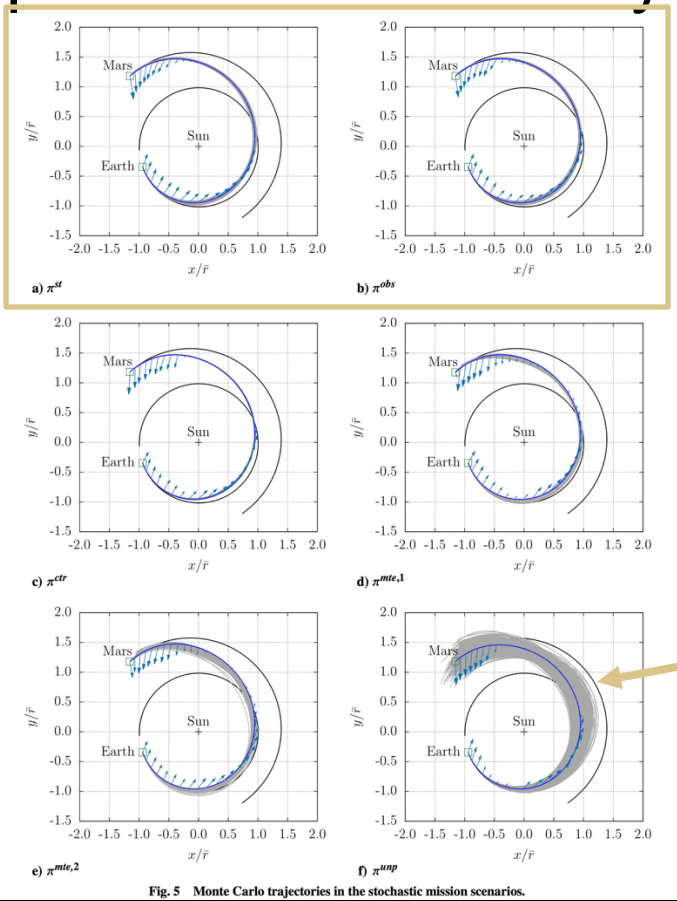


Fig. 5 Monte Carlo trajectories in the stochastic mission scenarios.

MTE shows largest deviations (unable to maneuver for at least 18 days)

What happens when you train without uncertainties and test with

Results - robust policies & what they trade

Why is there a trend in terminal constraint error?

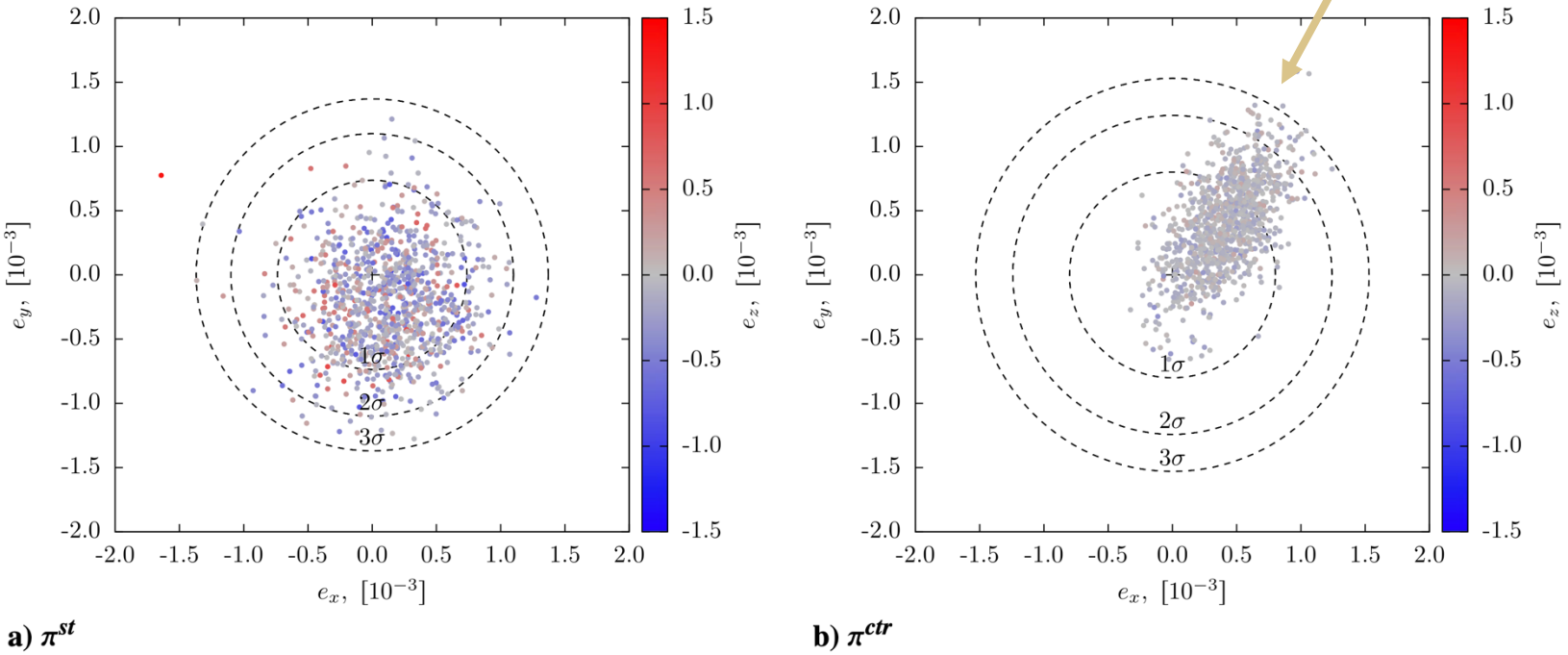


Fig. 6 Distribution of the position errors at Mars encounter for policies π^{st} and π^{ctr}.

Results - robust policies

- **Reference robust trajectories:**
  - Approach Mars slightly earlier to leave room for corrections
  - Thrust magnitudes are spread out and below Δv maxima
  - Increasing prop usage **~<3%** for state/obs/control noise

- **In deterministic case performs near-optimal**

- **Monte Carlo success (SR):**
  • State noise: SR 91.5%
  • Observation noise: 89.1%
  • Control errors: 84.4%
  • Single-step MTE: 81.9%
  • Multi-step MTE: 62.8% (hardest case… some runs miss Mars when failures are late in the trajectory)

- **General, minimal reward shaping:**

→ uses an **ε-constraint** schedule to enforce terminal constraints without reward hacking

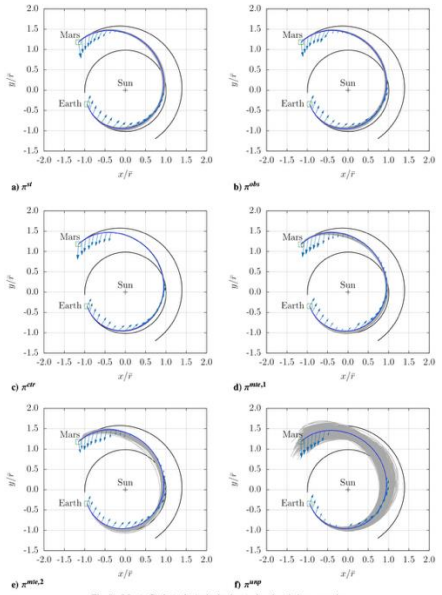


Fig. 5 Monte Carlo trajectories in the stochastic mission scenarios.

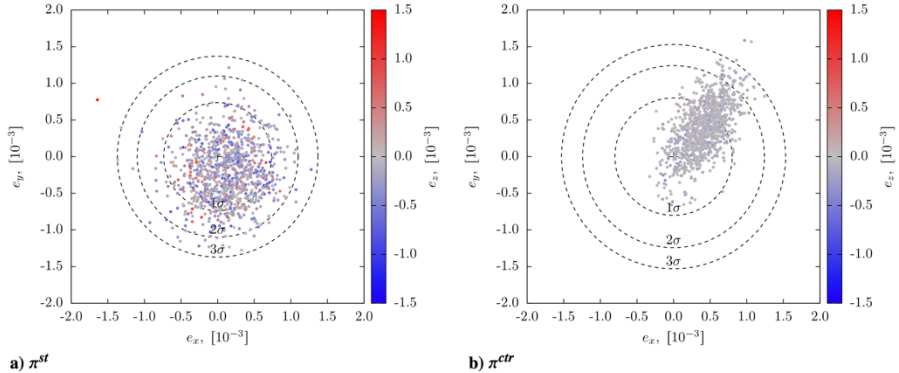


Fig. 6 Distribution of the position errors at Mars encounter for policies π^{st} and π^{ctr}.

Critiques & limitations

- **Impulsive, coarse grid (N=40)**:
  - ~9-day spacing
  - simplifies low-thrust into impulses → not flight-grade
- **Single-environment training:**
  - policies over-fit to the assumed failure model degrading in clean runs (ex: always expecting one MTE)
  - authors suggest meta-learning over environment families (exposing it to various uncertainties at once)
- **No formal stability guarantee for DNN guidance**

  → this is a common shortcoming in papers using RL in Aerospace
  - performance assessed via Monte Carlo
- **What happens when the various uncertainties and errors are combined?**

  → How would it perform given the same training?

  → What would be needed to improve it?

Computational resources

Can a student reproduce this?

→ **Yes, on a decent desktop!**

Authors trained 200M steps in ~12 h on an 8-core i7-9700K with 8 parallel envs @3.6GHz

Impact, legacy, and “next chapters”

- **Total citations: 51 citations** (as of September 2nd 2025)
- **Key contributions:**
  - showed **RL** can hit **near-optimal** deterministic performance
  - **meaningful robustness** under several uncertainties
- **Recommended Reading:**
  - Meta-reinforcement learning for adaptive spacecraft guidance during finite-thrust rendezvous missions

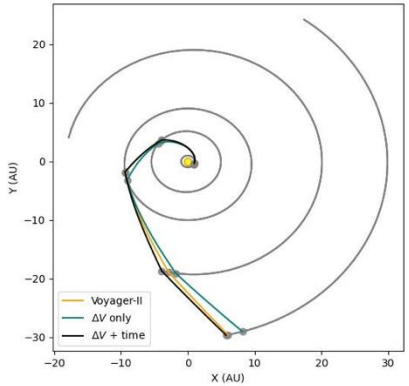
  → “future work” implemented by same authors

- **Legacy: led to other papers that built on top of this work to:**
  - guarantee optimality using RL: Optimal Multi-impulse Linear Rendezvous via Reinforcement Learning
  - Using image-based navigation: Image-Based Meta-Reinforcement Learning for Autonomous Guidance of an Asteroid Impactor

How does this work connect with interplanetary trajectory design overall?

- Much research in this field is not just the study of one leg but leveraging multiple legs to minimize delta-v for deep space missions
- My thesis solved multi-gravity-assist (MGA) routing with a genetic algorithm (GA)

→ typically Lambert-leg based with deterministic, Sun–only gravity between flybys.



Trajectory Result			
	Real	minimum ΔV ($\alpha = 1$)	ΔV and Time Tradeoff ($\alpha = 0.7$)
Earth Departure			
	$20-08-1977$	$04-09-1977$	$01-09-1977$
Date T_0	2443375.5	2443391	2443388
Departure Δv[m/s]	10230.7	9413.34	9855.66
Neptune Arrival			
	$25-08-1989$	$15-09-1991$	$25-04-1989$
Date T_4	2447763.5	2448484.125	2447277
Arrival V_{in}[m/s]	21551.0	15349.0	21862.2
Result			
Total cost [Δv]	**14830.12**	**9414.075**	**9856.34**

- **Where they meet:**
  - Treat the RL-leg as a drop-in robust leg solver inside MGA
  - GA chooses sequence and times (including terminal conditions to patch each leg)

  → use as inputs for RL-policy to get a full trajectory that is also robust to uncertainties and errors

Summary

- **Traditional methods:** Challenge to find robust solutions when stochastic uncertainties and errors are involved

  → RL can take non-convex constraints, uncertainties etc. into account in a highly flexible reward function

- **Big idea:** Train **once** on the ground to get a **onboard feedback policy** that is **robust to uncertainties or errors**
- **Result:**
  - Near-optimal in clean conditions
  - High Success rate: under several disturbances; struggles mainly with multi-step MTEs.

→ RL-trained policy present promising ‘robust solutions’ for in-space applications but more work needs to be done…

References

- **Zavoli, A., & Federici, L. (2021**). Reinforcement Learning for Robust Trajectory Design of Interplanetary Missions. Journal of Guidance, Control, and Dynamics, 44(8), 1440–1455. https://doi.org/10.2514/1.G005794
- **Sánchez-Sánchez, C., & Izzo, D.** Real-Time Optimal Control via Deep Neural Networks: Study on Landing Problems
- **Gaudet, B., Linares, R., & Furfaro, R. (2020).** Six Degree-of-Freedom Body-Fixed Hovering over Unmapped Asteroids via LIDAR Altimetry and Reinforcement Meta-Learning. Acta Astronautica, 172, 90–… https://doi.org/10.1016/j.actaastro.2020.03.026
- **Federici, L., et al. (2022).** Meta-Reinforcement Learning for Adaptive Spacecraft Guidance during Finite-Thrust Rendezvous Missions. Acta Astronautica, 201, 129–141. https://doi.org/10.1016/j.actaastro.2022.08.047
- **Federici, L., et al. (2022).** Image-Based Meta-Reinforcement Learning for Autonomous Guidance of an Asteroid Impactor. Journal of Guidance, Control, and Dynamics, 45, 2013. https://doi.org/10.2514/1.G006832
- **Xu, L., Zhang, G., Qiu, S., & Cao, X. (2023).** Optimal Multi-impulse Linear Rendezvous via Reinforcement Learning. Space: Science & Technology, 3, Article 0047. https://doi.org/10.34133/space.0047
- **Huterer Prats, D. (2023).** Using genetic algorithm optimization as a multi-gravity assist trajectory design tool (Master's thesis, Universitat Politècnica de Catalunya). UPC Open Access. Retrieved from https://hdl.handle.net/2117/392039