Trajectory Planning for Non-Differentially Flat Systems using Reinforcement Learning Framework

Oscar Jenot
Master Thesis in Mechanical Engineering
Swiss Federal Institute of Technology Lausanne (EPFL)
June 2021

Motivation and Objectives

Some classes of cranes have the physical or geometric properties of flatness. Trajectory planning and the construction of associated inputs are obtained by basic differentiation of a sufficiently smooth path of the flat outputs. The relationship between flatness and controllability is tight. [1]

The objective of this project is to study the resolution of non-differentially flat system's control problem using machine learning, and more specifically reinforcement learning (RL), in the case of non-flat systems obtained by a perturbation of flat systems.

The methods used will be to create a pure neural network-based controller, trained based on a model free reinforcement learning framework, in order to determine if such method can accomplish the task of controlling underactuated nonlinear systems.

Topic	Question	Why ?	Objectives
Trajectory planning and control laws for complex (non-flat) systems using reinforcement learning	Does the knowledge of the flat system helps in solving its geometrically modified non-flat version, using RL tools ?	Know if new tools such as Reinforcement Learning can be used for optimal control of complex systems	Present a procedure in an intent to solve the optimal control problem using a reinforcement learning framework. Three classes of cranes will be presented

Case Study

The examples that will be treated are three classes of cranes. The cart pendulum, an underactuated over-head crane (transporting a mass point), and a fixed length crane transporting a vertical load. This system (Crane V2) loses its flatness property when the crane displaces a very long object vertically.

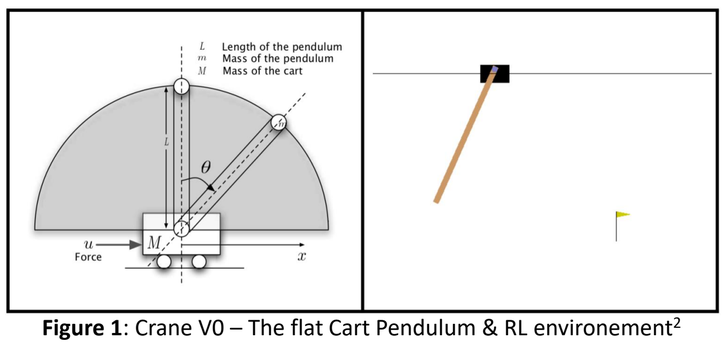

Figure 1: Crane V0 – The flat Cart Pendulum & RL environement[2]

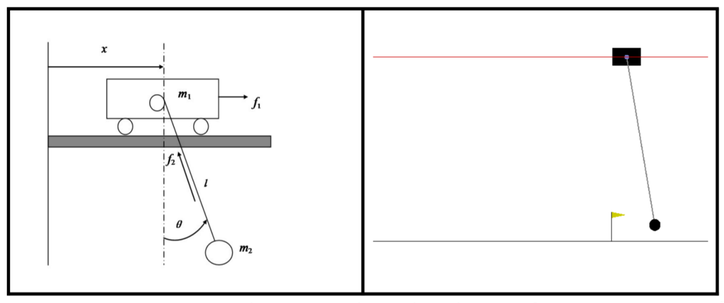

Figure 2: Crane V1 – Underactuated overhead crane & RL environement[3]

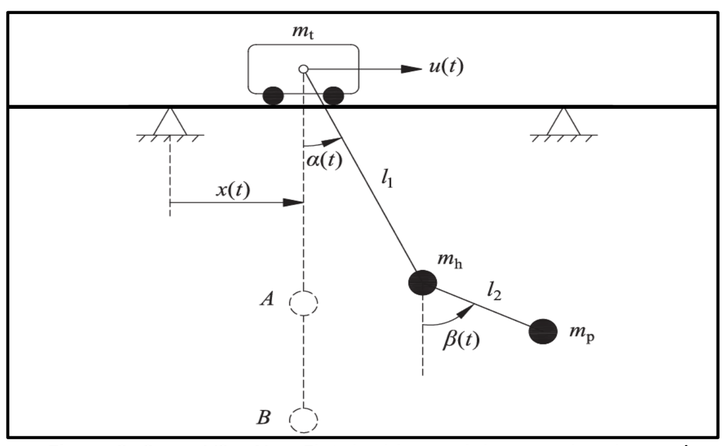

Figure 3: Crane V2 – Non-Flat double pendulum overhead crane[4]

Methodology

Reinforcement Learning is a framework based on an agent-environment interface. The agent (controller in engineering), interacts with the environment (system or plant). In each discrete time step, the agent takes an action (control signal). Depending on the action taken, the state of the environment will change. The new state, and a numerical reward is then given by the environment to the agent. Iteratively, the agent decides on a new action to take. **The agent's objective is to maximize cumulative rewards.** * [8]

$$q_\pi(s,a) = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a\right]$$

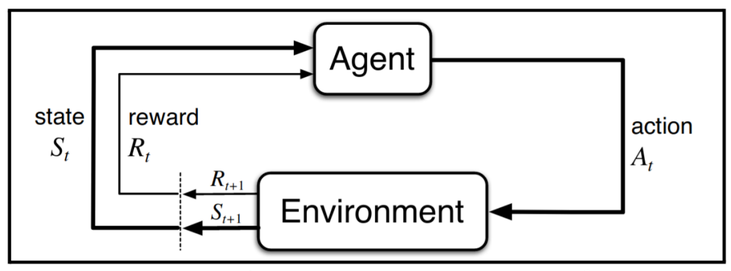

Figure 4: The agent-environment interaction[8]

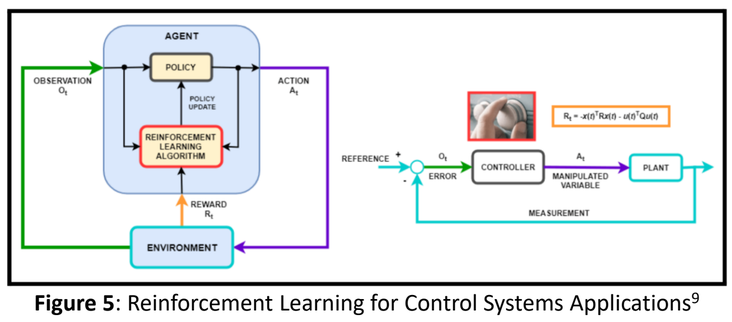

Figure 5: Reinforcement Learning for Control Systems Applications[9]

γ	τ	α	Update every	ϵ_{start}	ϵ_{end}	ϵ_{decay}	Max time step	Buffer Size	Batch Size
Discount factor	Update of target paramet-ers	Learning Rate	Network Update	Explorati on	Exploitati on	Epsilon greedy policy	Max time steps	Replay buffer size	Minibatch size
0.995	1e-3	5e-4	4	1	0.01	0.997	1500	1e5	64

Table 1: Agent and Model parameters for the environments' training

*The learning algorithm was implemented by adapting the Deep Q-Network model from Mnih et al., (2015)[5], implemented in python by Henri Chan[6], and adapted and tuned for our environment and rewards. We implemented it using Open AI's RL toolkit[7]

Results – Crane V0

- Episode solved around 1200 episodes
- Understanding the difficulty of rewards
- Stressing the model performs well

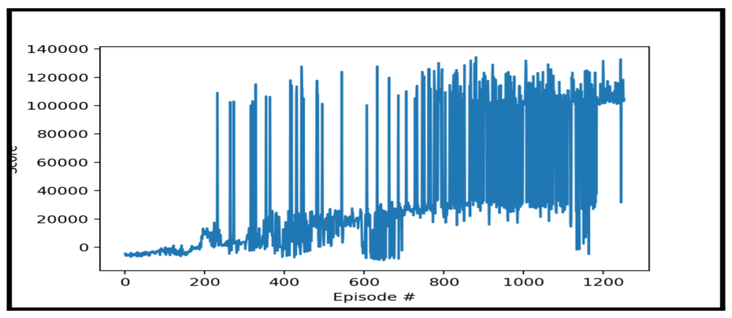

Figure 6: Score per episode during training for Crane V0

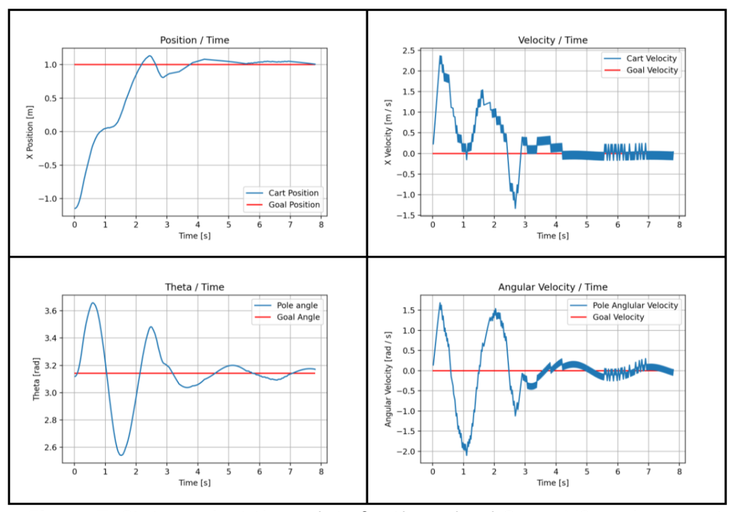

Figure 7: State against time plots for the solved Crane V0 environment

Results – Crane V1

- Episode not solved but near optimal solution after 800 episodes
- Reward difficulties:
  - No swaying but wrong length
  - Good length but swaying
- Difficulties stabilizing at the end of the trajectory
  - Controls two inputs independently from any control laws
  - We don't want find solutions that interfere with the agent's optimal trajectory

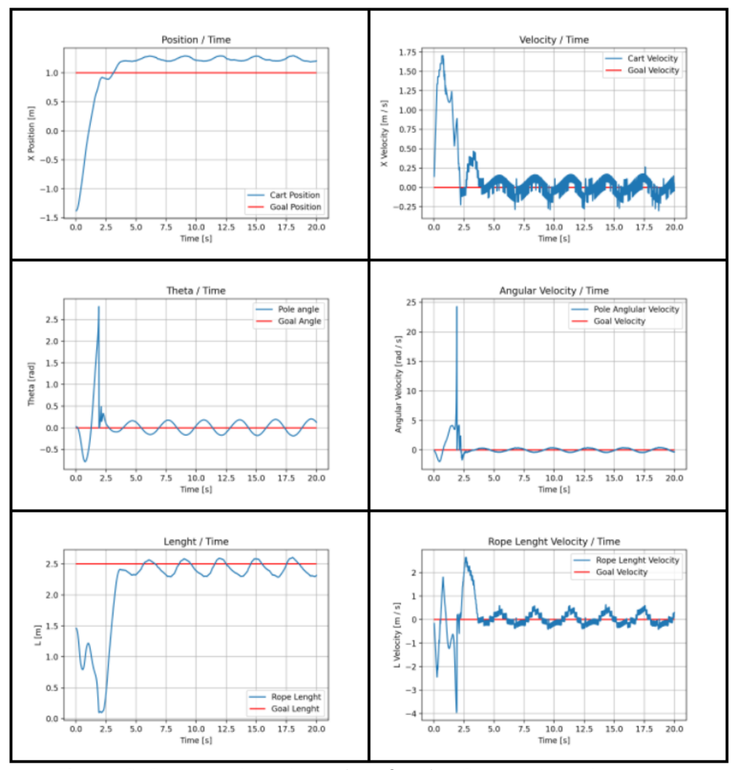

Figure 8: State against time plots for the Crane V1 environment

Results – Crane V2

- No control law was found
- Environment is too complex for the simple model
- Algorithm needs to train way longer for agent to find sparse reward
- The environment is ready to be trained by an ML expert ! (See bottom left GitHub link)

Discussion & Conclusion

The case study shows that a simple Deep Q-Network reinforcement learning algorithm can perfectly control a flat crane. When adding complexity to the system, that same learning algorithm is struggling to stabilize the system and oscillates around the equilibrium point.

Conclusion

- Classes of pure Reinforcement Learning controllers can perform well on complex systems
- Our simple Deep Q-Network model performs okey but there is place for improvement
- We can not conclude that RL controller can solve non-flat systems

Possible improvements :

- Imitation learning
- More computation power
- Smaller state space
- More complex training algorithms

Difficulties encountered

- Understanding the RL agent and how it performs
- Finding the right agent and Neural Network
- Finding the right reward signals

Supervisors & Links

Main supervisor: Dr. Philippe Muellhaupt
EPFL – Automatic Control Laboratory

External expert: Dr. Willson Sudarsandhari Shibani
ETEL S.A.

Project's website:
https://github.com/oscarjenot/pdm_oscar_jenot
Contact: oscar.jenot@alumni.epfl.ch

References

1. Fliess, M., Lévine, J., Martin, P., & Rouchon, P. (1995). Flatness and defect of non-linear systems: Introductory theory and examples. International Journal of Control, 61(6), 1327-1361. doi:10.1080/00207179508921959
2. Willson, S. S., Mullhaupt, P., & Bonvin, D. (2011). A quotient method for designing nonlinear controllers. IEEE Conference on Decision and Control and European Control Conference. doi:10.1109/cdc.2011.6160803
3. Liu, D., & Guo, W. (2012). Tracking Control for an Underactuated Two-Dimensional Overhead Crane. Journal of Applied Research and Technology, 10(4). doi:10.22201/icat.16656423.2012.10.4.383
4. Chen, Q., Cheng, W., Gao, L., & Fottner, J. (2019). A pure neural network controller for double-pendulum crane anti-sway control: Based on Lyapunov stability theory. Asian Journal of Control, 23(1), 387-398. doi:10.1002/asjc.2226
5. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., . . . Hassabis, D. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529-533. doi:10.1038/nature14236
6. "kinwo/deeprl-navigation: Deep Reinforcement Learning ... - GitHub." 1 Sept. 2018, https://github.com/kinwo/deeprl-navigation. Accessed 8 Jun. 2021.
7. (n.d.). openai's gym's envs - OpenAI Gym. Retrieved April 30, 2021, from https://gym.openai.com/envs/
8. Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. The MIT Press.
9. Reinforcement Learning for Control Systems Applications - MATLAB. Retrieved April 30, 2021, from https://www.mathworks.com/help/reinforcement-learning/ug/reinforcement-learning-for-control-systems-applications.html