Information Sciences 682 (2024) 121283



Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins



# Explainable data-driven Q-learning control for a class of discrete-time linear autonomous systems

Adolfo Perrusquía *, Mengbang Zou, Weisi Guo

*School of Aerospace, Transport and Manufacturing, Cranfield University, Bedford MK43 0AL, UK*

ARTICLE INFO

*Keywords:*
Q-learning
State-transition function
Explainable Q-learning (XQL)
Control policy

ABSTRACT

Explaining what a reinforcement learning (RL) control agent learns play a crucial role in the safety critical control domain. Most of the approaches in the state-of-the-art focused on imitation learning methods that uncover the hidden reward function of a given control policy. However, these approaches do not uncover what the RL agent learns effectively from the agent-environment interaction. The policy learned by the RL agent depends in how good the state transition mapping is inferred from the data. When the state transition mapping is wrongly inferred implies that the RL agent is not learning properly. This can compromise the safety of the surrounding environment and the agent itself. In this paper, we aim to uncover the elements learned by data-driven RL control agents in a special class of discrete-time linear autonomous systems. Here, the approach aims to add a new explainable dimension to data-driven control approaches to increase their trust and safe deployment. We focus on the classical data-driven Q-learning algorithm and propose an explainable Q-learning (XQL) algorithm that can be further expanded to other data-driven RL control agents. Simulation experiments are conducted to observe the effectiveness of the proposed approach under different scenarios using several discrete-time models of autonomous platforms.

## 1. Introduction

Explaining reinforcement learning (RL) decisions remains a key challenge in explainable artificial intelligence (AI) and safety critical control domains [1,2]. The black-box nature of RL decisions hinders the interpretation and understanding of why a particular control policy is obtained and what does the RL agent is learning or inferring [3]. Despite diverse research has been conducted to explain the behaviour of supervised machine learning decisions [4], there is a gap in how we can explain RL decisions [5]. This is a challenge in current autonomous systems applications for navigation or tracking that involve drones or autonomous ground vehicles. Here, the level of autonomy of these platforms is high and require that the embedded data-driven algorithms are trustworthy for their safe deployment.

In view of this, the incorporation of explainable tools in the RL design is actually a crucial requirement in safety-critical domains such as transport and smart living, where autonomous decisions can have life threatening consequences. The state-of-the-art has adopted similar explainable techniques as those used in supervised learning models. Here, the obtained explanations highlight the mapping from states to actions, but without explaining what does the RL agent is effectively learning.

* Corresponding author.
*E-mail address:* adolfo.perrusquia-guzman@cranfield.ac.uk (A. Perrusquía).

https://doi.org/10.1016/j.ins.2024.121283
Received 3 May 2024; Received in revised form 20 July 2024; Accepted 31 July 2024

Available online 5 August 2024
0020-0255/© 2024 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

We can distinguish two main families of explanations given by gradient-based methods [6,7] and perturbation-based methods [8,9]. Both methods aim to uncover the local importance weight [10] of each input feature by either altering the inputs or computing the derivative of outputs against the inputs [11]. These explainable methods allow the construction of a low-complexity model that locally approximates the machine learning algorithm. These methods are effective on supervised-learning architectures to detect the feature importance of the predictive model. However, the control policy of a RL agent is not only a mapping between inputs to outputs, in fact, it is a mixture of different factors ranging from the system dynamics to the learning mechanisms of the RL agent.

This challenge is also observed in data-driven reinforcement learning control approaches where it is unknown what the RL control agent learns from the data either on-line or off-line [12]. Therefore, there is a need to increase the trust and safety of data driven control approaches similarly to what it has been developed so far for AI algorithms. So, in this paper we aim to tackle this challenge by developing a novel explainable RL control approach for a class of discrete-time linear autonomous systems that uncovers the elements inferred from the data by RL control agents which can be used for further explanations.

### 1.1. Related work

Some of the most recent approaches to explain RL decisions is by means of inverse reinforcement learning (IRL) and imitation learning (IL) [13–15] methods. The key idea is to uncover the reward function that is used to derive the control policy. This is particularly efficient in experience transference applications for safety critical control. Despite the reward function offers the most succinct definition of the task that a RL agent aims to achieve, there are some specific challenges that hinders the accurate interpretations of RL decisions. These challenges are: i) multiple reward functions can obtain the same control policy [16] which can lead to biased explanations and ii) IRL methods cannot uncover what does the RL agent is effectively learning from the collected data.

Other approach incorporates the concept of intended outcomes [4] to explain actions preference of RL agents. The key idea is to either decompose scalar reward functions or Q functions in terms of sub-reward goals and a state-action grid. This is similar of what we are interested to implement in this paper to understand the action selection in a specific state. However, this idea has not been explored in the data-driven control domain since the designers only care of the final control policy without considering other learning elements.

One of the main elements in any RL architecture is the exploration step [17,18] which is used to discover the state-transition function properties. This is a crucial factor to ensure that the learned policy is the one that satisfies the reward requirements in an infinite horizon [19]. Therefore, one key element to explain RL decisions can be regarded as how good it infers the state-transition function of the agent-environment interaction. State-transition function learning has been studied as a separate model to the RL using any system identification technique based in either least-squares (LS) or gradient rules [20,21]. Dynamic mode decomposition [22] has been used to uncover a discrete-time linear dynamics of diverse autonomous systems from the data which preserve dynamic properties in a low-dimensional representation. However, due to the independent nature of these models, we cannot affirm that the estimated model is the one that is learned by the RL agent since both methods have a different learning objective. Other approaches, incorporate an additional learning module in the RL architecture to identify the state transition function dynamics [23,24] and facilitate the development of the task. This helps to understand better what does the RL agent is learning with the compromise of increasing the computational complexity of the algorithm. However, these approaches are more oriented to a physics-informed approach that constraints the agent's learning in the direction of the learned state-transition model. Similarly, physics-informed approaches have been developed to provide physical information to data-driven methods that enables better inference and prediction capabilities [22,25].

### 1.2. Contributions and notations

In view of the above, this paper provides the first attempt to provide explanations of data-driven reinforcement learning control policies of a class of discrete-time linear autonomous systems. The approach is inspired in what state-transition dynamics the RL agent learns in each iteration and how it shapes the final control policy. To this end, we focus on the classical Q-learning algorithm (which can be expanded to other RL architectures) and propose a novel explainable Q-learning (XQL) algorithm. The proposed XQL approximately infers the state-transition dynamics that is learned by the Q-learning algorithm. This permits the construction of explanations that determine why a particular policy is learned based on the inner learning elements of the Q-learning algorithm. In contrast to standard and independent identification algorithms, the proposed approach aims to discover what does the reinforcement learning algorithm learns from the data to explain why a particular control policy is obtained. Fig. 1 depicts the high-level diagram of the proposed XQL.

The main contributions of the proposed approach are:

1. A natural state-transition inference algorithm is developed based on the outputs of the Q-learning algorithm.
2. A learning hierarchy is established to identify the main elements that are inferred from the data and how they are interconnected.
3. To best of our knowledge, this approach gives the first attempt for providing explanations of what the reinforcement learning agent is learning from the data for a class of discrete-time linear autonomous systems.

Throughout this paper, $\mathbb{N}$, $\mathbb{R}$, $\mathbb{R}^n$, $\mathbb{R}^{n\times m}$ denote the spaces of natural numbers, real numbers, real $n$-vectors, and real $n\times m$-matrices, respectively; $\boldsymbol{I}_n \in \mathbb{R}^{n\times n}$ denotes an $n\times n$ identity matrix, $\mathcal{L}_\infty$ defines the set of bounded functions, $\lambda_{\min}(\boldsymbol{A})$ and $\lambda_{\max}(\boldsymbol{A})$ denote the minimum and maximum eigenvalues of matrix $\boldsymbol{A}$, respectively; $\otimes$ and vech($\boldsymbol{A}$) defines the symmetric Kronecker product

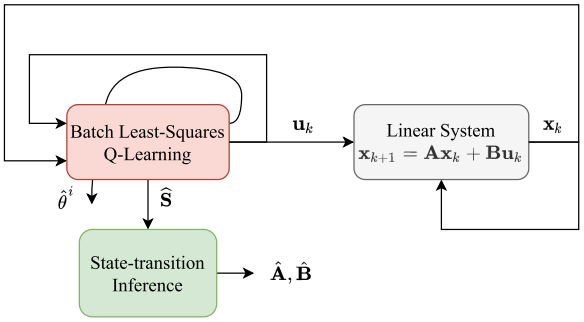



**Fig. 1.** Explainable Q-learning (XQL) scheme.

and the half vectorization of matrix $\boldsymbol{A}$, $\|\boldsymbol{x}\|$ and $\|\boldsymbol{A}\|$ define the Euclidean and induced norms of a vector and matrix, respectively, where $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and $n, m \in \mathbb{N}$.

## 2. The optimal control problem background

Consider the following linear time invariant discrete-time system,

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k, \qquad \boldsymbol{x}(t_0) = \boldsymbol{x}_0, \tag{1}$$

where $\boldsymbol{x}_k \in \mathbb{R}^n$ is a measurable state vector, $\boldsymbol{u}_k \in \mathbb{R}^m$ is the control input and $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{B} \in \mathbb{R}^{n\times m}$ define the dynamic of the system which are assumed to be unknown, and $\boldsymbol{x}_0 \in \mathbb{R}^n$ is the initial condition. It is assumed that the pair $(\boldsymbol{A}, \boldsymbol{B})$ is controllable.

**Assumption 1.** In this paper, it is assumed that matrix $\boldsymbol{A}$ is known and has an upper triangular structure. This structure is common in several autonomous platforms including aerial and ground vehicles.

The main objective of the optimal control problem is to find an optimal control policy $\boldsymbol{u}$ that minimizes the following infinite horizon cost index [26],

$$J = \sum_{i=k}^{\infty} r_i = \sum_{i=k}^{\infty} (\boldsymbol{x}_i^\top \boldsymbol{Q}\boldsymbol{x}_i + \boldsymbol{u}_i^\top \boldsymbol{R}\boldsymbol{u}_i), \tag{2}$$

where $\boldsymbol{Q} = \boldsymbol{Q}^\top \succeq 0 \in \mathbb{R}^{n\times n}$ and $\boldsymbol{R} = \boldsymbol{R}^\top \succ 0 \in \mathbb{R}^{m\times m}$ are weight matrices that impose a desired behaviour that the user aims to inject to system (1). $r_i$ is known as the reward, objective or utility function that we aim to optimize in an infinite horizon.

### 2.1. The linear quadratic regulator

For a given control policy $\boldsymbol{u}_k = \boldsymbol{\mu}(\boldsymbol{x}_k)$, the associated value function [27] is given by

$$V(\boldsymbol{x}_k) = \sum_{i=k}^{\infty} (\boldsymbol{x}_i^\top \boldsymbol{Q}\boldsymbol{x}_i + \boldsymbol{u}_i^\top \boldsymbol{R}\boldsymbol{u}_i). \tag{3}$$

The value function (3) can be written as the Bellman equation as follows,

$$V(\boldsymbol{x}_k) = \boldsymbol{x}_k^\top \boldsymbol{Q}\boldsymbol{x}_k + \boldsymbol{u}_k^\top \boldsymbol{R}\boldsymbol{u}_k + \sum_{i=k+1}^{\infty} (\boldsymbol{x}_i^\top \boldsymbol{Q}\boldsymbol{x}_i + \boldsymbol{u}_i^\top \boldsymbol{R}\boldsymbol{u}_i)$$

$$V(\boldsymbol{x}_k) = r_k + V(\boldsymbol{x}_{k+1}). \tag{4}$$

Assume the value function is quadratic in the state, i.e.,

$$V(\boldsymbol{x}_k) = \boldsymbol{x}_k^\top \boldsymbol{P}\boldsymbol{x}_k, \tag{5}$$

for some kernel matrix $\boldsymbol{P} = \boldsymbol{P}^\top \succ 0 \in \mathbb{R}^{n\times n}$ which is the solution of the following discrete Algebraic Ricatti equation (DARE),

$$\boldsymbol{A}^\top \boldsymbol{P}\boldsymbol{A} + \boldsymbol{Q} - \boldsymbol{A}^\top \boldsymbol{P}\boldsymbol{B}(\boldsymbol{R} + \boldsymbol{B}^\top \boldsymbol{P}\boldsymbol{B})^{-1} \boldsymbol{B}^\top \boldsymbol{P}\boldsymbol{A} = \boldsymbol{P}. \tag{6}$$

Then the following Hamilton-Jacobi-Bellman (HJB) equation is derived by inserting (1) in (4) and using the optimal value function definition (5),

$$H(\boldsymbol{x}_k, \boldsymbol{u}_k) = (\boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k)^\top \boldsymbol{P}(\boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k) - \boldsymbol{x}_k^\top \boldsymbol{P}\boldsymbol{x}_k$$

$$+\boldsymbol{x}_k^\top \boldsymbol{Q}\boldsymbol{x}_k + \boldsymbol{u}_k^\top \boldsymbol{R}\boldsymbol{u}_k = 0. \tag{7}$$

The $Q$-value function can be defined as the sum of the optimal value function (5) and the HJB (7) as $Q(\boldsymbol{x}_k,\boldsymbol{u}_k) = V(\boldsymbol{x}_k) + H(\boldsymbol{x}_k,\boldsymbol{u}_k)$. Then, the $Q$-value function can be written quadratic in terms of the state $\boldsymbol{x}_k$ and control $\boldsymbol{u}_k$ as

$$Q(\boldsymbol{x}_k,\boldsymbol{u}_k) := \boldsymbol{z}_k^\top \boldsymbol{S}\boldsymbol{z}_k = \begin{bmatrix}\boldsymbol{x}_k\\ \boldsymbol{u}_k\end{bmatrix}^\top \begin{bmatrix}\boldsymbol{S}_{xx} & \boldsymbol{S}_{xu}\\ \boldsymbol{S}_{xu}^\top & \boldsymbol{S}_{uu}\end{bmatrix}\begin{bmatrix}\boldsymbol{x}_k\\ \boldsymbol{u}_k\end{bmatrix}, \tag{8}$$

where $\boldsymbol{z}_k = [\boldsymbol{x}_k^\top, \boldsymbol{u}_k^\top]^\top \in \mathbb{R}^{n+m}$ and $\boldsymbol{S} = \boldsymbol{S}^\top > 0 \in \mathbb{R}^{(n+m)\times(n+m)}$ with $\boldsymbol{S}_{xx} = \boldsymbol{A}^\mathsf{T}\boldsymbol{P}\boldsymbol{A} + \boldsymbol{Q}$, $\boldsymbol{S}_{xu} = \boldsymbol{A}^\mathsf{T}\boldsymbol{P}\boldsymbol{B}$ and $\boldsymbol{S}_{uu} = \boldsymbol{B}^\mathsf{T}\boldsymbol{P}\boldsymbol{B} + \boldsymbol{R}$.

The optimal control policy $\boldsymbol{u}_k^*$ is obtained by differentiating the $Q$-value function (8) with respect to $\boldsymbol{u}_k$ and equalling to zero as

$$\begin{aligned}\boldsymbol{u}_k^* &= \arg\min_{\boldsymbol{u}_k\in\mathbb{R}^m} Q(\boldsymbol{x},\boldsymbol{u}_k)\\ &= -\boldsymbol{S}_{uu}^{-1}\boldsymbol{S}_{xu}^\top \boldsymbol{x}_k := -\boldsymbol{K}\boldsymbol{x}_k.\end{aligned} \tag{9}$$

The control policy (9) is the optimal control policy also known as the linear quadratic regulator (LQR) control [28]. This control policy is explainable by definition since it is based on the knowledge of the system dynamics $\boldsymbol{A}$ and $\boldsymbol{B}$. Here, the explanations can be easy derived by analysing how the policy changes by varying either $\boldsymbol{A}$, $\boldsymbol{B}$ or both.

### 2.2. Batch least-squares Q-learning

When matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are unknown, then a model-free algorithm can be used to derive the optimal control policy. This algorithm is known as Q-learning which has been well-studied in previous works [29,30]. The idea consists in estimating the elements of matrix $\boldsymbol{S}$ by using the following approximation,

$$\begin{aligned}\widehat{Q}(\boldsymbol{x}_k,\boldsymbol{u}_k) &= \boldsymbol{z}_k^\top \widehat{\boldsymbol{S}}\boldsymbol{z}_k,\\ \widehat{Q}(\boldsymbol{x}_k,\boldsymbol{u}_k) &= (\boldsymbol{z}_k^\top \otimes \boldsymbol{z}_k^\top)\mathrm{vech}(\widehat{\boldsymbol{S}}) := \boldsymbol{\phi}_k^\top \widehat{\theta}^i,\end{aligned} \tag{10}$$

where $\widehat{\boldsymbol{S}} = \widehat{\boldsymbol{S}}^\top > 0 \in \mathbb{R}^{(n+m)\times(n+m)}$ is an estimate of matrix $\boldsymbol{S}$, $\boldsymbol{\phi}_k = (\boldsymbol{z}_k \otimes \boldsymbol{z}_k)$ is the set of basis functions, and $\widehat{\theta}^i \in \mathbb{R}^{\frac{1}{2}(n+m)(n+m+1)}$ defines the upper triangular elements of matrix $\widehat{\boldsymbol{S}}$ which are updated each iteration $i$ of the Q-learning algorithm. Here, both the $Q$-function (8) and its approximation (10) verify the Bellman equation (4). For Q-learning we have

$$\begin{aligned}\widehat{Q}(\boldsymbol{x}_k,\boldsymbol{u}_k) &= r_k + \widehat{Q}(\boldsymbol{x}_{k+1},\boldsymbol{u}_{k+1})\\ (\boldsymbol{\phi}_k - \boldsymbol{\phi}_{k+1})^\top \widehat{\theta}^i &= r_k.\end{aligned} \tag{11}$$

Assume we collect $\kappa \geq w = \frac{1}{2}(n+m)(n+m+1)$ samples of $(\boldsymbol{\phi}_k - \boldsymbol{\phi}_{k+1})$ and $r_k$ respectively, to construct the following matrices

$$\boldsymbol{\Phi} = \begin{bmatrix} | \\ \boldsymbol{\phi}_i - \boldsymbol{\phi}_{i+1} \\ | \end{bmatrix} \in \mathbb{R}^{w\times\kappa}, \qquad \boldsymbol{r} = \begin{bmatrix} | \\ r_i \\ | \end{bmatrix} \in \mathbb{R}^{\kappa}.$$

Then, the parameter estimates $\widehat{\theta}^i$ of the $Q$-function are estimated by solving the following batch least-squares (BLS) rule in each step of the Q-learning algorithm,

$$\widehat{\theta}^i = (\boldsymbol{\Phi}\boldsymbol{\Phi}^\mathsf{T})^{-1}\boldsymbol{\Phi}\boldsymbol{r}, \tag{12}$$

where the elements of $\widehat{\boldsymbol{S}}$ can be recovered by the symmetric matrization as $\widehat{\boldsymbol{S}} = \mathrm{mat}(\widehat{\theta}^i)$ The optimal/near optimal control policy is given by

$$\boldsymbol{u}_k^* = -\widehat{\boldsymbol{S}}_{uu}^{-1}\widehat{\boldsymbol{S}}_{xu}\boldsymbol{x}_k = -\widehat{\boldsymbol{K}}\boldsymbol{x}_k. \tag{13}$$

Convergence of the BLS Q-learning update rule is given in [31]. The computational complexity of the algorithm is $\mathcal{O}(i\kappa w^2)$, which is associated to the least-squares rule iterated $i$ times. The BLS Q-learning algorithm is summarized in Algorithm 1.

**Remark 1.** Learning optimal controllers from data is not a simple task. Traditional controllers use knowledge of the system dynamics to compute optimal controller or heuristically tune controllers until a performance is achieved. This is not the philosophy of data-driven control approaches, which aims to discover control laws from the data that preserve the dynamic properties and expert's requirements given a cost function. Q-learning is one of the simplest optimal data-driven controllers that fulfil the objectives of data-driven control approaches. Therefore, the motivation of using this kind of controllers is to maintain optimal control performances even the dynamic changes and without human intervention.

**Algorithm 1** BLS Q-learning algorithm.

Initialize with an admissible control policy $\boldsymbol{u}_k^0$ and add a probing noise $\mu$ signal for excitation, stopping threshold $\epsilon$, and set $i=0$
**for** $k=0,1,\ldots$ **do**
  Collect data $\boldsymbol{z}_k$ and $r_k$ to construct the matrices $\boldsymbol{\Phi}$ and $\boldsymbol{r}$.
  **if** rank$(\boldsymbol{\Phi})=\frac{1}{2}(n+m)(n+m+1)$ **then**
    Set $i\leftarrow i+1$.
    Solve the BLS rule (12) and construct $\widehat{\boldsymbol{S}}$ from $\widehat{\boldsymbol{\theta}}^i$.
    Update control policy $\boldsymbol{u}_k^i$ using (13).
    Stop algorithm until convergence $\|\widehat{\boldsymbol{\theta}}^i-\widehat{\boldsymbol{\theta}}^{i-1}\|\leq\epsilon$
  **else**
    Go to step 3.
  **end if**
**end for**

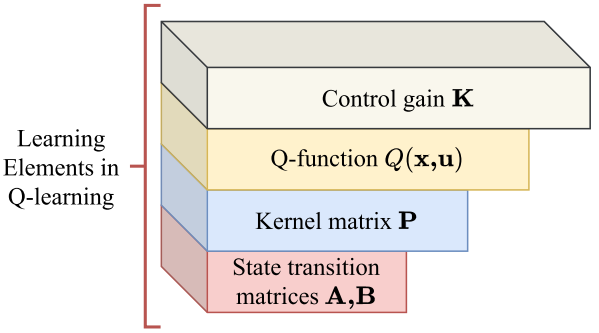



**Fig. 2.** Elements learned by the Q-learning control algorithm.

## 3. Explainable Q-learning (XQL)

The main goal of the Q-learning algorithm is to learn an optimal control policy from the data collected from the interaction between the agent and the environment [32]. Q-learning is based on a value iteration algorithm whose aim is to find the optimal state-action value function (also known as $Q$-function) to derive the optimal control policy. Here, both the control policy and value function play a fundamental role in the observed performance of the RL agent. However, there are hidden elements that are not explicitly learned by the Q-learning algorithm, instead they are inferred from the data to obtain both the optimal value function and control policy. These hidden elements are the solution of the DARE (6) given by the kernel matrix $\boldsymbol{P}$ and the state-transition dynamics $\boldsymbol{A},\boldsymbol{B}$ that obey the agent-environment interaction. These elements establish levels of hierarchy that allow the construction of explanations of the complete learning process. The elements learned by the Q-learning control algorithm and most of the data-driven reinforcement learning control agents are depicted in Fig. 2. Here, the control gain $\boldsymbol{K}$ is the one that depends highly on the learned $Q$-function. Equivalently, the precision of the learned $Q$-function depends in how good the kernel matrix $\boldsymbol{P}$ and the state-transition model $\boldsymbol{A},\boldsymbol{B}$ are inferred from observational data.

In this paper, we hypothesize that the final control policy depends highly in how good the state transition function is inferred. Here, the inference of the state-transition model affects the complete learning chain which covers the kernel matrix $\boldsymbol{P}$, the $Q$-function $Q(\boldsymbol{x},\boldsymbol{u})$ and finally the optimal control gain $\boldsymbol{K}$. To this end, we aim to provide explanations of what the Q-learning algorithm is learning and the mechanisms to modify its performance into a desired one. Parameter identification algorithms [33] such as least-squares, dynamic mode decomposition, and gradient rules can deal effectively with the state transition dynamics identification. However, they cannot ensure that the estimated model is the one that is learned by the RL control algorithm. This is evident since Q-learning learns a coupled matrix $\widehat{\boldsymbol{S}}$ whose elements are quadratic in terms of $\widehat{\boldsymbol{A}}$ and $\widehat{\boldsymbol{B}}$. Therefore, multiple combinations between the elements of $\widehat{A}_{ij}$ and $\widehat{B}_{ik}$ can yield the same matrix $\widehat{\boldsymbol{S}}$.

In this paper, we propose a novel explainable Q-learning (XQL) algorithm to infer what state-transition function model is learned by the Q-learning algorithm. The proposed approach maintains the traditional Q-learning structure, but it incorporates a new module to infer the state transition dynamics using a matrix decomposition approach of the learned matrix $\widehat{\boldsymbol{S}}$ and a simple heuristic rule. This will allow to uncover the state-transition model in each iteration $i$ of the Q-learning algorithm and avoid the biased conclusions of independent identification algorithms.

### 3.1. State-transition inference

It is well known that the $Q$-function under the optimal control policy $u^*$ is equivalent to the quadratic form of the value function (5). Equivalently, from the $Q$-function approximation (10) we obtain the following relation

$$\widehat{\boldsymbol{P}}=\begin{bmatrix}\boldsymbol{I}_n & -\widehat{\boldsymbol{K}}^\top\end{bmatrix}\widehat{\boldsymbol{S}}\begin{bmatrix}\boldsymbol{I}_n & -\widehat{\boldsymbol{K}}^\top\end{bmatrix}^\top, \tag{14}$$

where $\widehat{\boldsymbol{P}} = \widehat{\boldsymbol{P}}^{\top} \succ 0$ is an estimate of the real kernel matrix $\boldsymbol{P}$. From the estimated matrix $\widehat{\boldsymbol{S}}$ we have that

$$\widehat{\boldsymbol{A}}^{\top}\widehat{\boldsymbol{P}}\widehat{\boldsymbol{A}} = \widehat{\boldsymbol{S}}_{xx} - \boldsymbol{Q}. \tag{15}$$

The eigendecomposition [34] of matrix $(\widehat{\boldsymbol{S}}_{xx} - \boldsymbol{Q})$ is

$$(\widehat{\boldsymbol{S}}_{xx} - \boldsymbol{Q}) = \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^{\top}, \tag{16}$$

where $\boldsymbol{V} \in \mathbb{R}^{n\times n}$ is an orthonormal matrix composed of the eigenvectors of $(\widehat{\boldsymbol{S}}_{xx} - \boldsymbol{Q})$, and $\boldsymbol{D} \in \mathbb{R}^{n\times n}$ is a diagonal matrix whose entries are given by the eigenvalues of $(\widehat{\boldsymbol{S}}_{xx} - \boldsymbol{Q})$. Then, we can equivalently write (15) as

$$\begin{aligned} \boldsymbol{V}^{\top}\widehat{\boldsymbol{A}}^{\top}\widehat{\boldsymbol{P}}\widehat{\boldsymbol{A}}\boldsymbol{V} &= \boldsymbol{D} \\ \boldsymbol{W}^{\top}\widehat{\boldsymbol{P}}\boldsymbol{W} &= \boldsymbol{D}, \end{aligned} \tag{17}$$

where $\boldsymbol{W} = \widehat{\boldsymbol{A}}\boldsymbol{V} \in \mathbb{R}^{n\times n}$. We can additionally find the eigendecomposition of the learned kernel matrix $\widehat{\boldsymbol{P}}$ to obtain,

$$\begin{aligned} \boldsymbol{W}^{\top}\boldsymbol{U}\boldsymbol{\mathcal{D}}\boldsymbol{U}^{\top}\boldsymbol{W} &= \boldsymbol{D}, \\ \boldsymbol{T}^{\top}\boldsymbol{\mathcal{D}}\boldsymbol{T} &= \boldsymbol{D} \end{aligned} \tag{18}$$

where $\boldsymbol{U} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{n\times n}$ denote the matrices of eigenvectors and eigenvalues of $\widehat{\boldsymbol{P}}$, and $\boldsymbol{T} = \boldsymbol{U}^{\top}\boldsymbol{W} \in \mathbb{R}^{n\times n}$. The last equality of (18) means that $\boldsymbol{D}$ and $\boldsymbol{\mathcal{D}}$ are congruent matrices given a transformation matrix $\boldsymbol{T}$. Due to the relation in (18) we can define $\boldsymbol{T}$ as a diagonal matrix whose entries are given by

$$T_{ii} = \pm\sqrt{\frac{D_{ii}}{\mathcal{D}_{ii}}}. \tag{19}$$

The sign of each element of $\boldsymbol{T}$ can be easily computed as the sign of the diagonal elements of $\boldsymbol{U}^{\top}\boldsymbol{V}$. Therefore, the learned matrix $\widehat{\boldsymbol{A}}$ can be computed as

$$\widehat{\boldsymbol{A}} = \boldsymbol{U}\boldsymbol{T}\boldsymbol{V}^{\top}. \tag{20}$$

The approximate matrix $\widehat{\boldsymbol{B}}$ can be obtained from $\widehat{\boldsymbol{S}}$ by using the following relation,

$$\widehat{\boldsymbol{B}} = \left(\widehat{\boldsymbol{A}}^{\top}\widehat{\boldsymbol{P}}\right)^{-1}\widehat{\boldsymbol{S}}_{xu}. \tag{21}$$

Notice that the estimated matrices $\widehat{\boldsymbol{A}}$ and $\widehat{\boldsymbol{B}}$ fulfil the DARE (6),

$$\widehat{\boldsymbol{A}}^{\top}\widehat{\boldsymbol{P}}\widehat{\boldsymbol{A}} + \boldsymbol{Q} - \widehat{\boldsymbol{A}}^{\top}\widehat{\boldsymbol{P}}\widehat{\boldsymbol{B}}(\boldsymbol{R} + \widehat{\boldsymbol{B}}^{\top}\widehat{\boldsymbol{P}}\widehat{\boldsymbol{B}})^{-1}\widehat{\boldsymbol{B}}^{\top}\widehat{\boldsymbol{P}}\widehat{\boldsymbol{A}} = \widehat{\boldsymbol{P}}. \tag{22}$$

This means that the estimated matrices $\widehat{\boldsymbol{A}}$ and $\widehat{\boldsymbol{B}}$ under the same weight matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ yields the same kernel matrix $\widehat{\boldsymbol{P}}$. Notice that $\widehat{\boldsymbol{A}}$ and $\widehat{\boldsymbol{B}}$ are not necessarily equivalent to $\boldsymbol{A}$ and $\boldsymbol{B}$ due to the diagonal structure assumption of matrix $\boldsymbol{T}$. This means we have a loss of information in the approximation which slightly modifies the closed-loop performance.

To clearly identify if the Q-learning algorithm is effectively learning the appropriate dynamics from the data, we use the following heuristic based on the spectral norm

$$\mathcal{H} = \|\boldsymbol{A}\| - \|\widehat{\boldsymbol{A}}\| \leq \varepsilon_A, \tag{23}$$

where $\varepsilon_A > 0$ is a small threshold associated to the loss of information in the inference algorithm. This means that if the heuristic is satisfied then the Q-learning algorithm is effectively learning the state transition dynamics. In this case and without loss of generality, we can use $\boldsymbol{A}$ instead of $\widehat{\boldsymbol{A}}$ and compute directly $\boldsymbol{B}$ as

$$\boldsymbol{B} = \left(\boldsymbol{A}\widehat{\boldsymbol{P}}\right)^{-1}\widehat{\boldsymbol{S}}_{xu}. \tag{24}$$

Conversely, if the heuristic is not satisfied means that the Q-learning algorithm is inferring a wrong dynamics and require either more episodes or increase the richness of the collected data, i.e., to improve the proving noise signal or excitation signal.

### 3.2. Approximation error due to the loss of information

The following theorem establishes that the approximation error between the real and estimated state transition models yields trajectories with a small bounded error which decreases as $k \to \infty$.

**Theorem 1.** *Consider two asymptotic stable closed-loop systems (eigenvalues lie in the unit disk) constructed from the real and the estimated matrices,*

$$\boldsymbol{x}_{k+1} = (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K})\boldsymbol{x}_k := \boldsymbol{A}_c\boldsymbol{x}_k$$

$$\boldsymbol{y}_{k+1} = (\widehat{\boldsymbol{A}} - \widehat{\boldsymbol{B}}\widehat{\boldsymbol{K}})\boldsymbol{y}_k := \widehat{\boldsymbol{A}}_c \boldsymbol{y}_k = (\boldsymbol{A}_c + \boldsymbol{\Delta A}_c)\boldsymbol{y}_k,$$

*where $\boldsymbol{y}_k \in \mathbb{R}^n$ defines the states of the estimated model and $\boldsymbol{\Delta A}_c \in \mathbb{R}^{n\times n}$ is an approximation error of the closed-loop state transition model. Define the estimation error as $\boldsymbol{e}_k = \boldsymbol{x}_k - \boldsymbol{y}_k$ with its corresponding estimation error dynamics,*

$$\boldsymbol{e}_{k+1} = \boldsymbol{A}_c \boldsymbol{e}_k - \boldsymbol{\Delta A}_c \boldsymbol{y}_k. \tag{25}$$

*Assume there are two symmetric and positive definite matrices $\boldsymbol{M} = \boldsymbol{M}^\top \succ 0 \in \mathbb{R}^{n\times n}$ and $\boldsymbol{N} = \boldsymbol{N}^\top \succ 0 \in \mathbb{R}^{n\times n}$ that fulfil the Lyapunov equation*

$$\boldsymbol{A}_c^\top \boldsymbol{M} \boldsymbol{A}_c - \boldsymbol{M} + \boldsymbol{N} = \boldsymbol{0}_n. \tag{26}$$

*Then, the estimation error $\boldsymbol{e}_k$ remains bounded within a small set of radius $\mu$ as $k \to \infty$.*

The following two definitions are required before we prove Theorem 1.

**Definition 1.** The closed-loop error dynamics (25) is input-to-state stable (ISS) [35] if there exists a $\mathcal{KL}$-function $\beta$ and a $\mathcal{K}$-function $\gamma$ such that for each bounded input $\boldsymbol{y}_k \in \mathcal{L}_\infty$ and each initial condition $\boldsymbol{e}_0 \in \mathbb{R}^n$, it holds that

$$\|\boldsymbol{e}_k\| \leq \beta(\|\boldsymbol{e}_0\|) + \gamma(\|\boldsymbol{y}_k\|).$$

**Definition 2.** A smooth function $L_k = L(\boldsymbol{e}_k) \geq 0 \in \mathbb{R}$ is called a ISS-Lyapunov function for (25) if there exist a $\mathcal{K}_\infty$-functions $\alpha_1(\cdot)$, $\alpha_2(\cdot)$, and $\alpha_3(\cdot)$, and $\mathcal{K}$-function $\alpha_4(\cdot)$, such that for any $\boldsymbol{r} \in \mathbb{R}^n$, each $\boldsymbol{e}_k$ and $\boldsymbol{y}_k$ the following holds,

$$\alpha_1(\|\boldsymbol{r}\|) \leq L_k(\boldsymbol{r}) \leq \alpha_2(\|\boldsymbol{r}\|)$$
$$L_{k+1} - L_k \leq -\alpha_3(\|\boldsymbol{e}_k\|) + \alpha_4(\|\boldsymbol{y}_k\|).$$

The first definition establishes that the behaviour of the closed-loop error trajectories remains bounded when the inputs $\boldsymbol{y}_k$ are bounded. This statement holds in the proposed approach since the input signals $\boldsymbol{y}_k$ belong to a bounded and asymptotic stable system. With the above definitions we can proceed with the proof of Theorem 1.

**Proof 1.** From the closed-loop error trajectories (25) we can verify the following state transition model,

$$\boldsymbol{e}_{k+1} = \boldsymbol{A}_c^{k+1}\boldsymbol{e}_0 - \sum_{i=0}^{k} \boldsymbol{A}_c^{k-i}\boldsymbol{\Delta A}_c \boldsymbol{y}_i.$$

Since the eigenvalues of $\boldsymbol{A}_c$ lie within the unit disk, then there exist constants $\xi > 0$ and $\sigma \in [0, 1)$ such that $\|\boldsymbol{A}_c^k\| \leq \xi\sigma^k$. Therefore, from Definition 1 we have that

$$\beta(\|\boldsymbol{e}_0\|, k) = \xi\sigma^k\|\boldsymbol{e}_0\|,$$
$$\gamma(\|\boldsymbol{y}_k\|) = \sum_{j=0}^{\infty} \xi\sigma^j\|\boldsymbol{\Delta A}_c\|\|\boldsymbol{y}_k\| = \frac{\xi\|\boldsymbol{\Delta A}_c\|\|\boldsymbol{y}_k\|}{1-\sigma}.$$

To prove Definition 2, consider the following Lyapunov function

$$L_k = \boldsymbol{e}_k^\top \boldsymbol{M} \boldsymbol{e}_k. \tag{27}$$

By means of the Rayleigh inequality we can verify the first inequality of Definition 2 by defining $\alpha_1(\|\boldsymbol{e}\|) = \lambda_{\min}(\boldsymbol{M})\|\boldsymbol{e}\|^2$ and $\alpha_2(\|\boldsymbol{e}\|) = \lambda_{\max}(\boldsymbol{M})\|\boldsymbol{e}\|^2$. The second inequality of Definition 2 can be obtained from the direct computation of $\Delta L_k = L_{k+1} - L_k$ under the error dynamics (25) and Lyapunov function (26) as

$$\begin{aligned}
\Delta L_k =& \boldsymbol{e}_{k+1}^\top \boldsymbol{M} \boldsymbol{e}_{k+1} - \boldsymbol{e}_k^\top \boldsymbol{M} \boldsymbol{e}_k \\
=& \boldsymbol{e}_k^\top \boldsymbol{A}_c^\top \boldsymbol{M} \boldsymbol{A}_c \boldsymbol{e}_k - 2\boldsymbol{e}_k^\top \boldsymbol{A}_c^\top \boldsymbol{M} \boldsymbol{\Delta A}_c \boldsymbol{y}_k + \boldsymbol{y}_k^\top \boldsymbol{\Delta A}_c^\top \boldsymbol{M} \boldsymbol{\Delta A}_c \boldsymbol{y}_k - \boldsymbol{e}_k^\top \boldsymbol{M} \boldsymbol{e}_k \\
=& -\boldsymbol{e}_k^\top \boldsymbol{N} \boldsymbol{e}_k - 2\boldsymbol{e}_k^\top \boldsymbol{A}_c^\top \boldsymbol{M} \boldsymbol{\Delta A}_c \boldsymbol{y}_k + \boldsymbol{y}_k^\top \boldsymbol{\Delta A}_c^\top \boldsymbol{M} \boldsymbol{\Delta A}_c \boldsymbol{y}_k \\
\leq& -\frac{1}{2}\lambda_{\min}(\boldsymbol{N})\|\boldsymbol{e}_k\|^2 + \left(\frac{2\|\boldsymbol{A}_c^\top \boldsymbol{M} \boldsymbol{\Delta A}_c\|^2}{\lambda_{\min}(\boldsymbol{N})} + \|\boldsymbol{\Delta A}_c^\top \boldsymbol{M} \boldsymbol{\Delta A}_c\|^2\right)\|\boldsymbol{y}_k\|^2.
\end{aligned}$$

From the above inequality it follows that

$$\alpha_3(r) = \frac{1}{2}\lambda_{\min}(\boldsymbol{N})r^2,$$

$$\alpha_4(r) = \left( \frac{2\|\boldsymbol{A}_c^\top \boldsymbol{M} \boldsymbol{\Delta A}_c\|^2}{\lambda_{\min}(\boldsymbol{N})} + \|\boldsymbol{\Delta A}_c^\top \boldsymbol{M} \boldsymbol{\Delta A}_c\|^2 \right) r^2.$$

Therefore, from Definition 2 we can conclude that $L_k$ in (27) is an ISS-Lyapunov function, such that we can write

$$\Delta L_k \leq -\alpha_5(L_k) + \alpha_4(\|\boldsymbol{y}_k\|),$$

where $\alpha_5 := \alpha_3 \circ \alpha_2^{-1}$. This allows to conclude that the estimation error dynamics (25) is ISS and the trajectories converge to a bounded set $S_\mu$ of radius $\mu = \sqrt{\alpha_3^{-1} \circ \rho^{-1} \circ \alpha_4(\|\boldsymbol{y}_k\|)}$ as $k \to \infty$ for any $\mathcal{K}_\infty$-function $\rho(\cdot)$, i.e., $\|\boldsymbol{e}_k\| \leq \mu$. This completes the proof.

**Remark 2.** The asymptotic stability assumption in Theorem 1 is required to analyse the error of approximation caused by the loss of information using the proposed state-transition inference algorithm. Here, unstable control policies are obtained when the rank of $\boldsymbol{\Phi}$ is less than $w$, i.e., $\text{rank}(\boldsymbol{\Phi}) < w$, which means that the collected data lacks of richness. Unstable closed-loop systems cannot be used to determine the approximation error due to its divergent nature.

## 4. Results

In this section we verify the effectiveness of the proposed approach under several autonomous systems. Each model is discretized [36] using a sample time of $t_s = 0.02$ seconds. This is consistent with current on-board control modules embedded in autonomous platforms. A Gaussian distributed probing noise $\eta \sim \mathcal{N}(0, 4)$ with mean zero and variance 4 is used to ensure parameter estimates convergence of the Q-learning algorithm. A stabilization task at the origin is considered for each autonomous platform. We use a threshold of $\varepsilon_A = 0.001$ for the proposed heuristic.

### 4.1. Car bicycle model

Consider the following discrete-time car bicycle model [37]

$$\boldsymbol{x}_{k+1} = \begin{bmatrix} 1 & 0 & t_s & 0 \\ 0 & 1 & 0 & t_s \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \boldsymbol{x}_k + \begin{bmatrix} 0 & 0.0002 \\ 0.1111 & 0 \\ 0 & 0.02 \\ 0.0851 & 0 \end{bmatrix} \boldsymbol{u}_k,$$

where $\boldsymbol{x} = [\xi, y, v_x, \psi]^\top \in \mathbb{R}^4$ denotes the state vector composed by the longitudinal and lateral displacements, the longitudinal velocity of the vehicle, and the inertia heading angle, respectively, and $\boldsymbol{u} = [\delta_f, a_x]^\top \in \mathbb{R}^2$ is the control input given by the steering angle and longitudinal acceleration, respectively. The following weight matrices are used for the reward function design: $\boldsymbol{Q} = 10\boldsymbol{I}_4$ and $\boldsymbol{R} = \boldsymbol{I}_2$. The kernel solution and control gain of the DARE (6) under the proposed reward weights are given by

$$\boldsymbol{K} = \begin{bmatrix} 0 & 2.425 & 0 & 2.1579 \\ 3.0371 & 0 & 3.9113 & 0 \end{bmatrix},$$

$$\boldsymbol{P} = \begin{bmatrix} 643.9193 & 0 & 158.1929 & 0 \\ 0 & 44.4920 & 0 & -9.6538 \\ 158.1929 & 0 & 207.1450 & 0 \\ 0 & -9.6538 & 0 & 46.0325 \end{bmatrix}.$$

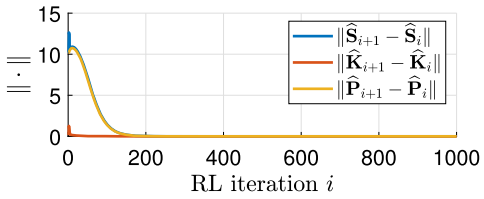
These matrices are used as ground-truth to verify the effectiveness of the Q-learning control algorithm and provide explanations. The XQL algorithm is trained using 5,000 iterations. Each RL iteration requires to collect at least 21 linear independent samples to apply the BLS Q-learning update rule. The learned-matrix $\widehat{\boldsymbol{S}}$ is

$$\widehat{\boldsymbol{S}}_{5000} = \begin{bmatrix} 653.9193 & 0 & 171.0713 & 0 & 0 & 3.2926 \\ 0 & 54.4920 & 0 & -0.7554 & 4.1236 & 0 \\ 171.0713 & 0 & 223.7303 & 0 & 0 & 4.2404 \\ 0 & -0.7554 & 0 & 53.9506 & 3.6694 & 0 \\ 0 & 4.1236 & 0 & 3.6694 & 1.7004 & 0 \\ 3.2926 & 0 & 4.2404 & 0 & 0 & 1.0841 \end{bmatrix}.$$

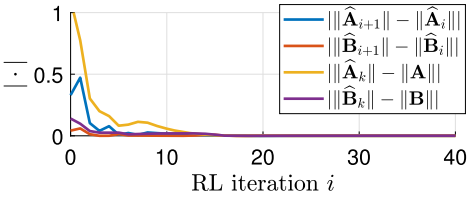
From this matrix we can obtain the control gain and kernel matrices using (14) and (13) which gives,

$$\widehat{\boldsymbol{K}}_{5000} = \begin{bmatrix} 0 & 2.425 & 0 & 2.1579 \\ 3.0371 & 0 & 3.9113 & 0 \end{bmatrix},$$

$$\widehat{\boldsymbol{P}}_{5000} = \begin{bmatrix} 643.9193 & 0 & 158.1929 & 0 \\ 0 & 44.492 & 0 & -9.6538 \\ 158.1929 & 0 & 207.145 & 0 \\ 0 & -9.6538 & 0 & 46.0325 \end{bmatrix}.$$




(a) RL convergence results




(b) State transition inference

**Fig. 3.** XQL results in the car bicycle model.

The results show that the Q-learning algorithm converges to the real values which gives an indicator that the algorithm effectively learns the state-transition dynamics. The matrices obtained from the proposed decomposition are

$$\boldsymbol{U} = \begin{bmatrix} 0 & 0 & 0.3083 & -0.9513 \\ -0.7347 & -0.6784 & 0 & 0 \\ 0 & 0 & -0.9513 & -0.3083 \\ -0.6784 & 0.7347 & 0 & 0 \end{bmatrix},$$

$$\boldsymbol{V} = \begin{bmatrix} 0 & 0 & -0.3297 & -0.9441 \\ 0.5756 & 0.8177 & 0 & 0 \\ 0 & 0 & 0.9441 & -0.3297 \\ 0.8177 & -0.5756 & 0 & 0 \end{bmatrix},$$

$$\boldsymbol{T} = \text{diag}\{-1.1047, -0.9052, -0.994, 1.0061\}.$$

Using (20) and (21) yield the following approximations for the matrices $\widehat{\boldsymbol{A}}$ and $\widehat{\boldsymbol{B}}$, respectively

$$\widehat{\boldsymbol{A}}_{5000} = \begin{bmatrix} 1.0046 & 0 & 0.0262 & 0 \\ 0 & 0.9693 & 0 & 0.3102 \\ -0.0188 & 0 & 0.9949 & 0 \\ 0 & -0.1124 & 0 & 0.9956 \end{bmatrix}, \quad \widehat{\boldsymbol{B}}_{5000} = \begin{bmatrix} 0 & 0.0003 \\ 0.1176 & 0 \\ 0 & 0.0199 \\ 0.0742 & 0 \end{bmatrix}.$$

Notice that the estimates $\widehat{\boldsymbol{A}}$ and $\widehat{\boldsymbol{B}}$ are not equivalent to $\boldsymbol{A}$ and $\boldsymbol{B}$ due to the use of the diagonal matrix $\boldsymbol{T}$. It is observed that the spectral norms of each matrix given by $\|\boldsymbol{A}\| = 0.105$, $\|\widehat{\boldsymbol{A}}\| = 0.1047$ which means that the estimated matrices distribute the information within their components with a minor loss of information. This distribution of values implies a minor change of eigenvalues. So, by applying the proposed heuristic in (23) gives $\mathcal{H} = 3 \times 10^{-4} < \varepsilon_A$, which implies that the Q-learning model infers accurately the state-transition matrices from the data.

Fig. 3 depicts the spectral error norm of the matrix estimates: Q-function $\widehat{\boldsymbol{S}}$, control gain $\widehat{\boldsymbol{K}}$, kernel matrix $\widehat{\boldsymbol{P}}$, and state-transition dynamics $\widehat{\boldsymbol{A}}$ and $\widehat{\boldsymbol{B}}$ of the proposed XQL algorithm. Here, fast convergence results are observed of both the Q-learning and the state-transition function. Here, the values of the spectral error norm are reduced as the number of iterations $i$ increases.

For comparison purposes, we used the enhanced Q-learning approach proposed in [38]. This approach uses a DMDc in parallel with the Q-learning algorithm to infer the state-transition matrices from the data. The basic scheme is shown Fig. 4.

In this experiment, the Q-learning element of the enhanced Q-learning algorithm behaves exactly the same as the proposed XQL. However, the main difference appears in the inference of the state-transition matrices. As previously discussed, system identification techniques such as DMD are usually used as independent algorithms such that the outputs of the Q-learning does not affect the system identification algorithm (unless unstable control policies are obtained). In this case, the DMDc needs to collect 24 linear independent samples to estimate the state-transition matrices, that is, 3 more samples compared with the proposed XQL. The number of samples required in this approach is $n^2 + nm \geq \frac{1}{2}(n+m)(n+m+1)$ which is computationally more expensive than the proposed approach. Fig. 5 shows the inference results using the DMDc in the enhanced Q-learning algorithm.

The plot shows that the DMDc algorithm finds the exact state-transition matrices in the first iteration. This is not what we expect to observe, since the Q-learning requires more iterations to converge to the optimal solution. This result is obtained since the Q-learning and DMDc have different optimisation objective. On the other hand, the proposed approach uses the outputs of the Q-learning to

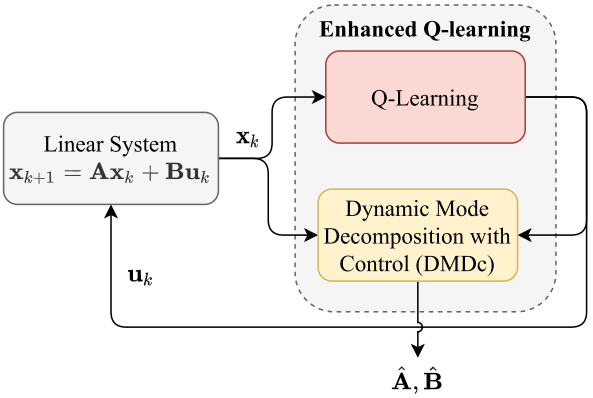



**Fig. 4.** Enhanced Q-learning approach.

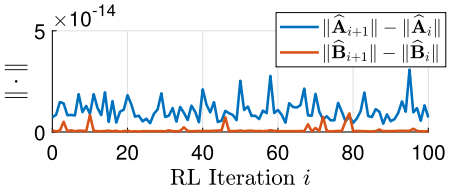



**Fig. 5.** State-transition matrices inference using Enhanced Q-learning proposed in [38].

compute the state-transition matrices learned by the Q-learning algorithm. One advantage of the DMDc is that it can be used for any linear autonomous system and it does not require a specific structure for matrix $\boldsymbol{A}$.

### 4.2. Quadcopter model

Consider the following discrete-time linear drone model obtained from the hover condition

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 0 & t_s & 0 & 0 \\ 0 & 1 & 0 & 0 & t_s & 0 \\ 0 & 0 & 1 & 0 & 0 & t_s \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} 0 & -\frac{t_s}{10} & 0 \\ \frac{t_s}{10} & 0 & 0 \\ 0 & 0 & 0.0004 \\ 0 & -gt_s & 0 \\ gt_s & 0 & 0 \\ 0 & 0 & 0.0411 \end{bmatrix}.$$

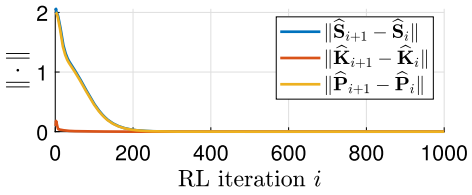
Here, the state is given by $\boldsymbol{x} = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^\top \in \mathbb{R}^6$ which is composed by the linear positions and velocities in each axis, $\boldsymbol{u} = [\phi, \theta, \mu]^\top \in \mathbb{R}^3$ is the control input composed by the roll and pitch Euler angles, and the total thrust, and $g$ is the gravitational acceleration. The following weight matrices are used for the reward function design: $\boldsymbol{Q} = \text{diag}\{1, 1, 1, 2, 2, 2\}$ and $\boldsymbol{R} = 2\boldsymbol{I}_3$. The kernel solution and control gain of the DARE (6) are given by

$$\boldsymbol{K} \approx \begin{bmatrix} 0 & 0.637 & 0 & 0 & 0.970 & \\ -0.637 & 0 & 0 & -0.97 & 0 & 0 \\ 0 & 0 & 0.688 & 0 & 0 & 1.272 \end{bmatrix},$$

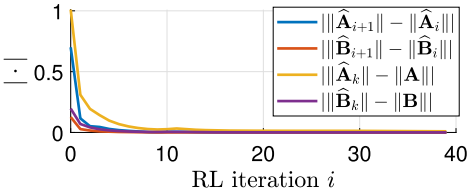
$$\boldsymbol{P} \approx \begin{bmatrix} 76.16 & 0 & 0 & 7.243 & 0 & 0 \\ 0 & 76.16 & 0 & 0 & 7.243 & 0 \\ 0 & 0 & 92.394 & 0 & 0 & 34.443 \\ 7.243 & 0 & 0 & 11.96 & 0 & 0 \\ 0 & 7.243 & 0 & 0 & 11.96 & 0 \\ 0 & 0 & 34.443 & 0 & 0 & 64.303 \end{bmatrix}.$$

The XQL algorithm is trained using 5,000 iterations. Here, each RL iteration needs 45 linear independent data samples to compute the BLS update rule. In this case, we do not display matrix $\hat{\boldsymbol{S}} \in \mathbb{R}^{9\times 9}$ due to its large dimension. However, we provide the control gain and kernel matrix that the Q-leaning algorithm converges after 5,000 iterations,

$$\hat{\boldsymbol{K}}_{5000} \approx \begin{bmatrix} 0 & 0.637 & 0 & 0 & 0.97 & 0 \\ -0.637 & 0 & 0 & -0.97 & 0 & 0 \\ 0 & 0 & 0.688 & 0 & 0 & 1.272 \end{bmatrix},$$




(a) RL convergence results




(b) State transition inference

**Fig. 6.** XQL results in the Quadcopter model.

$$\widehat{\boldsymbol{P}}_{5000} \approx \begin{bmatrix} 76.16 & 0 & 0 & 7.243 & 0 & 0 \\ 0 & 76.16 & 0 & 0 & 7.243 & 0 \\ 0 & 0 & 92.394 & 0 & 0 & 34.443 \\ 7.243 & 0 & 0 & 11.96 & 0 & 0 \\ 0 & 7.243 & 0 & 0 & 11.96 & 0 \\ 0 & 0 & 34.443 & 0 & 0 & 64.303 \end{bmatrix}.$$

Similar results are obtained for the quadcopter model. Both the control gain and the kernel matrix converge to their respective real values with minor numerical errors. However, it is observed that these numerical errors cause slightly changes in the inferred weight matrices $\widehat{\boldsymbol{A}}$ and $\widehat{\boldsymbol{B}}$ in each iteration. Since the proposed inference model depends on the eigendecomposition of specific matrices, then for small numeric changes in the estimated Q-function $\widehat{Q}(\boldsymbol{x}_k, \boldsymbol{u}_k)$ produce different eigenvectors. This gives as outcome slightly different state transition matrices $\widehat{\boldsymbol{A}}$ and $\widehat{\boldsymbol{B}}$ in each iteration. Here, these slightly changes are observed in how the values are distributed across all the elements of the inferred matrices. For example, the state transition matrix $\widehat{\boldsymbol{A}}$ in iteration 4,999 and 5,000 are the following

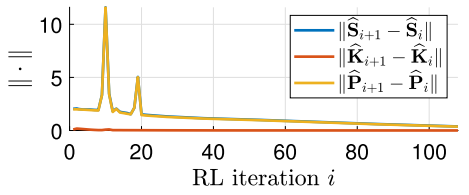
$$\widehat{\boldsymbol{A}}_1 \approx \begin{bmatrix} 1.002 & -0.004 & 0 & 0.024 & 0 & 0 \\ 0.004 & 1.002 & 0 & 0 & 0.024 & 0 \\ 0 & 0 & 1.003 & 0 & 0 & 0.026 \\ -0.02 & 0 & 0 & 0.997 & 0.001 & 0 \\ 0 & -0.02 & 0 & 0 & 0.997 & 0 \\ 0 & 0 & -0.009 & 0 & 0 & 0.996 \end{bmatrix}$$

$$\widehat{\boldsymbol{A}}_2 \approx \begin{bmatrix} 1.002 & -0.003 & 0 & 0.024 & 0 & 0 \\ 0.003 & 1.002 & 0 & 0 & 0.024 & 0 \\ 0 & 0 & 1.003 & 0 & 0 & 0.026 \\ -0.02 & 0 & 0 & 0.997 & -0.004 & 0 \\ 0 & -0.02 & 0 & 0 & 0.997 & 0 \\ 0 & 0 & -0.009 & 0 & 0 & 0.996 \end{bmatrix}$$

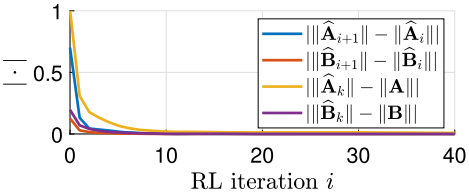
where $\widehat{\boldsymbol{A}}_1$ and $\widehat{\boldsymbol{A}}_2$ are the matrices obtained in iterations 4,999 and 5,000, respectively. Notice that slightly different matrices are inferred in each iteration. However, the spectral norm of both matrices remains the same, i.e., $\|\widehat{\boldsymbol{A}}_1\| = \|\widehat{\boldsymbol{A}}_2\| = 1.0094$, whilst the spectral norm of the real matrix is $\|\boldsymbol{A}\| = 1.01$. Here, each matrix has different but close eigenvalues and eigenvectors.

Applying the heuristic (23) gives $\mathcal{H} = 6 \times 10^{-4} < \varepsilon_A$ concluding that the Q-learning is learning the state-transition matrices properly. The numerical issue observed in the eigendecomposition is topic for further work in order to ensure stable convergence of the proposed state-transition decomposition model. Fig. 6 shows the evolution of the spectral error norm. Similar results to the car bicycle model are observed for this particular model, that is, asymptotic convergence and inferred state transition function close to the real values.

The experiments conducted so far demonstrate that the Q-learning learns well the optimal control problem for diverse systems using the proposed noise and number of iterations. To show how the proposed approach introduces a new explainable dimension to detect learning issues we reduce the amplitude of the probing noise into $\eta \sim 0.09\mathcal{N}(0,4)$ and maintain the same number of iterations.




(a) RL convergence results




(b) State transition inference

**Fig. 7.** XQL results of the quadcopter model under different probing noise.

In this setting, the learned control gain and kernel matrices converge to the following values after 5,000 iterations,

$$\widehat{\boldsymbol{K}}_{5000} \approx \begin{bmatrix} -0.011 & 0.578 & -0.002 & -0.001 & 0.964 & -0.001 \\ -0.583 & 0.011 & -0.01 & -0.964 & 0.001 & -0.005 \\ 0.012 & -0.003 & 0.58 & 0.001 & 0 & 1.217 \end{bmatrix},$$

$$\widehat{\boldsymbol{P}}_{5000} \approx \begin{bmatrix} 70.22 & -1.166 & 1.129 & 6.632 & -0.12 & 0.581 \\ -1.166 & 69.7311 & -0.269 & -0.12 & 6.582 & -0.143 \\ 1.13 & -0.269 & 81.751 & 0.116 & -0.028 & 29.061 \\ 6.632 & -0.12 & 0.116 & 11.897 & -0.012 & 0.06 \\ -0.12 & 6.582 & -0.028 & -0.012 & 11.892 & -0.015 \\ 0.581 & -0.143 & 29.061 & 0.06 & -0.015 & 61.562 \end{bmatrix}.$$

Notice that with this slightly change causes that the Q-learning algorithm is not capable to find the optimal solution using the same number of iterations as in the previous case. This means that more iterations are required or that $\boldsymbol{\Phi}$ is not rich enough. Additionally, the previous results mean that the Q-learning algorithm is not effectively learning the state-transition dynamics. The inferred state-transition matrices are

$$\widehat{\boldsymbol{A}}_{5000} \approx \begin{bmatrix} 1.001 & 0 & 0.001 & 0.024 & 0 & 0 \\ 0 & 1.001 & 0 & 0 & 0.024 & 0 \\ 0 & 0 & 1.002 & 0 & 0 & 0.026 \\ -0.023 & 0 & 0 & 0.997 & 0 & 0 \\ 0 & -0.023 & 0 & 0 & 0.997 & 0 \\ 0 & 0 & -0.01 & 0 & 0 & 0.996 \end{bmatrix}$$

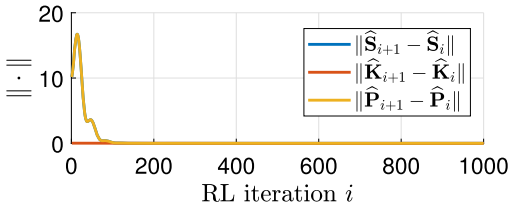
$$\widehat{\boldsymbol{B}}_{5000} \approx \begin{bmatrix} 0 & -0.0027 & 0 \\ 0.0027 & 0 & 0 \\ 0 & 0 & 0.0007 \\ 0 & -0.1957 & 0 \\ 0.19570 & 0 & \\ 0 & 0 & 0.0409 \end{bmatrix}.$$

The spectral norm of the inferred matrix $\widehat{\boldsymbol{A}}$ is $\|\widehat{\boldsymbol{A}}_{5000}\| = 1.0078$. So, $\mathcal{H} = 0.0022 > \varepsilon_A$ which means that the proposed heuristic is not satisfied and therefore, the Q-learning algorithm has not learned the adequate state-transition model. This result is coherent to what we expect to obtain since there exists a loss of information derived by the wrong state-transition inference. This is also observed in the XQL results of Fig. 7.

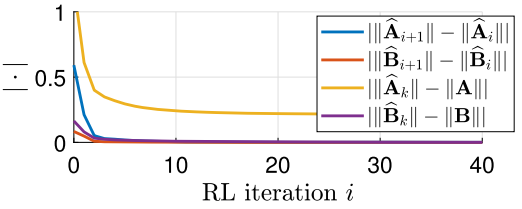
So, the proposed XQL provides a new explainable dimension to the classic Q-learning control algorithm that permits to study their learning outcomes based on the elements inferred from the data.

### *4.3. Challenges*

One of the main challenges of the proposed approach is caused by the assumption of a diagonal matrix $\boldsymbol{T}$. This constraints the class of autonomous systems that the approach can deal with. For example, consider the discrete-time model of a 1 degree-of-freedom (DOF) planar robot,




(a) RL convergence results




(b) State transition inference

**Fig. 8.** XQL results in a 1-DOF planar robot.

$$\boldsymbol{x}_{k+1} = \begin{bmatrix} 0.9961 & 0.02 \\ -0.3919 & 0.9961 \end{bmatrix} \boldsymbol{x}_k + \begin{bmatrix} 0.0016 \\ 0.1641 \end{bmatrix} u_k.$$

Notice that this model violates Assumption 1, that is, $\boldsymbol{A}$ should be an upper triangular matrix. Here, we aim to design an optimal control law using the following weight matrices for the reward function $\boldsymbol{Q} = \text{diag}\{10, 1\}$ and $R = 10$. The control gain and the kernel matrix of the DARE (6) using the planar robot model are

$$\boldsymbol{K} = \begin{bmatrix} 0.1201 & 0.3755 \end{bmatrix}, \quad \boldsymbol{P} = \begin{bmatrix} 504.8505 & 12.2732 \\ 12.2732 & 24.0084 \end{bmatrix}.$$

The same probing noise is used to ensure parameter estimates convergence in the XQL. The estimated control gain and kernel matrix are

$$\widehat{\boldsymbol{K}}_{5000} = \begin{bmatrix} 0.1201 & 0.3755 \end{bmatrix}, \widehat{\boldsymbol{P}}_{5000} = \begin{bmatrix} 504.8505 & 12.2732 \\ 12.2732 & 24.0084 \end{bmatrix}.$$

The Q-learning algorithm can effectively learn the optimal control problem. However, the state transition inference part cannot obtain good estimations since the diagonal matrix $\boldsymbol{T}$ causes a large loss of information. Here, the estimated state-transition dynamics is given as follows

$$\widehat{\boldsymbol{A}}_{5000} = \begin{bmatrix} 0.9903 & 0.0011 \\ -0.0021 & 1.0098 \end{bmatrix}, \quad \widehat{\boldsymbol{B}}_{5000} = \begin{bmatrix} -0.0015 \\ 0.1657 \end{bmatrix}.$$

In this case, the spectral norm of $\|\boldsymbol{A}\| = 1.2031$ and $\|\widehat{\boldsymbol{A}}_{5000}\| = 1.0099$, this means that the inferred matrix losses more information with $\mathcal{H} = 0.1932 > \varepsilon_A$. This is more acute on relatively large scale systems, where the loss of information is notably increased. Fig. 8 shows the evolution of the spectral error norm. Here, the Q-learning part can effectively learn the optimal control solution in an infinite horizon. On the other hand, the inferred state transition matrices have large error which can lead to biased conclusions of the quality of learning of the Q-learning algorithm. The state-transition inference for general linear matrices is topic for further work.

## 5. Conclusions

This paper reports an explainable Q-learning (XQL) algorithm for low-level reinforcement learning policies of autonomous systems. The approach aims to explain what does the reinforcement learning agent is learning in order to explain why a particular control policy is obtained. To this end, this paper hypothesizes that the obtained control policy is subject in how good the RL agent is capable to infer the state-transition dynamics. Hence, this paper provides a natural algorithm to infer what state-transition dynamics is learned in each iteration of the standard Q-learning algorithm. The proposed algorithm uses a matrix decomposition approach with a simple heuristic to approximately infer the real state-transition matrices of an autonomous platform. Simulation studies in different autonomous platforms verify the approach and provide some of the challenges and future research vectors in this field.

Future work will address the design of explainable tools for high-level decision making strategies, which requires the design of novel tools and theory to incorporate the diverse factors involved in reinforcement learning decisions.

## CRediT authorship contribution statement

**Adolfo Perrusquía:** Writing – original draft, Validation, Project administration, Methodology, Investigation, Conceptualization. **Mengbang Zou:** Validation, Methodology, Investigation, Formal analysis, Conceptualization. **Weisi Guo:** Writing – original draft, Supervision, Project administration, Funding acquisition, Conceptualization.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

No data was used for the research described in the article.

## References

[1] P. Wendt, A. Voltes-Dorta, P. Suau-Sanchez, Estimating the costs for the airport operator and airlines of a drone-related shutdown: an application to Frankfurt international airport, J. Transp. Secur. 13 (1) (2020) 93–116.
[2] C. Gruffeille, A. Perrusquía, A. Tsourdos, W. Guo, Disaster area coverage optimisation using reinforcement learning, in: 2024 International Conference on Unmanned Aircraft Systems (ICUAS), IEEE, 2024, pp. 61–67.
[3] Y. Bathaee, The artificial intelligence black box and the failure of intent and causation, Harv. J. Law Technol. 31 (2017) 889.
[4] H. Yau, C. Russell, S. Hadfield, What did you think would happen? Explaining agent behaviour through intended outcomes, Adv. Neural Inf. Process. Syst. 33 (2020) 18375–18386.
[5] X. Kong, Y. Xing, A. Tsourdos, Z. Wang, W. Guo, A. Perrusquia, A. Wikander, Explainable interface for human-autonomy teaming: a survey, preprint, arXiv: 2405.02583, 2024.
[6] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, K.-R. Müller, Evaluating the visualization of what a deep neural network has learned, IEEE Trans. Neural Netw. Learn. Syst. 28 (11) (2016) 2660–2673.
[7] R.R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-cam: visual explanations from deep networks via gradient-based localization, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 618–626.
[8] S.M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, Adv. Neural Inf. Process. Syst. 30 (2017).
[9] M.T. Ribeiro, S. Singh, C. Guestrin, "Why should I trust you?" explaining the predictions of any classifier, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1135–1144.
[10] A. Shrikumar, P. Greenside, A. Kundaje, Learning important features through propagating activation differences, in: International Conference on Machine Learning, PMLR, 2017, pp. 3145–3153.
[11] M. Sundararajan, A. Taly, Q. Yan, Axiomatic attribution for deep networks, in: International Conference on Machine Learning, PMLR, 2017, pp. 3319–3328.
[12] R.-E. Precup, R.-C. Roman, A. Safaei, Data-Driven Model-Free Controllers, CRC Press, 2021.
[13] T. Huber, M. Demmler, S. Mertes, M. Olson, GANterfactual-RL: understanding reinforcement learning agents' strategies through visual counterfactual explanations, preprint, arXiv:2302.12689, 2023.
[14] J. Ramírez, W. Yu, A. Perrusquía, Model-free reinforcement learning from expert demonstrations: a survey, Artif. Intell. Rev. 55 (2022) 3213–3241.
[15] J. Sun, L. Yu, P. Dong, B. Lu, B. Zhou, Adversarial inverse reinforcement learning with self-attention dynamics model, IEEE Robot. Autom. Lett. 6 (2) (2021) 1880–1886.
[16] A. Perrusquia, W. Guo, Drone's objective inference using policy error inverse reinforcement learning, IEEE Trans. Neural Netw. Learn. Syst. (2023).
[17] H. Tang, Towards Informed Exploration for Deep Reinforcement Learning, University of California, Berkeley, 2019.
[18] E. Bildik, A. Tsourdos, A. Perrusquía, G. Inalhan, Swarm decoys deployment for missile deceive using multi-agent reinforcement learning, in: 2024 International Conference on Unmanned Aircraft Systems (ICUAS), IEEE, 2024, pp. 256–263.
[19] W. Yu, A. Perrusquia, Human-Robot Interaction Control Using Reinforcement Learning, John Wiley & Sons, 2021.
[20] A. Vahidi-Moghaddam, M. Mazouchi, H. Modares, Memory-augmented system identification with finite-time convergence, IEEE Control Syst. Lett. 5 (2) (2020) 571–576.
[21] A. Perrusquía, W. Guo, Closed-loop output error approaches for drone's physics informed trajectory inference, IEEE Trans. Autom. Control (2023).
[22] P.J. Baddoo, B. Herrmann, B.J. McKeon, J. Nathan Kutz, S.L. Brunton, Physics-informed dynamic mode decomposition, Proc. R. Soc. A 479 (2271) (2023) 20220576.
[23] S. Bhasin, R. Kamalapurkar, M. Johnson, K.G. Vamvoudakis, F.L. Lewis, W.E. Dixon, A novel actor–critic–identifier architecture for approximate optimal control of uncertain nonlinear systems, Automatica 49 (1) (2013) 82–92.
[24] A. Perrusquía, W. Yu, Identification and optimal control of nonlinear systems using recurrent neural networks and reinforcement learning: an overview, Neurocomputing 438 (2021) 145–154.
[25] A. Perrusquía, W. Guo, Physics informed trajectory inference of a class of nonlinear systems using a closed-loop output error technique, IEEE Trans. Syst. Man Cybern. Syst. (2023).
[26] B. Kiumarsi, K.G. Vamvoudakis, H. Modares, F.L. Lewis, Optimal and autonomous control using reinforcement learning: a survey, IEEE Trans. Neural Netw. Learn. Syst. 29 (6) (2017) 2042–2062.
[27] F.L. Lewis, D. Vrabie, Reinforcement learning and adaptive dynamic programming for feedback control, IEEE Circuits Syst. Mag. 9 (3) (2009) 32–50.
[28] A. Perrusquía, Solution of the linear quadratic regulator problem of black box linear systems using reinforcement learning, Inf. Sci. 595 (2022) 364–377.
[29] K.G. Vamvoudakis, Q-learning for continuous-time linear systems: a model-free infinite horizon optimal control approach, Syst. Control Lett. 100 (2017) 14–20.
[30] I.A. Zamfirache, R.-E. Precup, E.M. Petriu, Q-learning, policy iteration and actor-critic reinforcement learning combined with metaheuristic algorithms in servo system control, Facta Univ., Mech. Eng. 21 (4) (2023) 615–630.
[31] B. Kiumarsi, F.L. Lewis, H. Modares, A. Karimpour, M.-B. Naghibi-Sistani, Reinforcement q-learning for optimal tracking control of linear discrete-time systems with unknown dynamics, Automatica 50 (4) (2014) 1167–1175.
[32] W. Xue, B. Lian, J. Fan, P. Kolaric, T. Chai, F.L. Lewis, Inverse reinforcement Q-learning through expert imitation for discrete-time systems, IEEE Trans. Neural Netw. Learn. Syst. 34 (5) (2023).
[33] A. Perrusquía, R. Garrido, W. Yu, Stable robot manipulator parameter identification: a closed-loop input error approach, Automatica 141 (2022) 110294.
[34] M.P. Deisenroth, A.A. Faisal, C.S. Ong, Mathematics for Machine Learning, Cambridge University Press, 2020.

[35] A. Perrusquía, W. Yu, Discrete-time $h_2$ neural control using reinforcement learning, IEEE Trans. Neural Netw. Learn. Syst. 32 (11) (2020) 4879–4889.
[36] F.L. Lewis, D. Vrabie, V.L. Syrmos, Optimal Control, John Wiley & Sons, 2012.
[37] S. Taherian, K. Halder, S. Dixit, S. Fallah, Autonomous collision avoidance using mpc with lqr-based weight transformation, Sensors 21 (13) (2021) 4296.
[38] M. Zou, A. Perrusquía, W. Guo, Explaining data-driven control in autonomous systems: a reinforcement learning case study, in: 10th International Conference on Control, Decision and Information Technologies, 2024.



https://dspace.org/

# Explainable data-driven Q-learning control for a class of discrete-time linear autonomous systems

**Perrusquía, Adolfo; Zou, Mengbang; Guo, Weisi**

**2024-11-01**
Elsevier
https://doi.org/10.1016/j.ins.2024.121283
Perrusquía A, Zou M, Guo W. (2024) Explainable data-driven Q-learning control for a class of discrete-time linear autonomous systems. Information Sciences, Volume 682, November 2024, Article number 121283

Downloaded from a DSpace repository. https://dspace.org/