



# **Reinforcement Learning for Tracking Control in Robotics**

## **Delft University of Technology**

Yudha Prawira Pane

February 20, 2015

Supervisors: Prof. Dr. Robert Babuška

Subramanya P. Nageshrao



# Outline

## 1 Preliminaries

Basics of Reinforcement Learning (RL) & Tracking Control

Experimental Setup: 3D Printing Robot System

## 2 Literature Survey Result

RL for Tracking: A Survey

Research Plan & Conclusion

# Next Subsection

## 1 Preliminaries

Basics of Reinforcement Learning (RL) & Tracking Control

Experimental Setup: 3D Printing Robot System

## 2 Literature Survey Result

RL for Tracking: A Survey

Research Plan & Conclusion

# Motivation of RL

- Inspired by how animals learn

# Motivation of RL

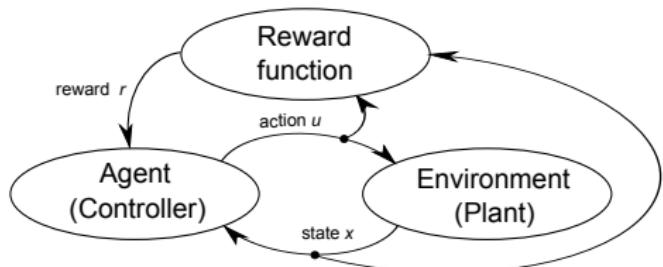
- Inspired by how animals learn
- Learning through interaction with environment

# Motivation of RL

- Inspired by how animals learn
- Learning through interaction with environment



source: google



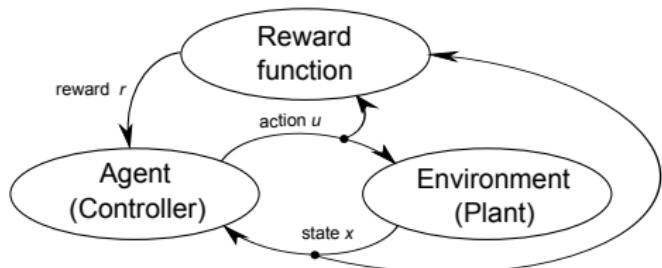
source: SC4081 slide

# Motivation of RL

- Inspired by how animals learn
- Learning through interaction with environment



source: google



source: SC4081 slide

- Introduce the reward / reinforcement signal

# Mathematical Formalization of RL

## Definition

A Markov Decision Process (MDP) is defined as a tuple  $\langle X, U, \bar{f}, \rho \rangle$  where

- $X$  = state space
- $U$  = action space
- $\bar{f} : X \times U \rightarrow X$  = system dynamics
- $\rho : X \times U \rightarrow \mathbb{R}$  = reward (cost) function

# Mathematical Formalization of RL

## Definition

A Markov Decision Process (MDP) is defined as a tuple  $\langle X, U, \bar{f}, \rho \rangle$  where

- $X$  = state space
- $U$  = action space
- $\bar{f} : X \times U \rightarrow X$  = system dynamics
- $\rho : X \times U \rightarrow \mathbb{R}$  = reward (cost) function

## Definition

System and policy dynamics are defined as:

$$x_{k+1} = \bar{f}(x_k, u_k) = f(x_k) + g(x_k)u_k \quad (1)$$

$$u_{k+1} = \pi(x_k, u_k) \quad (2)$$

$$r_{k+1} = \rho(x_k, u_k) \quad (3)$$

# Mathematical Formalization of RL (2)

## Definition

Formulate goal as return  $R$

$$R_k = r_k + r_{k+1} + r_{k+2} + \dots + r_T \quad (4)$$

Discounted return:

$$R_k = r_{k+1} + \gamma r_{k+2} + \gamma^2 r_{k+3} + \dots = \sum_{j=0}^{\infty} \gamma^j r_{k+j+1} \quad (5)$$

# Mathematical Formalization of RL (2)

## Definition

Formulate goal as return  $R$

$$R_k = r_k + r_{k+1} + r_{k+2} + \dots + r_T \quad (4)$$

Discounted return:

$$R_k = r_{k+1} + \gamma r_{k+2} + \gamma^2 r_{k+3} + \dots = \sum_{j=0}^{\infty} \gamma^j r_{k+j+1} \quad (5)$$

## Definition

Value function  $V$  measures how good it is to be at a certain state  $x$

$$V(x_k) = \rho(x_k, u_k) + \gamma \rho(x_{k+1}, u_{k+1}) + \gamma^2 \rho(x_{k+2}, u_{k+2}) + \dots \quad (6)$$

$$V(x_k) = \rho(x_k, u_k) + \gamma V(x_{k+1}) \quad (7)$$

# Mathematical Formalization of RL (3)

How to obtain an optimal policy  $\pi^*$ ?

# Mathematical Formalization of RL (3)

How to obtain an optimal policy  $\pi^*$ ?

First, an exact value function  $V$   $\forall x \in \mathcal{X}$  must be found

# Mathematical Formalization of RL (3)

How to obtain an optimal policy  $\pi^*$ ?

First, an exact value function  $V$   $\forall x \in \mathcal{X}$  must be found

## Definition

Bellman optimality principle:

$$V^*(x_k) = \min_{u_k} \left[ \rho(x_k, u_k) + \gamma V^*(x_{k+1}) \right] \quad (8)$$

Optimal policy:

$$\pi^*(x_{k-1}) = u_k^* = \operatorname{argmin}_{u_k} \left[ \rho(x_k, u_k) + \gamma V^*(x_{k+1}) \right] \quad (9)$$

# Solutions to RL problem

Dynamic Programming (DP):

- needs a system model

# Solutions to RL problem

Dynamic Programming (DP):

- needs a system model
- example: policy iteration (PI)

# Solutions to RL problem

Dynamic Programming (DP):

- needs a system model
- example: policy iteration (PI)

---

## Algorithm 3 Policy Iteration

---

```
1: Initialization:
2:   Start from an admissible policy  $\pi$ , assign  $V^\pi(x) \leftarrow 0$ 
3: repeat
4:   Policy Evaluation:
5:   repeat
6:      $\Delta \leftarrow 0$ 
7:     For each  $x \in \mathcal{X}$  :
8:        $v \leftarrow V^\pi(x)$ 
9:        $V^\pi(x) \leftarrow \rho(x, \pi(x)) + \gamma V^\pi(x')$ 
10:       $\Delta = \max(\Delta, |v - V^\pi(x)|)$ 
11:    until  $\Delta < \epsilon$  (a small positive number)
12:   Policy Improvement:
13:   For each  $x \in \mathcal{X}$  :
14:      $\pi(x) = \arg \min_u \rho(x, u) + \gamma V^\pi(x')$ 
15: until  $\pi$  converges
```

---

# Solutions to RL problem (2)

Temporal Difference (TD)

- does not need a system model

# Solutions to RL problem (2)

Temporal Difference (TD)

- does not need a system model
- examples: Q-learning, actor-critic

# Solutions to RL problem (2)

## Temporal Difference (TD)

- does not need a system model
- examples: Q-learning, actor-critic

# Solutions to RL problem (2)

## Temporal Difference (TD)

- does not need a system model
- examples: Q-learning, actor-critic

---

## Algorithm 7 Actor-critic algorithm

---

- 1: **For every trial:**
  - 2: Initialize  $x_0$  and  $u_0 = \tilde{u}_0$
  - 3: **repeat**
  - 4:   apply  $u_k$ , measure  $x_{k+1}$ , receive  $r_{k+1}$
  - 5:   choose next action  $u_{k+1} = \hat{\pi}(x_{k+1}, \psi_k) + \tilde{u}_{k+1}$
  - 6:    $\delta_k = r_{k+1} + \hat{V}(x_{k+1}, \theta_k) - \hat{V}(x_k, \theta_k)$
  - 7:    $\theta_{k+1} = \theta_k + \alpha_c \delta_k \frac{\partial \hat{V}(x, \theta)}{\partial \theta} \Big|_{\substack{x=x_k, \theta=\theta_k}}$
  - 8:    $\psi_{k+1} = \psi_k + \alpha_a \delta_k \frac{\partial \hat{\pi}(x, \psi)}{\partial \psi} \Big|_{\substack{x=x_k, \psi=\psi_k}}$
  - 9: **until** terminal state
-

# Tracking Control

Typical tracking controllers:

- Open-loop control

# Tracking Control

Typical tracking controllers:

- Open-loop control
- State/Output Feedback control (e.g. PID controller)

# Tracking Control

Typical tracking controllers:

- Open-loop control
- State/Output Feedback control (e.g. PID controller)
- Feedback + feedforward control (e.g. LQT optimal controller)

# Tracking Control

Typical tracking controllers:

- Open-loop control
- State/Output Feedback control (e.g. PID controller)
- Feedback + feedforward control (e.g. LQT optimal controller)

# Tracking Control

Typical tracking controllers:

- Open-loop control
- State/Output Feedback control (e.g. PID controller)
- Feedback + feedforward control (e.g. LQT optimal controller)

**Drawback:** even the best of the 3 only performs as good as the model!

# Research Question

The following research question is raised:

*"Is it possible to improve the tracking performance of a nominal controller using Reinforcement Learning?"*

# Next Subsection

## 1 Preliminaries

Basics of Reinforcement Learning (RL) & Tracking Control

Experimental Setup: 3D Printing Robot System

## 2 Literature Survey Result

RL for Tracking: A Survey

Research Plan & Conclusion

# 3D Printing Robot

- The UR5 robot with unknown internal controller

# 3D Printing Robot

- The UR5 robot with unknown internal controller
- 4 types of command (default controllers):

# 3D Printing Robot

- The UR5 robot with unknown internal controller
- 4 types of command (default controllers):
  - Tool Position ( $x, y, z$ ) in meter
  - Tool Velocity ( $\dot{x}, \dot{y}, \dot{z}$ ) in meter/s
  - Joint Position ( $q_1, q_2, q_3, q_4, q_5, q_6$ ) in rad
  - Joint velocity ( $\dot{q}_1, \dot{q}_2, \dot{q}_3, \dot{q}_4, \dot{q}_5, \dot{q}_6$ ) in rad/s

# 3D Printing Robot

- The UR5 robot with unknown internal controller
- 4 types of command (default controllers):
  - Tool Position ( $x, y, z$ ) in meter
  - Tool Velocity ( $\dot{x}, \dot{y}, \dot{z}$ ) in meter/s
  - Joint Position ( $q_1, q_2, q_3, q_4, q_5, q_6$ ) in rad
  - Joint velocity ( $\dot{q}_1, \dot{q}_2, \dot{q}_3, \dot{q}_4, \dot{q}_5, \dot{q}_6$ ) in rad/s
- The laser scanner

# 3D Printing Robot

- The UR5 robot with unknown internal controller
- 4 types of command (default controllers):
  - Tool Position ( $x, y, z$ ) in meter
  - Tool Velocity ( $\dot{x}, \dot{y}, \dot{z}$ ) in meter/s
  - Joint Position ( $q_1, q_2, q_3, q_4, q_5, q_6$ ) in rad
  - Joint velocity ( $\dot{q}_1, \dot{q}_2, \dot{q}_3, \dot{q}_4, \dot{q}_5, \dot{q}_6$ ) in rad/s
- The laser scanner
- The 3D Print head

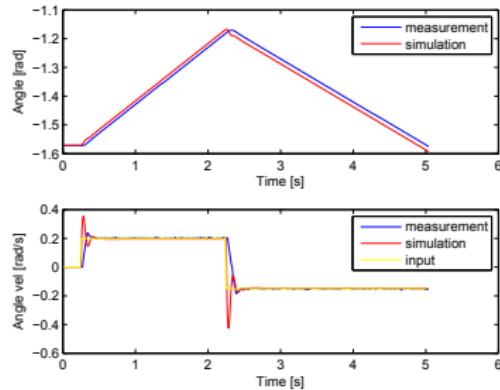
# 3D Printing Robot

- The UR5 robot with unknown internal controller
- 4 types of command (default controllers):
  - Tool Position ( $x, y, z$ ) in meter
  - Tool Velocity ( $\dot{x}, \dot{y}, \dot{z}$ ) in meter/s
  - Joint Position ( $q_1, q_2, q_3, q_4, q_5, q_6$ ) in rad
  - Joint velocity ( $\dot{q}_1, \dot{q}_2, \dot{q}_3, \dot{q}_4, \dot{q}_5, \dot{q}_6$ ) in rad/s
- The laser scanner
- The 3D Print head



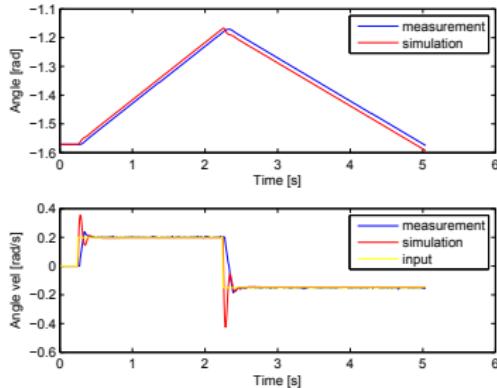
# Previous work: system identification

- Subspace identification



# Previous work: system identification

- Subspace identification



**Table 1.** VAF scores of the simulated outputs for all joints

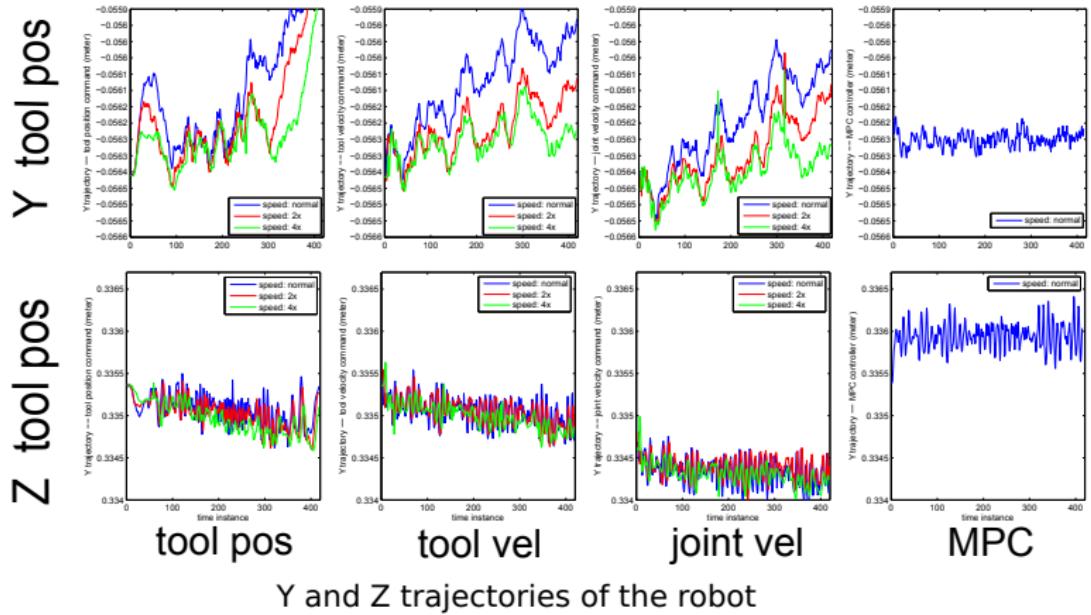
Joint	Position	Velocity
1	98.64	87.33
2	98.05	88.33
3	98.55	88.47
4	98.97	89.50
5	99.46	90.32
6	98.87	85.13

reference: de Gier (2014)

# The 3D Printing Robot in Action

# Previous work: MPC controller

- test a simple trajectory along X-axis and compare with default controllers



reference: de Gier (2014)

# Hypothesis

- "*Based on the system identification, there is a non-linear dynamics which can't be captured*"
- "*Current controller (MPC) relies heavily on the identified model which is not perfect. Therefore, the model-mismatch induced by the unknown dynamics is responsible for the non-optimal performance of the MPC controller*"

# Next Subsection

## 1 Preliminaries

Basics of Reinforcement Learning (RL) & Tracking Control

Experimental Setup: 3D Printing Robot System

## 2 Literature Survey Result

RL for Tracking: A Survey

Research Plan & Conclusion

# RL for Optimal Tracking Control

Assume a SISO LQT problem:

$$\begin{aligned}x_{k+1} &= Ax_k + Bu_k \\y_k &= Cx_k\end{aligned}\tag{10}$$

with reference signal  $r_k$ .

## Definition

The cost function:

$$J = V(x_k, r_k) := \frac{1}{2} \sum_{i=k}^{\infty} (Cx_i - r_i)^T Q (Cx_i - r_i) + u_i^T R u_i \tag{11}$$

references: Kiumarsi and Lewis (2014), Modares and Lewis (2014)

# RL for Optimal Tracking Control (2)

Important assumption:

$$r_{k+1} = Fr_k \quad (12)$$

Construct an augmented state:

$$\begin{bmatrix} x_{k+1} \\ r_{k+1} \end{bmatrix} = \begin{bmatrix} A & 0 \\ 0 & F \end{bmatrix} \begin{bmatrix} x_k \\ r_k \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_k$$
$$X_{k+1} = TX_k + B_1 u_k \quad (13)$$

## Definition

Define a lyapunov function:

$$V(x_k, r_k) = V(X_k) = \frac{1}{2} X_k^T P X_k \quad (14)$$

Combining the infinite cost with lyapunov function yields a Bellman equation for augmented LQT:

$$X_k^T P X_k = X_k^T Q_1 X_k + u_k^T R u_k + X_{k+1}^T P X_{k+1} \quad (15)$$

# RL for Optimal Tracking Control (3)

Taking the time derivative of LQT Bellman, we obtain the LQT ARE

$$Q_1 - P + T^T PT - T^T PB_1(R + B_1^T PB_1)^{-1}B_1^T PT = 0 \quad (16)$$

The optimal policy is given by:

$$u_k = -K_1 X_k \quad (17)$$

with

$$K_1 = (R + B_1^T PB_1)^{-1} B_1^T PT \quad (18)$$

and

$$Q_1 = \begin{bmatrix} C^T Q C & -C^T Q \\ -Q C & Q \end{bmatrix} \quad (19)$$

# RL for Optimal Tracking Control (4)

From the LQT, obtain lyapunov equation

$$P = Q_1 + K_1^T R K_1 + (T - B_1 K_1)^T P (T - B_1 K_1) \quad (20)$$

Solve for  $P$  which satisfies (20)

---

## Algorithm 8 Offline Policy Iteration

---

- 1: **Initialization:** Select an admissible (stable) gain  $K_1^0$
  - 2: **repeat**
  - 3:   **Policy evaluation:**
  - 4:    $P^{j+1} = Q_1 + (K_1^j)^T R K_1^j + (T - B_1 K_1^j)^T P^{j+1} (T - B_1 K_1^j)$
  - 5:
  - 6:   **Policy improvement:**
  - 7:    $K_1^{j+1} = (R + B_1^T P^{j+1} B_1)^{-1} B_1^T P^{j+1} T$
  - 8: **until**  $P$  converges
- 

Extension to nonlinear case by Modares and Lewis (2014)  
**catch:** assume a perfect system model is

# RL for OTC with unknown system dynamics

- Use a temporal difference (TD) RL technique

## Definition

Q-function:

$$Q(X(k), u(k)) = \frac{1}{2} X(k)^T P X(k) \quad (21)$$

Combining (21) with Bellman yields:

$$\begin{aligned} Q(X(k), u(k)) &= \frac{1}{2} X(k)^T Q_1 X(k) + \frac{1}{2} u(k)^T R u(k) + \frac{1}{2} \gamma X^T(k+1) P X(k+1) \\ &= \frac{1}{2} X(k)^T Q_1 X(k) + \frac{1}{2} u(k)^T R u(k) + \frac{1}{2} \gamma (T X(k) + B_1 u(k))^T P (T X(k) + B_1 u(k)) \\ &= \frac{1}{2} \begin{bmatrix} X(k) \\ u(k) \end{bmatrix}^T \begin{bmatrix} Q_1 + \gamma T^T P T & \gamma T^T P B_1 \\ \gamma B_1^T P T & R + \gamma B_1^T P B_1 \end{bmatrix} \begin{bmatrix} X(k) \\ u(k) \end{bmatrix} \end{aligned} \quad (22)$$

# RL for OTC with unknown system dynamics (2)

By defining:

$$H = \begin{bmatrix} Q_1 + \gamma T^T P T & \gamma T^T P B_1 \\ \gamma B_1^T P T & R + \gamma B_1^T P B_1 \end{bmatrix} = \begin{bmatrix} H_{XX} & H_{XU} \\ H_{UX} & H_{UU} \end{bmatrix} \quad (23)$$

The optimal input is reached when  $\frac{\partial Q(X(k), u(k))}{\partial u(k)} = 0$  which yields:

$$u(k) = -H_{uu}^{-1} H_{ux} X(k) \quad (24)$$

Fortunately, one can apply PI to solve for  $H$ :

---

## Algorithm 9 Model-free Policy Iteration

---

- 1: **Initialization:** Select an initial admissible (stable) control input  $u = -K_1^0 X_0$
- 2: **repeat**
- 3:   **Policy evaluation:**  
4:      $Z(k)^T H^{j+1} Z(k) = X(k)^T Q_1 X(k) + (u(k)^j)^T R u(k)^j + Z(k+1)^T H Z(k+1)$
- 5:   **Policy improvement:**  
6:      $u^{j+1}(k) = -(H_{uu}^{-1})^{j+1} H_{ux}^{j+1} X(k)$
- 7: **until**  $H$  converges

# RL for OTC: Summary

## Advantages:

- Mathematically rigorous
- Proven to converge

## Disadvantages:

- non-linear RL-based OTC does not exist
- Needs a persistently exciting input, could be dangerous for the robot

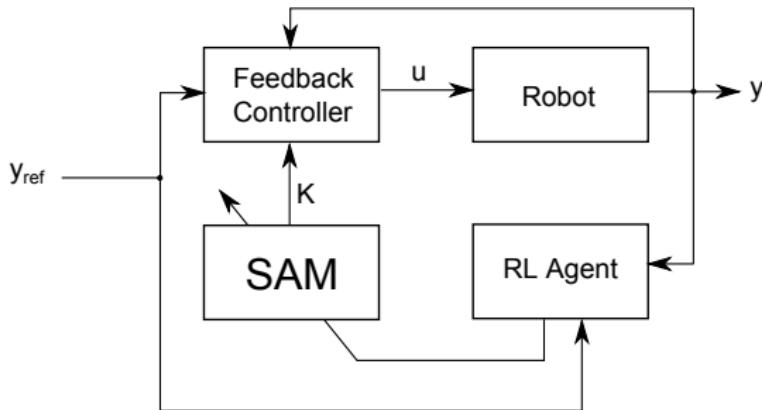
**Conclusion:** not the most suitable method for the problem!

# Dynamic Tuning via RL

- A feedback control e.g. PID performs well at a certain region
- For different regions, the controller needs to be retuned
- Solution: gain scheduling

# Dynamic Tuning via RL

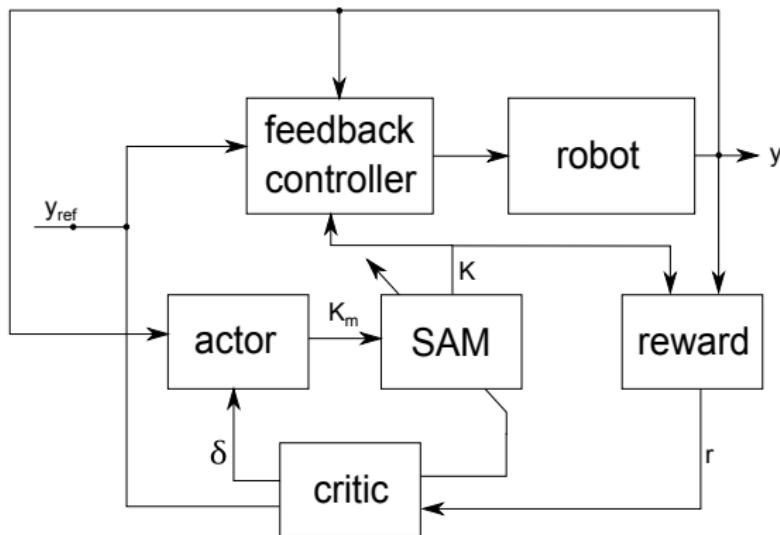
- A feedback control e.g. PID performs well at a certain region
- For different regions, the controller needs to be retuned
- Solution: gain scheduling
- The general diagram:



- Various types of RL algorithm can be used: actor-critic, SARSA, etc

# Dynamic Tuning via RL (2): Actor-critic

- Actor and critic can be parameterized with a basis function e.g. RBF neural network
- reference: Sedighizadeh and Rezazadeh (2008)



# Dynamic Tuning via RL (3): Summary

## Advantages:

- Intuitive and easier to implement

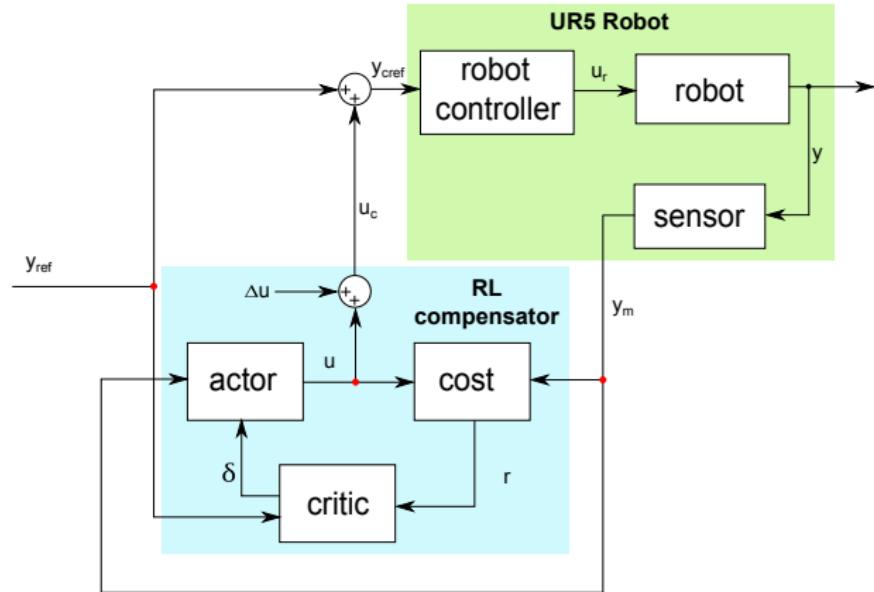
## Disadvantages:

- so far only applies to feedback controller
- feedback controller "waits" until error occurs, hence it is always late
- will not perform better due to this reason

**Conclusion:** not the most suitable method for the problem!

# Nonlinear Compensator using RL

- Acts as an additive input
- An actor critic RL is proposed



reference: Bayiz and Babuska (2010)

# Nonlinear Compensator using RL (2)

- cost function is defined to be similar to LQT

$$r_{k+1} = \rho(y_k, y_k^d, u_k) = (y_k^d - y_k)^T Q (y_k^d - y_k) + u_k^T R u_k \quad (25)$$

- Critic  $V(x_k, \theta_k)$  and actor  $\pi(x_k, \theta_{k-1})$  are parameterized by function approximators e.g. LLR, neural network, etc and respectively.
- Since for the error is repetitive for constant reference and speed, comparison (Iterative Learning Control) ILC is possible

# Nonlinear Compensator using RL (2)

- cost function is defined to be similar to LQT

$$r_{k+1} = \rho(y_k, y_k^d, u_k) = (y_k^d - y_k)^T Q (y_k^d - y_k) + u_k^T R u_k \quad (25)$$

- Critic  $V(x_k, \theta_k)$  and actor  $\pi(x_k, \theta_{k-1})$  are parameterized by function approximators e.g. LLR, neural network, etc and respectively.
- Since for the error is repetitive for constant reference and speed, comparison (Iterative Learning Control) ILC is possible

## Advantages:

- a relatively new approach, hence interesting for research
- the compensator does not depend directly on the nominal control, hence adding a degree of freedom in control design

# Nonlinear Compensator using RL (2)

- cost function is defined to be similar to LQT

$$r_{k+1} = \rho(y_k, y_k^d, u_k) = (y_k^d - y_k)^T Q (y_k^d - y_k) + u_k^T R u_k \quad (25)$$

- Critic  $V(x_k, \theta_k)$  and actor  $\pi(x_k, \theta_{k-1})$  are parameterized by function approximators e.g. LLR, neural network, etc and respectively.
- Since for the error is repetitive for constant reference and speed, comparison (Iterative Learning Control) ILC is possible

## Advantages:

- a relatively new approach, hence interesting for research
- the compensator does not depend directly on the nominal control, hence adding a degree of freedom in control design

## Disadvantages:

- Not mathematically as rigorous as RL-based optimal control
- Proof of stability does not exist yet

# Next Subsection

## 1 Preliminaries

Basics of Reinforcement Learning (RL) & Tracking Control

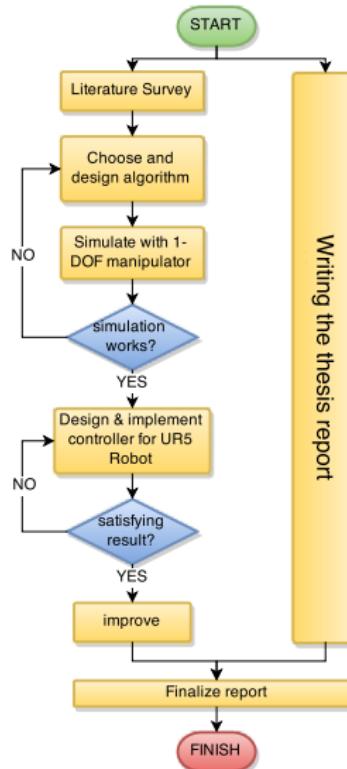
Experimental Setup: 3D Printing Robot System

## 2 Literature Survey Result

RL for Tracking: A Survey

Research Plan & Conclusion

# Research Plan



# Conclusion

- The RL-based additive compensator is chosen as the most promising solution
- Simulation on 1-DoF arm will be performed before starting implementation on UR5 robot
- Possible comparison with ILC

# End of presentation

Thank you for the attention

## Question?

### References:

- Y. Efe Bayiz and Robert Babuska. Nonlinear disturbance compensation and. reference tracking via reinforcement. learning with fuzzy approximators. *19th IFAC World Congress*, 2010.
- L.A. Brujeni, Jong Min Lee, and S.L. Shah. Dynamic tuning of pi-controllers based on model-free reinforcement learning methods. In *Control Automation and Systems (ICCAS), 2010 International Conference on*, pages 453–458, Oct 2010.
- M. de Gier. *UR5 MPC Program*. DCSC, TU Delft, 2014.
- B. Kiumarsi and F.L. Lewis. Actor-critic-based optimal tracking for partially unknown nonlinear discrete-time systems. *Neural Networks and Learning Systems, IEEE Transactions on*, PP(99):1–1, 2014.
- Hamidreza Modares and Frank L. Lewis. Optimal tracking control of nonlinear partially-unknown constrained-input systems using integral reinforcement learning. *Automatica*, 50(7):1780 – 1792, 2014.
- M. Sedighizadeh and A. Rezazadeh. Adaptive pid controller based on reinforcement learning for wind turbine control. In *World Academy of Science, Engineering and Technology Vol:2*, 2008.