# On Periodic Reference Tracking Using Batch-Mode Reinforcement Learning with Application to Gene Regulatory Network Control

Aivar Sootla, Natalja Strelkowa, Damien Ernst, Mauricio Barahona, Guy-Bart Stan



***Abstract*—In this paper, we consider the periodic reference tracking problem in the framework of batch-mode reinforcement learning, which studies methods for solving optimal control problems from the sole knowledge of a set of trajectories. In particular, we extend an existing batch-mode reinforcement learning algorithm, known as Fitted Q Iteration, to the periodic reference tracking problem. The presented periodic reference tracking algorithm explicitly exploits a priori knowledge of the future values of the reference trajectory and its periodicity. We discuss the properties of our approach and illustrate it on the problem of reference tracking for a synthetic biology gene regulatory network known as the generalised repressilator. This system can produce decaying but long-lived oscillations, which makes it an interesting system for the tracking problem. In our companion paper we also take a look at the regulation problem of the toggle switch system, where the main goal is to drive the systems states to a specific bounded region in the state space.**

***Index Terms*—batch-mode reinforcement learning; reference tracking; fitted Q iteration; synthetic biology; gene regulatory networks; generalised repressilator**

## I. Introduction

There are many problems in engineering, which require forcing the system to follow a desired periodic reference trajectory (e.g., in repetitive control systems [1]). One approach to solve these problems is to define first a distance between the state of the system $\boldsymbol{n}_t$ and the desired reference point $\boldsymbol{r}_t$ at time $t$. Then explicitly seek for a control policy that minimises this distance for all $t$ subject to problem specific constraints. In the case of systems in the Euclidean state-space, for example, the Euclidean distance can be used.

In this paper, we are interested in solving reference tracking problems of discrete-time systems using an optimal control approach. In our setting, the dynamics of the system is only known in the form of one-step system transitions. A one-step system transition is a triplet $\{\boldsymbol{n}, \boldsymbol{u}, \boldsymbol{n}^+\}$, where $\boldsymbol{n}^+$ denotes a successor state of the system in state $\boldsymbol{n}$ subjected to input $\boldsymbol{u}$. Inference of near-optimal policies from one-step system transitions is the central question of batch-mode reinforcement learning [2], [3]. Note that reinforcement learning [4] focuses on a more general problem of policy inference from interactions with the system. Usually, in batch-mode reinforcement learning, very few assumptions are made on the structure of the control problem. This gives batch-mode reinforcement learning algorithms a high degree of flexibility in comparison with other control methods. Batch-mode reinforcement learning has had numerous applications in many disciplines such as engineering [5], HIV treatment design [6], and medicine [7]. In this paper, one batch-mode reinforcement learning algorithm is considered, namely the Fitted $Q$ Iteration algorithm [8]. This algorithm focuses on the computation of a so called $Q$ function, which is used to compute a near-optimal control policy. The algorithm computes the $Q$ function using an iterative procedure based on the Bellman equation, a regression method and the collected samples of system trajectory.

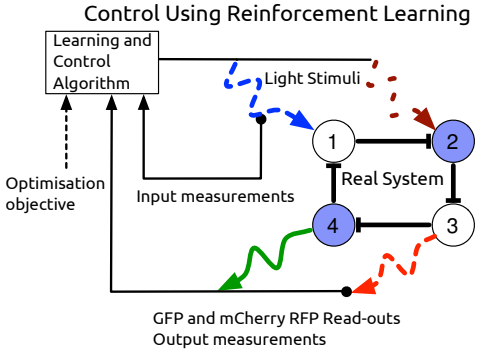


Fig. 1. A schematic depiction of an exogenously controlled generalised repressilator. Numbered circles represent different genes. An arrow with a flat end connecting two circles represents a repressive action from one gene product onto the other one. For example, gene 2 is repressed by the protein product of gene 1.

The focus of this paper is an extension of the Fitted $Q$ Iteration to the reference tracking problem. In our extension, we explicitly take into account the future reference trajectory values and the period $T$. Moreover, regression is separated into $T$ independent regression problems, which can save computational effort. The algorithm is implemented in Python using open-source packages and libraries [9], [10], [11], [12].

To illustrate our reference tracking method, we consider its application to a synthetic biology gene regulatory network known as the generalised repressilator. The repressilator is a ring of three mutually repressing genes pioneered in [13], and theoretically generalised to rings consisting of more than three

Aivar Sootla and Guy-Bart Stan (corresponding author) are with the Centre for Synthetic Biology and Innovation and the Department of Bioengineering, Mauricio Barahona is with the Department of Mathematics, Imperial College London, UK {a.sootla, g.stan, m.barahona}@imperial.ac.uk, Natalja Strelkowa is with Boehringer-Ingelheim Pharma GmbH & Co KG., Germany, natalja.strelkowa@boehringer-ingelheim.com, and Damien Ernst is with the Montefiore Institute, University of Liège, Belgium dernst@ulg.ac.be

genes in [14]. Repression is an interaction between two genes, such that the protein product of the repressing gene prevents protein expression of the repressed gene. Or simply put, where one gene can turn off the other. A generalised repressilator with a sufficiently large, even number of genes (such as the four-gene ring in Figure 1) can exhibit decaying but very long-lived oscillations [15]. The objective of this paper is to force one or several protein concentrations to follow *a priori* defined reference trajectories, namely: sinusoids and ramps. A deeper background on synthetic biology and its control applications is provided in our companion paper [16].

The paper is organised as follows. Section II covers batch-mode reinforcement learning. In Section III, an extension of the Fitted $Q$ Iteration to the reference tracking problem is presented. In Section IV our illustrative application, reference tracking for the generalised repressilator system, is given. Finally, Section V concludes the paper.

## II. Fitted $Q$ Iteration

### A. Problem Formulation

Consider a deterministic discrete-time dynamical system

$$\boldsymbol{n}_{t+1} = f(\boldsymbol{n}_t, \boldsymbol{u}_t) \tag{1}$$

where the bold values $\boldsymbol{n}$ stand for vectors with elements $n^i$, $\boldsymbol{n}_t$ is the state of the system at time $t$, and $\boldsymbol{u}_t$ is a control input, which at each time $t$ belongs to a compact set $U$. Consider also, associated with this dynamical system, an optimal control problem, which is defined in terms of the minimisation of an infinite sum of discounted costs $\boldsymbol{c}(\boldsymbol{n}, \boldsymbol{u})$:

$$\min_{\mu(\cdot)} \sum_{i=t}^{\infty} \gamma^{i-t} \boldsymbol{c}(\boldsymbol{n}_i, \mu(\boldsymbol{n}_i))$$

where $\mu(\cdot)$ is a mapping from the state-space onto $U$, which is called the feedback control policy, and $\gamma$ is a positive constant smaller than one, which is called the discount factor. For the purpose of this paper, it is assumed that $\boldsymbol{c}(\cdot,\cdot)$ is a given function. The goal is to compute the optimal policy based only on one-step transitions of the system (1). One-step system transitions are given as a set $\mathcal{F} = \{\boldsymbol{n}_l, \boldsymbol{u}_l, \boldsymbol{n}_l^+\}_{l=1}^{\#\mathcal{F}}$, where $\boldsymbol{n}_l^+$ denotes a successor state of the system in state $\boldsymbol{n}_l$ subjected to input $\boldsymbol{u}_l$ (if the function $f(\cdot,\cdot)$ is known then $\boldsymbol{n}_l^+$ is simply equal to $f(\boldsymbol{n}_l, \boldsymbol{u}_l)$).

### B. Algorithm

The central object in Fitted $Q$ Iteration is the $Q$ function, which is defined as follows:

$$Q(\boldsymbol{n}_t, \boldsymbol{u}_t) = \boldsymbol{c}(\boldsymbol{n}_t, \boldsymbol{u}_t) + \min_{\mu(\cdot)} \sum_{i=t+1}^{\infty} \gamma^{i-t} c(\boldsymbol{n}_i, \mu(\boldsymbol{n}_i)).$$

The main idea of the approach is to exploit the celebrated Bellman equation

$$Q(\boldsymbol{n}, \boldsymbol{u}) = \boldsymbol{c}(\boldsymbol{n}, \boldsymbol{u}) + \gamma \min_{\boldsymbol{u}' \in U} Q(f(\boldsymbol{n}, \boldsymbol{u}), \boldsymbol{u}'), \tag{2}$$

which provides a convenient expression for the computation of the optimal feedback control policy:

$$\mu^*(\boldsymbol{n}) = \operatorname*{argmin}_{\boldsymbol{u} \in U} Q(\boldsymbol{n}, \boldsymbol{u}) \tag{3}$$

**Algorithm 1** Fitted $Q$ iteration

**Inputs:** Set of one-step system transitions $\mathcal{F} = \{\boldsymbol{n}_l, \boldsymbol{u}_l, \boldsymbol{n}_l^+\}_{l=1}^{\#\mathcal{F}}$, cost $\boldsymbol{c}(\cdot,\cdot)$, stopping criterion
**Outputs:** Policy $\hat{\mu}^*(\boldsymbol{n})$

$k \leftarrow 0$
$\hat{Q}_0(\cdot,\cdot) \leftarrow \boldsymbol{c}(\cdot,\cdot)$
**repeat**
  $k \leftarrow k + 1$
  Compute (5) to obtain the values of $\hat{Q}_k(\cdot,\cdot)$ for all $\{\boldsymbol{n}_l, \boldsymbol{u}_l\}$ in $\mathcal{F}$
  Estimate the function $\hat{Q}_k(\boldsymbol{n}, \boldsymbol{u})$ using a regression algorithm with input pairs $(\boldsymbol{n}_l, \boldsymbol{u}_l)$ and function values $\hat{Q}_k(\boldsymbol{n}_l, \boldsymbol{u}_l)$.
**until** stopping criterion is satisfied
Compute the policy according to (6)

The Bellman equation can be theoretically solved using an iterative procedure

$$Q_k(\boldsymbol{n}, \boldsymbol{u}) = \boldsymbol{c}(\boldsymbol{n}, \boldsymbol{u}) + \gamma \min_{\boldsymbol{u}' \in U} Q_{k-1}(f(\boldsymbol{n}, \boldsymbol{u}), \boldsymbol{u}'),$$

where $Q_0 = \boldsymbol{c}$ and $Q_\infty$ is the unique solution to (2) due to the fact that $T(Q) = \boldsymbol{c} + \gamma \min_{\boldsymbol{u}' \in U} Q$ is a contraction mapping (cf. [3]). However in the continuous state-space case, this iterative procedure is hard to solve, especially when only the one-step system transitions in $\mathcal{F}$ are given. The Fitted Q iteration algorithm computes, from the sole knowledge of $\mathcal{F}$ a sequence of functions $\hat{Q}_1$, $\hat{Q}_2$, ... that approximates the sequence $Q_1$, $Q_2$, .... Let $\hat{Q}_0 = \boldsymbol{c}$ and for every $(\boldsymbol{n}_l, \boldsymbol{u}_l, \boldsymbol{n}_l^+)$ in $\mathcal{F}$ compute:

$$\hat{Q}_1(\boldsymbol{n}_l, \boldsymbol{u}_l) = \boldsymbol{c}(\boldsymbol{n}_l, \boldsymbol{u}_l) + \gamma \min_{\boldsymbol{u} \in U} \hat{Q}_0(\boldsymbol{n}_l^+, \boldsymbol{u}). \tag{4}$$

This expression gives $\hat{Q}_1$ only for $\boldsymbol{n}_l$, $\boldsymbol{u}_l$ in $\mathcal{F}$, while the entire function $\hat{Q}_1(\cdot,\cdot)$ is estimated using a regression algorithm (e.g., EXTRA Trees [17]). Now the iterative procedure is derived by generalising (4) as follows:

$$\hat{Q}_k(\boldsymbol{n}_l, \boldsymbol{u}_l) = \boldsymbol{c}(\boldsymbol{n}_l, \boldsymbol{u}_l) + \gamma \min_{\boldsymbol{u} \in U} \hat{Q}_{k-1}(\boldsymbol{n}_l^+, \boldsymbol{u}), \tag{5}$$

where at each step $\hat{Q}_k(\cdot,\cdot)$ is estimated using a regression algorithm. If the iteration procedure stops at the iteration number $k$, an approximate policy can be computed as follows:

$$\hat{\mu}^*(\boldsymbol{n}) = \operatorname*{argmin}_{\boldsymbol{u} \in U} \hat{Q}_k(\boldsymbol{n}, \boldsymbol{u}) \tag{6}$$

A simple stopping criterion can be, for example, an *a priori* fixed maximum number of iterations $N_{\text{it}}$. The value $N_{\text{it}}$ is chosen such that $\gamma^{N_{\text{it}}}$ is sufficiently small and, hence, the values $\hat{Q}_k(\boldsymbol{n}_l, \boldsymbol{u}_l)$ are not modified significantly for $k$ larger than $N_{\text{it}}$. Other, more advanced, stopping criteria are described in [8]. The resulting iterative method is outlined in Algorithm 1. A major property of the fitted $Q$ iteration algorithm is convergence. This is understood as convergence of $\hat{Q}_k$ to a fixed state-action value function $\hat{Q}^*$ given a fixed set $\mathcal{F}$ as $k$ grows to infinity. It was shown in [8], that under certain conditions Algorithm 1 converges and it is possible to estimate in advance the distance between $\hat{Q}^*$ and the iterate $\hat{Q}_k$.

## III. Periodic Reference Tracking Using the Fitted $Q$ Iteration Algorithm

### A. Problem Formulation

Consider, a system:

$$\begin{aligned} \boldsymbol{n}_{t+1} &= f(\boldsymbol{n}_t, \boldsymbol{u}_t) \\ \boldsymbol{r}_{t+1} &= g(\boldsymbol{r}_t) \end{aligned} \tag{7}$$

where the function $f(\cdot,\cdot)$ is unknown. The function $g(\cdot)$, however, is a known periodic function of period $T$, and the variable $\boldsymbol{r}$ takes only a finite number of values $\{\boldsymbol{v}_i\}_{i=1}^{T}$. Hence, $g(\boldsymbol{v}_i)$ is equal to $\boldsymbol{v}_{i+1}$ for all $i$ smaller than $T$, and $g(\boldsymbol{v}_T)$ is equal to $\boldsymbol{v}_1$. The reference tracking problem is defined as follows:

$$\begin{aligned} &\min_{\mu(\cdot,\cdot)} \sum_{i=t}^{\infty} \gamma^{i-t} \boldsymbol{c}(d(\boldsymbol{n}_i, \boldsymbol{r}_i), \boldsymbol{n}_i, \boldsymbol{u}_i) \\ &\text{subject to} \quad \text{system dynamics (7), and} \\ &\mu(\boldsymbol{n}_t, \boldsymbol{r}_t) = \boldsymbol{u}_t \in U \end{aligned} \tag{8}$$

where $\boldsymbol{c}$ is a given instantaneous cost function, $d(\cdot,\cdot)$ is a function defining the distance between the current state $\boldsymbol{n}_i$ and reference $\boldsymbol{r}_i$, and $\mu(\cdot,\cdot)$ is a feedback control policy. In order to track the reference $\boldsymbol{r}$, reducing the value of the cost $\boldsymbol{c}$ must reduce the distance $d$. The instantaneous cost can optionally depend on the control signal $\boldsymbol{u}$ and the states $\boldsymbol{n}$ in order to provide additional constraints in the state-space and/or control action space. As for the Fitted $Q$ Iteration, the control policy is inferred based solely on the trajectories given in the form of one-step system transitions $\mathcal{F} = \{\boldsymbol{n}_l, \boldsymbol{u}_l, \boldsymbol{n}_l^+\}_{l=1}^{\#\mathcal{F}}$, where $\boldsymbol{n}_l^+$ denotes a successor state of the system in state $\boldsymbol{n}_l$ subjected to input $\boldsymbol{u}_l$.

### B. Algorithm

The variable $\boldsymbol{r}$ can be seen as an additional state in the extended state-space $\{\boldsymbol{n}, \boldsymbol{r}\}$. Based on this extended state-space we can derive the Bellman equation for the tracking problem and the following iterative procedure for the computation of the $Q$ function:

$$\begin{aligned} Q_k(\boldsymbol{n}, \boldsymbol{r}, \boldsymbol{u}) = \boldsymbol{c}(d(\boldsymbol{n}, \boldsymbol{r}), \boldsymbol{n}, \boldsymbol{u}) + \\ \min_{\boldsymbol{u}' \in U} Q_{k-1}(\boldsymbol{n}^+, g(\boldsymbol{r}), \boldsymbol{u}') \quad \forall \boldsymbol{n}, \boldsymbol{r}, \boldsymbol{u} \end{aligned} \tag{9}$$

where $Q_0$ is equal to $\boldsymbol{c}$. It can be shown that this iterative procedure has a unique solution $Q^*$ based on a similar contraction mapping argument as in the previous section. Hence the optimal policy in the extended state-space is computed as:

$$\mu(\boldsymbol{n}, \boldsymbol{r}) = \min_{\boldsymbol{u} \in U} Q^*(\boldsymbol{n}, \boldsymbol{r}, \boldsymbol{u})$$

The input set to the algorithm normally consists of the one-step system transitions $\mathcal{F} = \{\boldsymbol{n}_l, \boldsymbol{u}_l, \boldsymbol{n}_l^+\}_{l=1}^{\#\mathcal{F}}$. However, since our state-space has been extended to include $\boldsymbol{r}$, $\mathcal{F}$ should now include $\boldsymbol{r}$ as well. Since the time evolution of $\boldsymbol{r}$ is known *a priori* we can simply modify the set as follows $\mathcal{TF} = \{\boldsymbol{n}_l, \boldsymbol{v}_i, \boldsymbol{u}_l, \boldsymbol{n}_l^+, g(\boldsymbol{v}_i)\}_{i,l}$, where $i = 1, \ldots, T$ and $l = 1, \ldots, \#\mathcal{F}$. This will effectively copy $T$ times the training set $\mathcal{F}$. Now given this modification, we can proceed to the computational procedure. As before $\hat{Q}_0$ is equal to $\boldsymbol{c}$ and the next iterates can be obtained according to:

$$\begin{aligned} \hat{Q}_k(\boldsymbol{n}_l, \boldsymbol{v}_i, \boldsymbol{u}_l) = \boldsymbol{c}(d(\boldsymbol{n}_l, \boldsymbol{v}_i), \boldsymbol{n}_l, \boldsymbol{u}_l) + \\ \min_{\boldsymbol{u}' \in U} \hat{Q}_{k-1}(\boldsymbol{n}_l^+, g(\boldsymbol{v}_i), \boldsymbol{u}') \quad \forall l,\, \boldsymbol{v}_i \end{aligned} \tag{10}$$

According to the Fitted Q Iteration framework, every function $\hat{Q}_k(\cdot,\cdot,\cdot)$ must be estimated by a regression algorithm, which uses the input set $\{\boldsymbol{n}_l, \boldsymbol{v}_i, \boldsymbol{u}_l\}_{i,l}$ and the corresponding values of the approximated function $\hat{Q}_k(\boldsymbol{n}_l, \boldsymbol{v}_i, \boldsymbol{u}_l)$. This input set can grow significantly, if the period $T$ of the reference trajectory is large, which can render a regression algorithm computationally intractable. For example, with only a thousand one-step system transitions $\{\boldsymbol{n}_l, \boldsymbol{u}_l, \boldsymbol{n}_l^+\}_{l=1}^{\#\mathcal{F}}$ and period $T$ equal to 200, the total number of samples $\{\boldsymbol{n}_l, \boldsymbol{v}_i, \boldsymbol{u}_l\}_{i,l}$ is equal to 200 000. Therefore, it is proposed to break up the regression of $\hat{Q}_k(\cdot,\cdot,\cdot)$ into $T$ independent regression problems, one for every function $\hat{Q}_k(\cdot, \boldsymbol{v}_i, \cdot)$. This can be done, because the evolution of the variable $\boldsymbol{r}$ is known in advance and $\boldsymbol{r}$ takes a finite number of values. To make these ideas more transparent, (10) is rewritten using a different notation as follows:

$$\begin{aligned} \hat{Q}_k^{\boldsymbol{v}_i}(\boldsymbol{n}_l, \boldsymbol{u}_l) = \boldsymbol{c}(d(\boldsymbol{n}_l, \boldsymbol{v}_i), \boldsymbol{n}_l, \boldsymbol{u}_l) + \\ \min_{\boldsymbol{u} \in U} \hat{Q}_{k-1}^{g(\boldsymbol{v}_i)}(\boldsymbol{n}_l^+, \boldsymbol{u}) \quad \forall l,\, \boldsymbol{v}_i, \end{aligned} \tag{11}$$

where $\hat{Q}_k^{\boldsymbol{v}_i}(\cdot,\cdot)$ stands for the function $\hat{Q}_k(\cdot, \boldsymbol{v}_i, \cdot)$. For every value $\boldsymbol{v}_i$, the regression algorithm will approximate the function $\hat{Q}_k^{\boldsymbol{v}_i}(\cdot,\cdot)$ by using the input set $\{\boldsymbol{n}_l, \boldsymbol{u}_l\}_l$ and the corresponding values $\hat{Q}_k^{\boldsymbol{v}_i}(\boldsymbol{n}_l, \boldsymbol{u}_l)$. Thus the initial regression problem is indeed separated into $T$ independent problems.

Our approach can be also seen as a modification of the $\hat{Q}_k$ function approximator. The first layer of our approximator is a deterministic branching according to the values $\boldsymbol{v}_i$. After that a regression algorithm is performed to approximate the functions $\hat{Q}_k^{\boldsymbol{v}_i}(\cdot,\cdot)$ as prescribed. Finally, if the iterative procedure has stopped at iteration $k$, a near-optimal policy can be computed as follows:

$$\hat{\mu}^*(\boldsymbol{n}, \boldsymbol{r}) = \min_{\boldsymbol{u} \in U} \hat{Q}_k^{\boldsymbol{r}}(\boldsymbol{n}, \boldsymbol{u}) \tag{12}$$

Our approach is outlined in Algorithm 2. Periodicity is a crucial assumption, since the period of the reference trajectory corresponds to the number of different $\hat{Q}_k^{\boldsymbol{v}}$ functions built in this algorithm. Convergence of Algorithm 2 can be established using the following lemma.

*Lemma 1:* Algorithm 2 converges under similar conditions and considerations as the fitted $Q$ iteration algorithm in [8]. Moreover, the stopping criteria from [8] can be directly applied to Algorithm 2.

To prove this lemma, we have to make sure that the approximator of the $\hat{Q}_k$ functions will not break the convergence proof in [8]. Only one modification in this approximator is made, which is the deterministic branching in its first layer according to $\boldsymbol{v}_i$. It can be shown that this modification does not violate the convergence arguments in [8].

Note, if we assume that the cost function is time-independent, i.e., $\boldsymbol{r}$ is constant and equal to $\boldsymbol{v}$, and $g(\boldsymbol{v})$ is

**Algorithm 2** Reference Tracking Fitted $Q$ Iteration

**Inputs:** Sets of one-step system transitions $\mathcal{F} = \{\boldsymbol{n}_l, \boldsymbol{u}_l, \boldsymbol{n}_l^+\}_{l=1}^{\#\mathcal{F}}$, function $g(\cdot)$ and reference values $\{\boldsymbol{v}_i\}_{i=1}^{T}$, cost $\boldsymbol{c}(d(\cdot,\cdot),\cdot,\cdot)$, stopping criterion

**Outputs:** Policy $\hat{\mu}^*(\boldsymbol{n}, \boldsymbol{r})$

$k \leftarrow 0$

$\hat{Q}_0(\cdot,\cdot) \leftarrow \boldsymbol{c}(d(\cdot,\cdot),\cdot,\cdot)$

**repeat**

$k \leftarrow k+1$

Compute (11) to obtain the values of $\hat{Q}_k^{\boldsymbol{v}_i}(\cdot,\cdot)$ for all $\{\boldsymbol{n}_l, \boldsymbol{u}_l\}$ in $\mathcal{F}$ and $\boldsymbol{v}_i$

Estimate the functions $\hat{Q}_k^{\boldsymbol{v}_i}(\boldsymbol{n}, \boldsymbol{u})$ for every $\boldsymbol{v}_i$ using a regression algorithm with input pairs $(\boldsymbol{n}_l, \boldsymbol{u}_l)$ and function values $\hat{Q}_k^{\boldsymbol{v}_i}(\boldsymbol{n}_l, \boldsymbol{u}_l)$.

**until** stopping criterion is satisfied

Compute the policy using (12)

equal to $\boldsymbol{v}$, equation (11) becomes:

$$\hat{Q}_k^{\boldsymbol{v}}(\boldsymbol{n}_l, \boldsymbol{u}_l) = \boldsymbol{c}(d(\boldsymbol{n}_l, \boldsymbol{v}), \boldsymbol{n}_l, \boldsymbol{u}_l) + \min_{\boldsymbol{u}' \in U} \hat{Q}_{k-1}^{\boldsymbol{v}}(\boldsymbol{n}_l^+, \boldsymbol{u}') \quad \forall l$$

which reduces Algorithm 2 to Algorithm 1 without any artefacts. Algorithm 1 is applied to control of a gene regulatory network in our companion paper [16].

### C. Other Applications of the Algorithm

The presented algorithm addresses the problem of optimal control of the system (7), where a part of the dynamics is known and a part of the dynamics is to be learned. In our setting, the known dynamics describe a reference trajectory, which takes only a finite number of values. However, there are other problems for which Algorithm 2 can be useful. Consider the system, where $f(\cdot,\cdot)$ describes the system dynamics and $g(\cdot,\cdot)$ is known:

$$\begin{aligned} \boldsymbol{n}^+ &= f(\boldsymbol{n}, \boldsymbol{r}) \\ \boldsymbol{r}^+ &= g(\boldsymbol{r}, \boldsymbol{u}) \end{aligned} \tag{13}$$

The function $g(\cdot,\cdot)$ could be a model of an actuator or some dynamics of the system, e.g., a time-delay in the control signal. In the latter case, the function $g(\cdot,\cdot)$ delays the application of the control action by a number of time samples. If $\boldsymbol{u}$ takes a finite number of values, then the problem can be solved using Algorithm 2 without major modifications.

Another application is a system with dynamics $f(\cdot,\cdot)$ and a given feedforward compensator $g(\cdot,\cdot)$, which helps to counteract the measured disturbance $\boldsymbol{w}$:

$$\begin{aligned} \boldsymbol{n}^+ &= f(\boldsymbol{n}, \boldsymbol{u} + \boldsymbol{r}) \\ \boldsymbol{r}^+ &= g(\boldsymbol{r}, \boldsymbol{w}) \end{aligned} \tag{14}$$

## IV. Tracking of Periodic Trajectories by the Generalized Repressilator

### A. System Description

As an illustrative application of our method we consider the problem of periodic reference tracking for a six gene

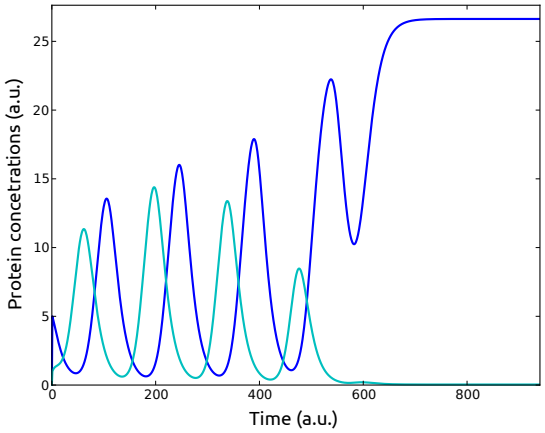


Fig. 2. Natural oscillatory behaviour of a generalised repressilator system consisting of 6 genes. The blue line represents the time evolution of the protein concentration produced by the expression of gene 1; the cyan line represents the time evolution of the protein concentration produced by the expression of gene 2. The oscillations have a period of approximately 150 arbitrary time units but are, however, not stable.

generalised repressilator system. There are two major species associated with every gene (the mRNA and protein concentrations), which results in a twelve state system. Throughout the section we adopt the following notation: $p^i$ denotes the protein concentration produced by the translation of mRNA $m^i$ of gene $i$. By definition of the generalised repressilator, the transcription of mRNA $m^i$ is repressed by the previous gene expression product $p^{i-1}$ in the network. With a slight abuse of notation, we assume that $p^{-1}$ is equal to $p^n$ in order to model the cyclic structure of the generalised repressilator. The dynamics of the generalised repressilator system can be described by the following set of deterministic equations:

$$\begin{aligned} \dot{m^i} &= \frac{c_1^i}{1 + (p^{i-1})^2} - c_2^i m^i + \delta_{i1} b_1 u^1 + \delta_{i2} b_2 u^2 \\ \dot{p^i} &= c_3^i m^i - c_4^i p^i \end{aligned} \tag{15}$$

where $i$ is an integer from one to six, and $\delta_{ij}$ is equal to one, if $i$ is equal to $j$, and equal to zero otherwise. We consider a control scheme similar to the one used for the toggle switch regulation problem (see our companion paper [16]), where the light control signals induce the expression of genes through the activation of their photo-sensitive promoters. The control signal $u^1$ only acts on the mRNA dynamics of gene 1, whereas the control signal $u^2$ only acts on the mRNA dynamics of gene 2. The influence of the light signals on the rate of mRNA production of genes 1 and 2 is denoted by $b_1$ and $b_2$, respectively. To simplify the system dynamics and as it is usually done for the repressilator model [13], we consider the corresponding parameters of the mRNA and protein dynamics for different genes to be equal. Hence, the trajectories will be very similar between the different genes. For the purpose of this paper, we chose:

$$\forall i: c_1^i = 1.6, \quad c_2^i = 0.16, \quad c_3^i = 0.16, \quad c_4^i = 0.06, \quad b_1 = b_2 = 5$$

As shown in [15], this system exhibits a long-lived oscillatory behaviour around an unstable limit cycle as depicted

in Figure 2. The "natural" period of these slowly decaying oscillations is around 150 arbitrary time units.

### B. Algorithm Parameters and Implementation

The instantaneous cost function is defined differently based on the considered example. In the first case, the objective is for the concentration of protein $p^2$ to track an *a priori* specified reference trajectory. In the other cases, the objective is for the concentrations of two proteins $p^1$ and $p^2$ to track their respective reference trajectories. When tracking of two protein concentrations needs to be ensured, the cost function will depend on these two protein concentrations $p^1$ and $p^2$ and on the corresponding references $r_t^1$ and $r_t^2$. Note that here and in the sequel $r_t^i$ stands for the $i$-th element of the vector $\boldsymbol{r}_t$. When tracking needs to be ensured for only one protein concentration (i.e., $p^2$), the cost function will only depend on $p^2$ and the scalar reference $r_t^2$. In both cases, the cost depends on the control signals $u^i$ in order to penalise the use of light stimuli and thus minimise the metabolic burden caused by heterologous gene expression. The cost is defined using a distance between the observed state $p^i$ and the reference trajectories $r_t^i$:

$$\boldsymbol{c}(\boldsymbol{p},\boldsymbol{r},\boldsymbol{u}) = 100\cdot\alpha(p^1-r_t^1)^2 + 100\cdot(p^2-r_t^2)^2 + 0.05\cdot u^1 + 0.05\cdot u^2$$

where $\alpha$ will be equal to one or zero depending on how many reference trajectories we want to track simultaneously.

The discount factor $\gamma$ is equal to 0.75, the choice of which is guided by considerations similar to the ones in [8]. The stopping criterion is simply a bound on the number of iterations, which for the purpose of this paper is 30. The set of system transitions is generated according to the following procedure. 300 system trajectories starting in a random initial state are generated. The control actions applied to generate the trajectories are random as well. Every trajectory has at most 300 samples. These state transitions are then gathered in the set $\mathcal{F}$. At every step every $Q^{\boldsymbol{r}_i}$ function is approximated using the regression algorithm EXTremly RAndomized Trees (EXTRA Trees), which was shown to be an effective regression algorithm for the fitted $Q$ iteration framework [17]. The parameters for the algorithm are set to the default values from [8].

The algorithm is implemented in Python using the machine learning [9], parallelisation [10], graphics [11] and scientific computation [12] toolboxes.

### C. Results

*1) One sinusoidal reference trajectory, different periods:* In this example, we are going to force the concentration of the protein 2 to track a sinusoid with different periods. The parameter $\alpha$ is set to zero and the sinusoid is chosen to resemble the natural oscillations in terms of amplitude and mean value:

$$r_t^2 = 8 + 7\cdot\sin(Tt/(2\pi)).$$

We are testing the algorithm for the following periods $T = 50, 150, 250$. We can increase the concentration of the protein 2 directly trough application of the control signal $u^2$. We can also decrease the concentration of the protein 2 through increasing the concentration of the protein 1, which acts as a repressor for gene 2. The results of several experiments are depicted in Figure 3. The natural oscillations have a period of approximately 150 arbitrary time units, therefore the blue dashed curve is easiest to follow. Using repression by gene 1, the algorithm manages to find a schedule of light pulses that allows to track the dotted red curve, even though the period is much larger than 150. However, due to the inherent dynamics of the system, the algorithm cannot bring down the concentration of protein 2 fast enough to properly track the cyan curve. The repression by gene 1 is not strong enough in this case to allow accurate tracking. It is important to remark that tracking of the trajectory by protein 2 results in damping of the oscillations in the other proteins and mRNA dynamics.

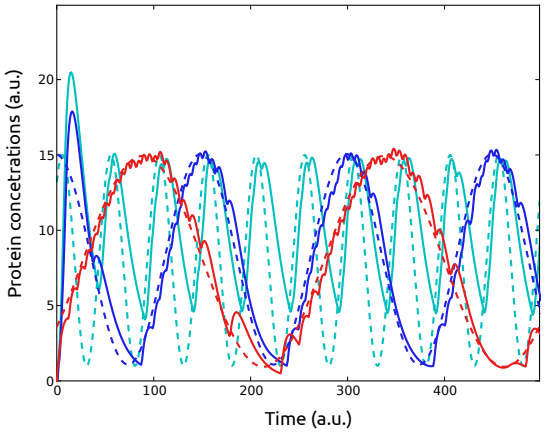


Fig. 3. Tracking a sinusoid in a six gene repressilator. The figure is colour coded: lines with the same colour correspond to the same experiment. Each solid curve represents the time evolution of the concentration protein 2, which attempts to follow the same colour reference trajectory, represented by a dashed line. The cyan colour corresponds to the period $T = 50$, the blue colour to $T = 150$ and the red colour to $T = 250$.

*2) Two sinusoidal reference trajectories:* For this experiment we chose two sinusoids (i.e., $\alpha$ is equal to one), where the second sinusoid lags behind the first one:

$$r_t^1 = 8 + 7\cdot\sin(200t/(2\pi))$$
$$r_t^2 = 8 + 7\cdot\sin(200(t+200/3)/(2\pi))$$

Genes, their protein products and the corresponding reference trajectories are colour coded in this example: the blue colour corresponds to protein 1 and the cyan colour corresponds to protein 2. The "blue" gene represses the "cyan" one, hence an increase in the "blue" protein concentration can be used to decrease the concentration of the "cyan" protein. However, the concentration of the "blue" protein cannot be decreased directly. The sinusoids are chosen with approximate knowledge of the natural oscillation dynamics: in terms of amplitude and mean value of oscillations. Their period is chosen equal to 200 time samples. As depicted in Figure 4, the algorithm can force the concentration of the first two proteins to follow both sinusoids. Moreover, numerous experiments were conducted starting from different initial conditions, which yielded similar tracking results after the initial transient

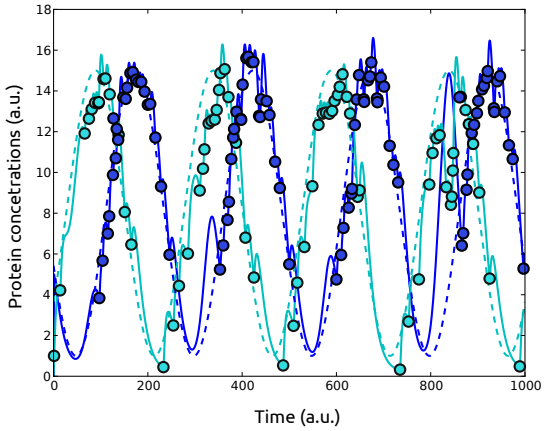


Fig. 4. Tracking two sinusoids in a six gene repressilator. The blue colour represents gene 1 in the repressilator, which represses gene 2 represented by the cyan colour. The dashed lines represent the reference trajectories; the solid lines represent the protein concentration tracking the reference with identical colour; finally, the coloured circled dots correspond to time instant at which control inputs in the form of light pulses were applied. Due to restrictions imposed by the system's dynamics the cyan reference should lag behind the blue one and the lag should be large enough to ensure appropriate tracking.

period had elapsed. It is worth noting the interesting behaviour exhibited by the "blue" protein concentration: At some point the protein concentration starts growing without any light stimulation. This can be explained by the repressilator oscillatory dynamics, where the protein's concentration can grow periodically. Moreover, since the "blue" protein concentration cannot be decreased directly it can grow significantly as can be observed during the time range between 800 and 1000 time units. It also means that controlling this system with a larger period will be harder due to this fast growth of the "blue" protein concentration induced by the dynamics of the system.

*3) Two ramp reference trajectories:* The two ramp tracking setting is very similar to the two sinusoids tracking setting. However, in the situation depicted in upper panel of Figure 5 unsuccessful simultaneous tracking of two ramps is occurring. The reasons for such a behaviour are the same as above. The difference is that they are more pronounced in this case due to large time intervals during which a low reference value needs to be followed by one of the proteins. Note that the first ramp is followed almost perfectly by both protein concentrations. However, after a long time interval of low reference value, the concentration of the "blue" protein starts to grow due the inherent dynamics of the system. With a modified ramp such behaviour disappears as depicted in the lower panel of Figure 5. Even though tracking is not perfect, the algorithm manages to find an approximate solution to the optimal control problem, which is quite remarkable given the very limited amount of information provided by the input-output data in such setting.

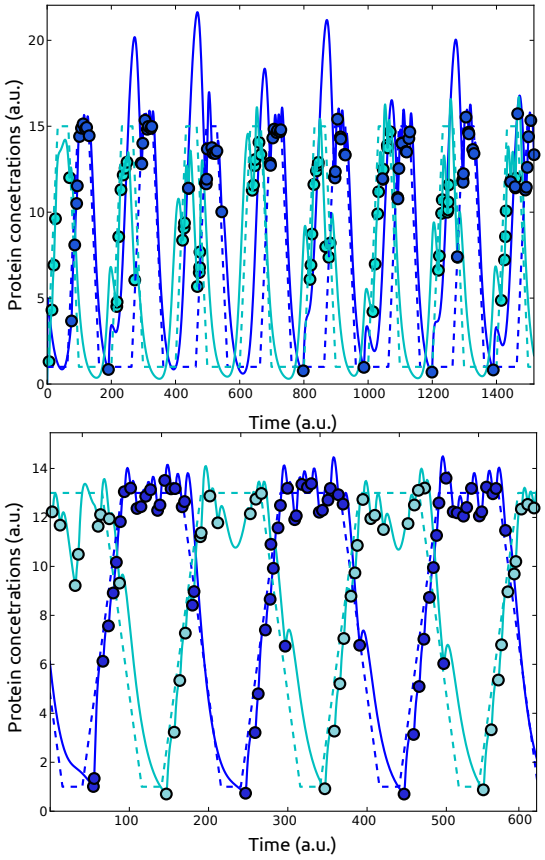


Fig. 5. Tracking the ramps in a six gene repressilator. Similar colour and line coding is used as in the figure above. In the upper panel, the algorithm finds it hard to keep both genes at low levels due to the inherent constraints imposed by the dynamics of the generalised repressilator; hence, the system cannot follow these ramp trajectories. Note that even though the first period is followed perfectly, the "blue" protein then starts to grow and we have no means to decrease its concentration. In the lower panel the ramp is changed so that the period of time spent at low concentrations is much shorter. This solves the above described issue.

## V. Discussion and Conclusion

In this paper, we have presented a periodic reference tracking reinforcement learning algorithm. The algorithm is based on the established fitted Q iteration framework, and inherits its properties. The proposed algorithm makes full use of the fact that the reference trajectory is known in advance, which results in better sample efficiency in comparison with other approaches proposed in the literature.

The algorithm is illustrated on the problem of tracking a periodic trajectory for the generalised repressilator system. This system has received considerable attention from the synthetic and systems biology community due to its ability to produce long-lived oscillatory behaviours. The algorithm was able to find near optimal solutions to the periodic tracking reference problem, even when the period of the reference trajectory was smaller than the natural period of oscillation of the system.

## References

[1] Shinji Hara, Yutaka Yamamoto, Tohru Omata, and Michio Nakano. Repetitive control system: a new type servo system for periodic exogenous signals. *Automatic Control, IEEE Tran on*, 33(7):659–668, 1988.

[2] R. Fonteneau, S.A. Murphy, L. Wehenkel, and D. Ernst. Batch mode reinforcement learning based on the synthesis of artificial trajectories. *Annals of Operations Research*, pages 1–34, 2012.

[3] L. Buşoniu, R. Babuška, B. De Schutter, and D. Ernst. *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Pr I Llc, 2010.

[4] R.S. Sutton and A.G. Barto. *Reinforcement Learning, an Introduction*. MIT Press, 1998.

[5] Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. *Machine Learning: ECML 2005*, pages 317–328, 2005.

[6] G.-B. Stan, F. Belmudes, R. Fonteneau, F. Zeggwagh, M.-A. Lefebvre, C. Michelet, and D. Ernst. Modelling the influence of activation-induced apoptosis of CD4+ and CD8+ T-cells on the immune system response of a HIV-infected patient. *IET Systems Biology*, 2(2):94–102, 2008.

[7] Susan A Murphy. Optimal dynamic treatment regimes. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65(2):331–355, 2003.

[8] D. Ernst, P. Geurts, and L. Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, April 2005.

[9] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

[10] Joblib: running python function as pipeline jobs. http://packages.python.org/joblib.

[11] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[12] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python. 2001–, http://www.scipy.org/.

[13] M.B. Elowitz and S. Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(6767):335–338, 2000.

[14] H. Smith. Oscillations and multiple steady states in a cyclic gene model with repression. *J Math Biol*, 25(15):169–190, Jul 1987.

[15] Natalja Strelkowa and Mauricio Barahona. Switchable genetic oscillator operating in quasi-stable mode. *J R Soc Interface*, 7(48):1071–1082, 2010.

[16] A. Sootla, N. Strelkowa, D. Ernst, M. Barahona, and G.B. Stan. Toggling the genetic switch using reinforcement learning. In *submission to the 52nd Conf. on Decision and Control*, Dec. 2013, http://www3.imperial.ac.uk/people/a.sootla/publications.

[17] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.