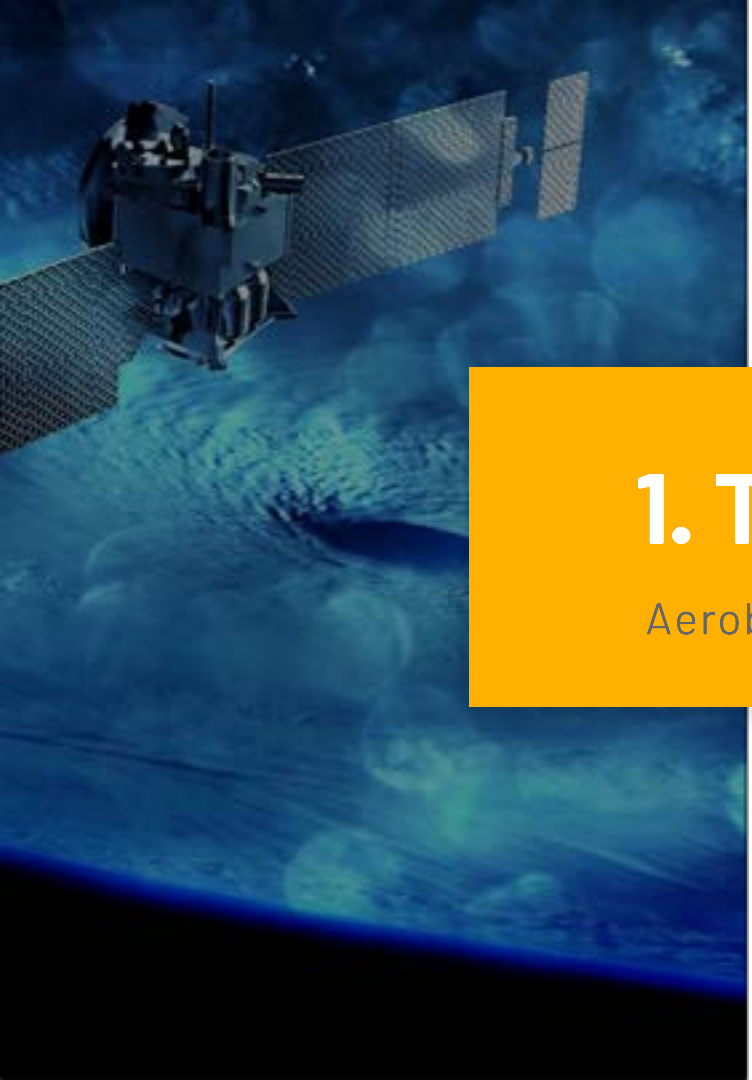
AEROBRAKING AT MARS: A Machine Learning Implementation

Giusy Falcone
Zachary R. Putnam

Department of Aerospace Engineering
University of Illinois at Urbana-Champaign



1. THE CONCEPT

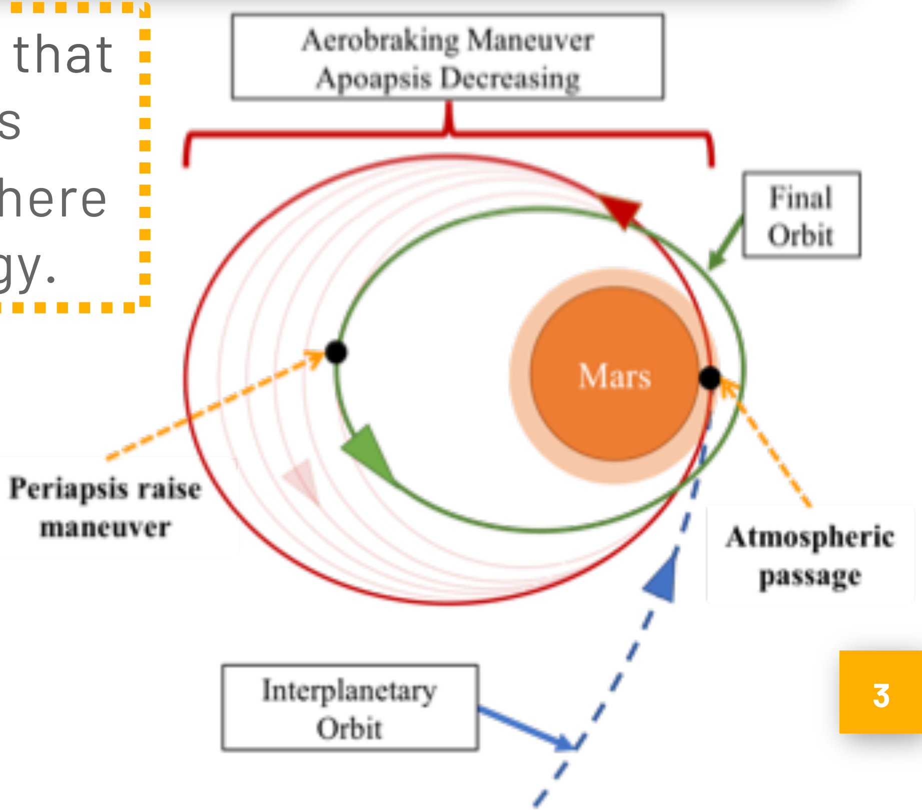
Aerobraking, Autonomy, Goal & Challenges

AEROBRAKING

Aerobraking is a maneuver that uses successive passes through the upper atmosphere to dissipate orbital energy.

Creation of drag is a function of velocity and flight-path angle

Propulsive maneuvers at apoapsis control velocity and flight-path angle



IMPORTANCE OF AEROBRAKING & AUTONOMY

Benefits and Costs

Massive propellant saving WRT single propulsive maneuver at the expense of risk caused by variability in atmospheric density (**heat rate & dynamic pressure**)

Successfully performed for three Mars missions:

- Mars Global Surveyor (1996)
- Mars Odyssey (2001)
- Mars Reconnaissance Orbiter (2005)

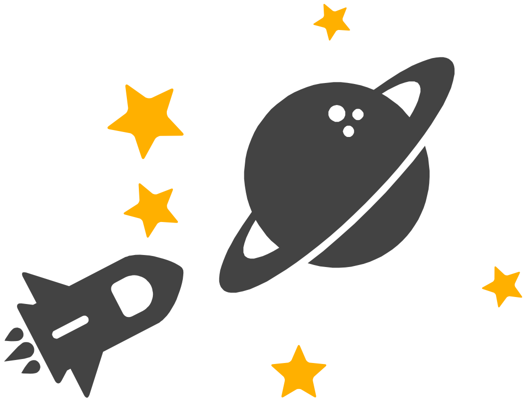
Importance of Autonomy

Ground cost, a team of engineers always-online, less aggressive conditions (more when orbital period decreases)

Previous efforts to address these issues with autonomy:

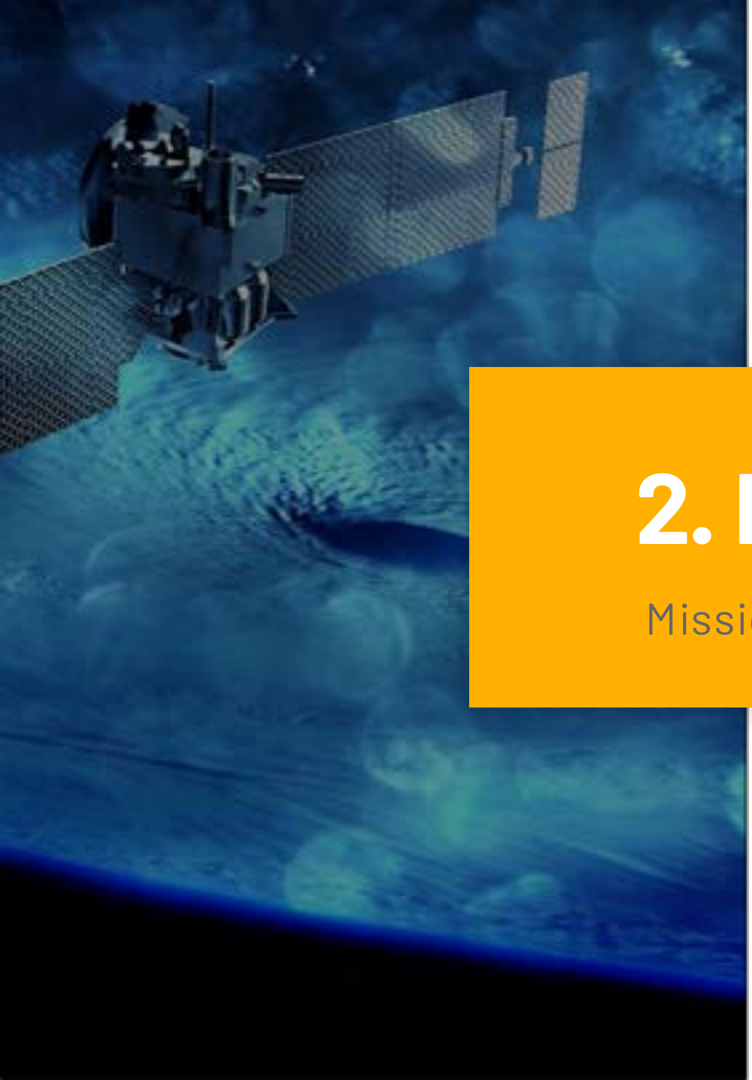
Aerobraking Autonomous Control (1999-2012)

GOAL AND CHALLENGES



Perform a complete and successful autonomous aerobraking campaign at Mars with a learning and adaptive behavior approach while:

1. Satisfying constraints on dynamic pressure and heat rate
2. Managing mission risk
3. Minimizing control effort and time of flight (**cost**)



2. PROBLEM FORMULATION

Mission Modeling, Reinforcement Learning and Interface

AEROBRAKING MISSION MODELING

Aerobraking mission: vary periapsis altitude to lower apoapsis altitude while satisfying constraints

r_{ao}

r_{pi}

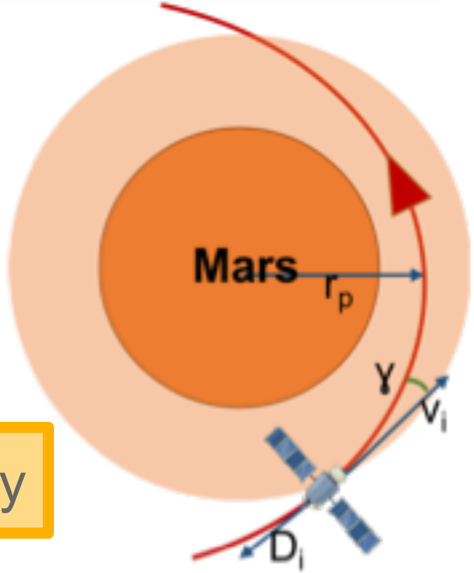
Keplerian Trajectory Simulator

v_i, fpa_i

Aeroassist Simulation

r_{ai}, dp_i, hr_i

Autonomy



REINFORCEMENT LEARNING

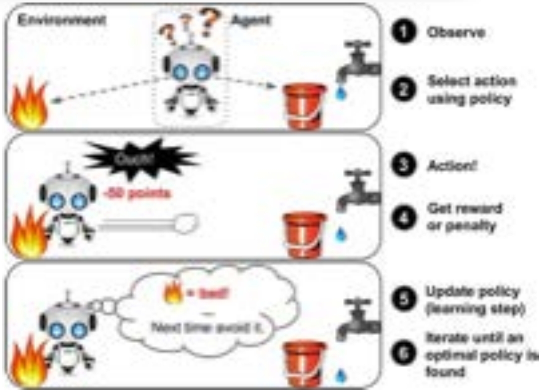
Tabular Q-Learning Algorithm with ε-greedy policy search

$$<S, A, P^{a}_{ss'}, R^{a}_{ss'}, \gamma>$$

S = apoapsis radius

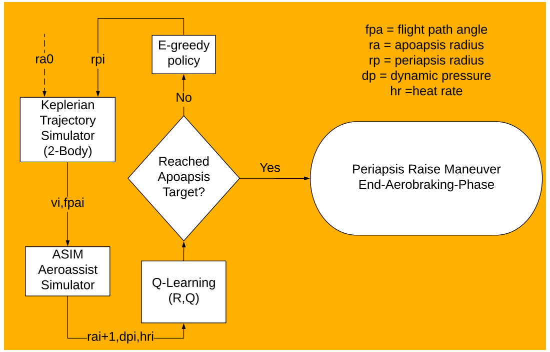
A = periapsis altitude

$R^{a}_{ss'}$ =reward built to minimize the aerobraking whole time.

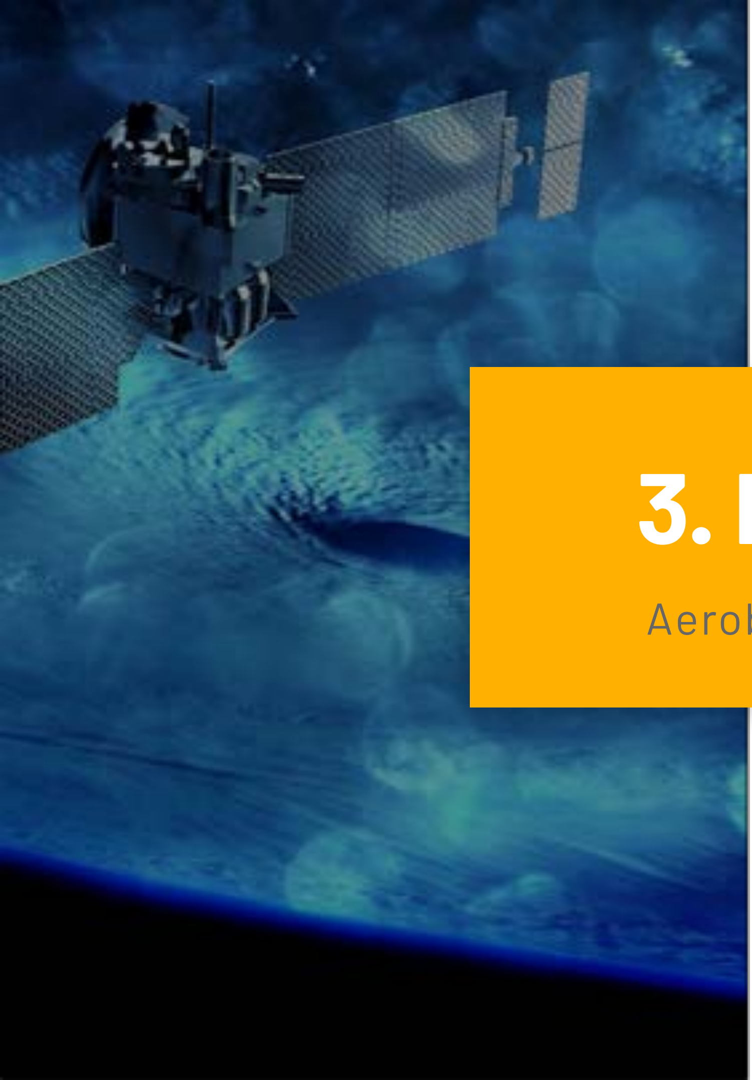


“ **Reinforcement learning is learning what to do...so as to maximize a numerical reward signal. (Sutton)**

INTERFACE BETWEEN MISSION AND RL



A trained policy chooses when perform a trim maneuver to minimize the aerobraking time and to avoid the violating constraints.



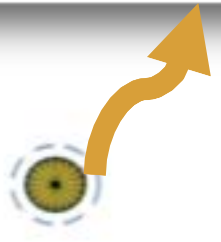
3. RESULTS

Aerobraking Simulation, Constraints & Corridor, Learning

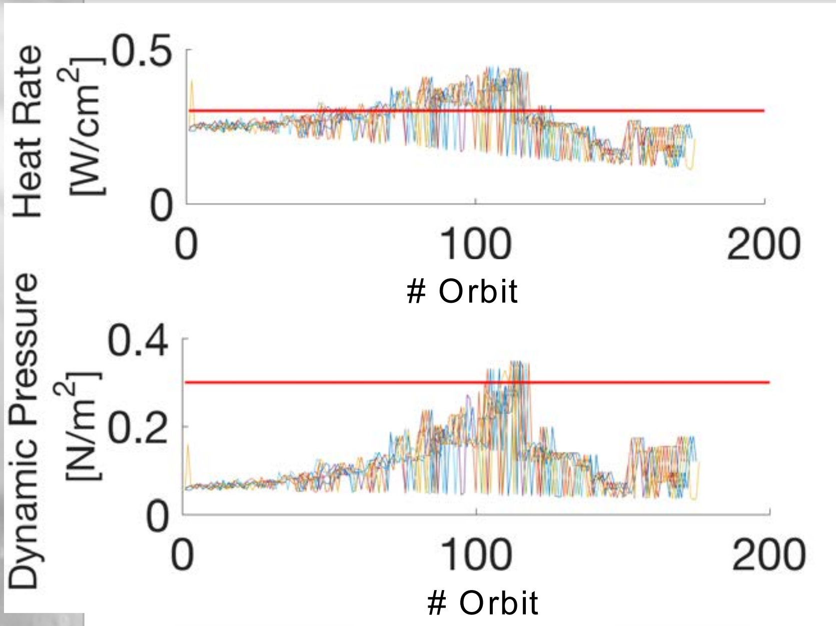
AEROBRAKING SIMULATION

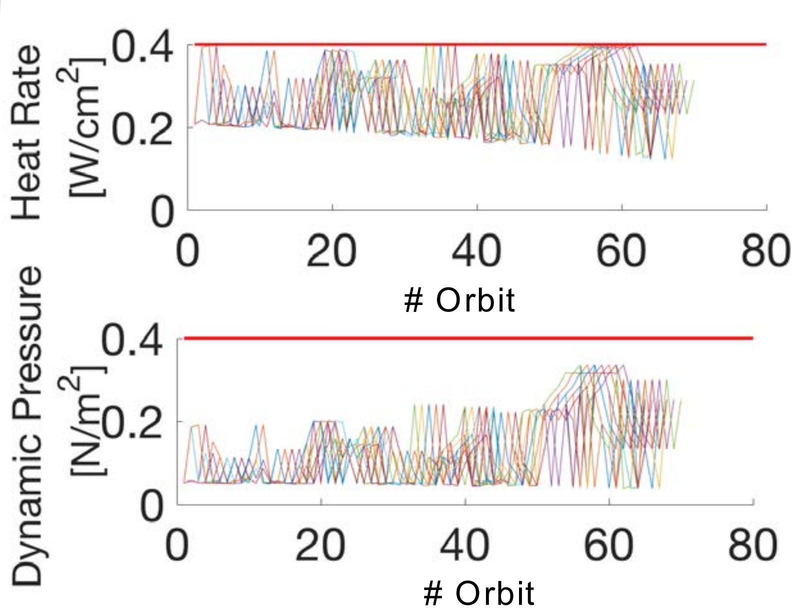
Apoapsis radius from 200,000 km to 5,000 km

PERIAPSIS



CONSTRAINTS AND CORRIDOR

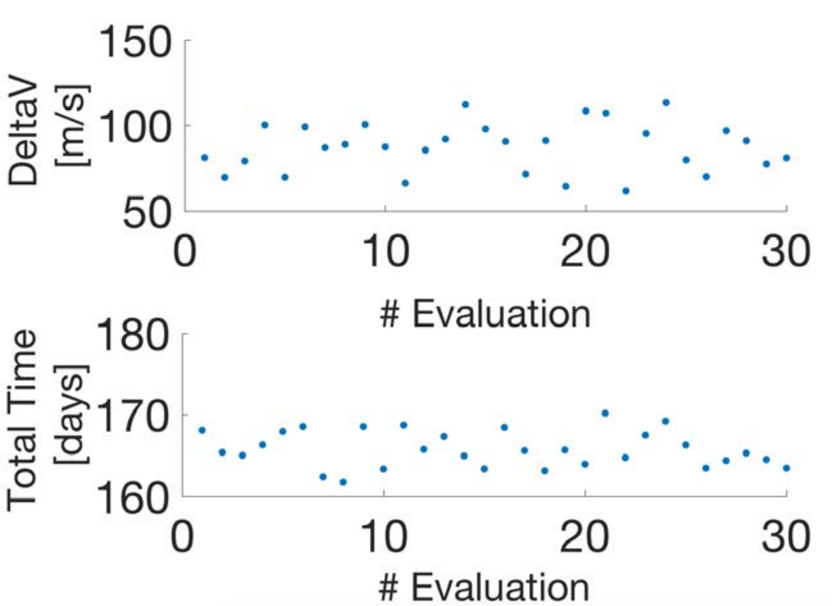


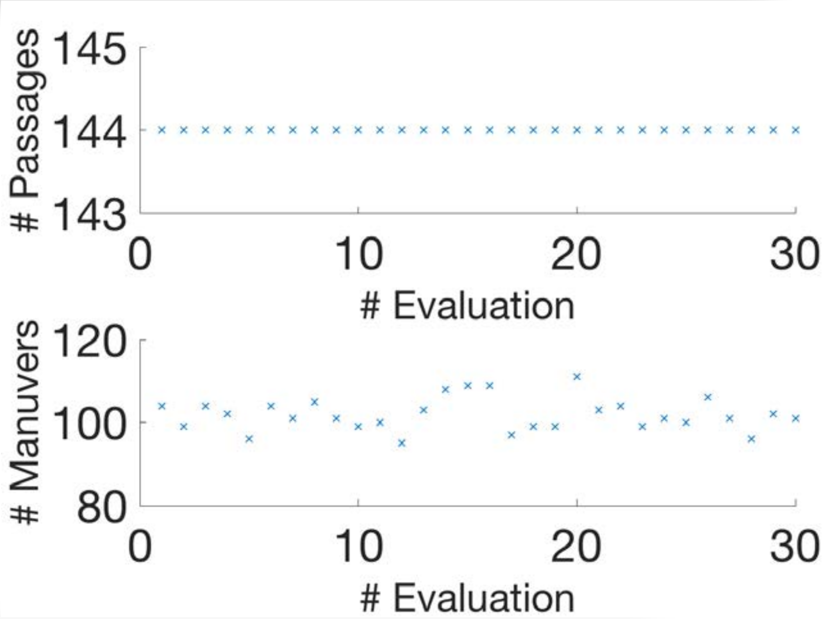


NOT TRAINED MACHINE

TRAINED MACHINE

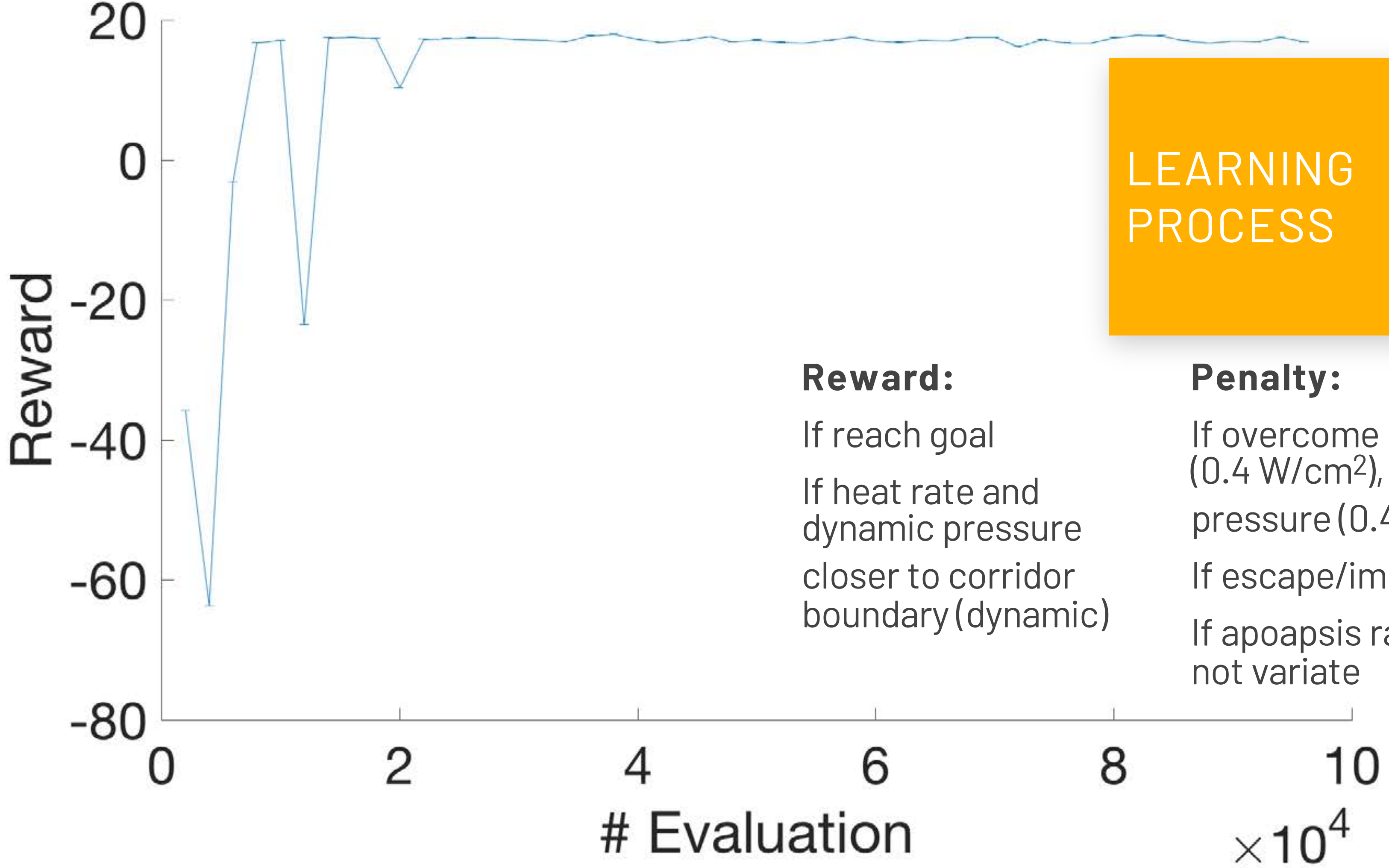
RESULTS DELTA-V AND TOTAL TIME





REWARD FUNCTION DECREASES ONLY TOTAL TIME

LEARNING PROCESS



Reward:

If reach goal

If heat rate and dynamic pressure closer to corridor boundary (dynamic)

Penalty:

If overcome heat rate (0.4 W/cm^2), dynamic pressure (0.4 N/m^2)

If escape/impact

If apoapsis radius does not variate

THANKS!

Any questions?

You can find me at:

gfalcon2@illinois.edu

REFERENCES

1. J. L. Hanna, R. Tolson, A. D. Cianciolo, and J. Dec, "Autonomous aerobraking at mars," 2002.
2. J. L. Prince, R. W. Powell, and D. Murri, "Autonomous aerobraking: A design, development, and feasibility study," 2011.
3. J. A. Dec and M. N. Thornblom, "Autonomous aerobraking: Thermal analysis and response surface development," 2011.
4. R. Maddock, A. Bowes, R. Powell, J. Prince, and A. Dwyer Cianciolo, "Autonomous aerobraking development software: Phase one performance analysis at mars, venus, and titan," in AIAA/AAS Astrodynamics Specialist Conference , 2012, p. 5074.
5. D. G. Murri, R. W. Powell, and J. L. Prince, "Development of autonomous aerobraking (phase 1),"2012.
6. D. G. Murri, "Development of autonomous aerobraking-phase 2," 2013.
7. Z. R. Putnam, "Improved analytical methods for assessment of hypersonic drag-modulation trajectory control," Ph.D. dissertation, Georgia Institute of Technology, 2015.
8. A. Geramifard, T. J. Walsh, S. Tellex, G. Chowdhary, N. Roy, J. P. How et al. , "A tutorial on linear function approximators for dynamic programming and reinforcement learning," Foundations and Trends® in Machine Learning , vol. 6, no. 4, pp. 375–451, 2013.
9. J. L. H. Prince and S. A. Striepe, "Nasa langley trajectory simulation and analysis capabilities for mars reconnaissance orbiter."
10. V. Mnih, "Playing atari with deep reinforcement learning." *arXiv preprint arXiv:1312.5602* (2013).
11. R.S . Sutton, and A. G. Barto. " Reinforcement learning: An introduction. ", Vol. 1. No. 1. Cambridge: MIT press, 1998.
12. M. Techlabs, " Picture Reinforcement Learning Explanation. ", Retrieved from https://goo.gl/images/JTxJpf

MISSION DESIGN CHARACTERISTICS

Small Satellite

Mass: 110 kg

Drag Area: 10 m^2

C_D: 1.2

Keplerian Trajectory Simulator (2-bodies)

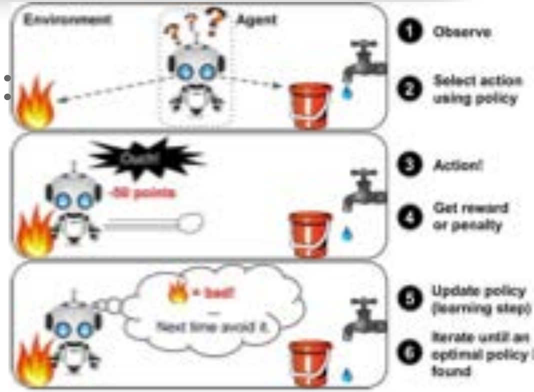
$$\ddot{r} - r\dot{\vartheta}^2 = -\frac{\mu}{r}$$

Aeroassist Simulator

3 DOF equations of motion. Numeric simulation integrated using a forth order Runge-Kutta scheme.

Density model: Mars-Gram 2010

REINFORCEMENT LEARNING



Markov Decision Process (stochastic):

<S, A,P $^{a}_{ss'}$, R $^{a}_{ss'}$, γ >

S = state space (apoapsis radius)

A = action space (periapsis altitude)

P $^{a}_{ss}$ ==probability of getting into s′ after a from s

R $^{a}_{ss'}$ =expected reward from s to s′ after a

Find policy π^* through iteration Bellman eq.

$$Q^*(s,a) = \sum_{s' \in S} P_{ss'}^{\pi(s)}\left[R_{ss'}^{a} + \gamma max_{a'} Q^*(s',a')\right]$$

MDP

STATE SPACE

Apoapsis radius

From 4000 to 400000 km.

981 states.

ACTION SPACE

Periapsis altitude

From 105 to 127 km.

147 states.

REWARD FUNCTION

Reward:

Reach goal.

If closer to corridor boundary (dynamic).

Penalty:

Overcome heat rate (0.4 W/cm^2), dynamic pressure (0.4 N/m^2).

Escape/Impact.

If apoapsis radius does not variate.

Q-LEARNING ALGORITHM

Algorithm 5: Q-Learning	Complexity
Input: MDP $\setminus \{\mathcal{P}, \mathcal{R}\}, \alpha, \epsilon$	
Output: π	
$\boldsymbol{\theta} \leftarrow$ Initialize arbitrarily	
$\langle s, a \rangle \leftarrow \langle s_0, \pi^{\epsilon}(s_0) \rangle$	
while *time left* **do**	
Take action a and receive reward r and next state s'	
$Q^{+}(s, a) \leftarrow r + \gamma \max_{a'} Q(s', a')$	$\mathcal{O}(n\|\mathcal{A}\|)$
$\delta \leftarrow Q^{+}(s, a) - Q(s, a)$	
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \delta \phi(s, a)$	$\mathcal{O}(n)$
$\langle s, a \rangle \leftarrow \langle s', \pi^{\epsilon}(s') \rangle$	$\mathcal{O}(n\|\mathcal{A}\|)$
return π greedy w.r.t. Q	

Off-policy and model free algorithm

ε-GREEDY POLICY SEARCH

ε-greedy policy spans between exploration and exploitation:

- With probability 1- ε, uses the greedy action:

$$a_i = \arg\max_a \hat{Q}(s_i, a)$$

- With probability ε, play random action.