52nd IEEE Conference on Decision and Control
December 10-13, 2013. Florence, Italy

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 application for the tracking problem.**

***Index Terms*— reference tracking; fitted Q iteration; reinforcement learning, synthetic biology; gene regulatory networks**

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 $\boldsymbol{d}_t$ at time t. The next step is to 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 data-based optimal control. 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 four-tuple $\{\boldsymbol{n}, \boldsymbol{u}, \boldsymbol{r}, \boldsymbol{n}^+\}$, where $\boldsymbol{n}^+$ denotes a successor state of the system in a state $\boldsymbol{n}$ subjected to an input $\boldsymbol{u}$, and $\boldsymbol{r}$ is an instantaneous reward for performing an action $\boldsymbol{u}$ at a state $\boldsymbol{n}$. We assume that $\boldsymbol{r}$ can be computed for every state-action pairs; therefore, we consider only triplets $\{\boldsymbol{n}, \boldsymbol{u}, \boldsymbol{n}^+\}$ as one-step transitions. 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 (FQI) 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 trajectories.

A. Sootla and G.-B. Stan (corresponding author) are with the Centre for Synthetic Biology and Innovation and the Department of Bioengineering, M. Barahona is with the Department of Mathematics, Imperial College London, UK {a.sootla, g.stan, m.barahona}@imperial.ac.uk, N. Strelkowa is with Boehringer-Ingelheim Pharma GmbH & Co KG., Germany, natalja.strelkowa@boehringer-ingelheim.com, and D. Ernst is with the Montefiore Institute, University of Liège, Belgium dernst@ulg.ac.be, A. Sootla and G.-B. Stan acknowledge the support of EPSRC through the project EP/J014214/1. Dr Stan is further supported by the EPSRC Science and Innovation Award EP/G036004/1. N. Strelkowa contributed to the paper while being a PhD student at Imperial College, and was supported by the Wellcome Trust grant 080711/Z/06/Z. D. Ernst acknowledges support of the Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office.

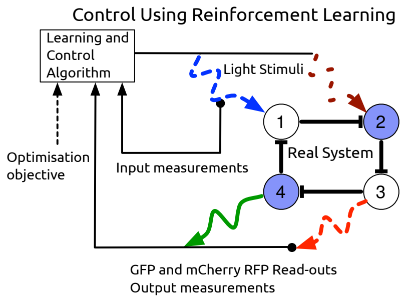
Control Using Reinforcement Learning
Learning and Control Algorithm
Light Stimuli
Optimisation objective
Input measurements
Real System
1
2
3
4
GFP and mCherry RFP Read-outs
Output measurements

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 gene 1.

The focus of this paper is on an extension of FQI 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 and is available at www3.imperial.ac.uk/people/a.sootla.

To illustrate our reference tracking method, we consider its application to synthetic biology gene regulatory networks, specifically to a network called a generalised repressilator. The repressilator is a ring of three mutually repressing genes pioneered in [9], and theoretically generalised to rings consisting of more than three genes in [10]. 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 [11]. 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 [12], where we consider a regulation problem of a toggle switch system.

The paper is organised as follows. Section II covers the FQI Algorithm. In Section III, an extension of FQI to the reference tracking problem is presented. In Section IV our illustrative application, reference tracking for the generalised repressilator system, is given. Section V concludes the paper.

II. Fitted Q Iteration (FQI)

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 $\boldsymbol{U}$. Consider also, associated with this dynamical system, an optimal control problem, which is defined in terms of the maximisation of an infinite sum of discounted rewards:

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

where $\boldsymbol{r}(\cdot,\cdot)$ is a reward function, $\mu(\cdot)$ is a mapping from the state-space onto $\boldsymbol{U}$, which is called the feedback control policy, and γ is a positive constant smaller than one, which is called the discount factor. In this paper, it is assumed that $\boldsymbol{r}(\cdot,\cdot)$ 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 $f(\boldsymbol{n}_l, \boldsymbol{u}_l)$).

The central object in FQI is the Q function, which is defined as follows:

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

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

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

which can be theoretically solved using an iterative procedure

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

where $Q_0 = \boldsymbol{r}$ and Q_∞ is the unique solution to (2) due to the fact that $T(Q) = \boldsymbol{r} + \gamma \max_{\boldsymbol{u}' \in \boldsymbol{U}} Q$ is a contraction mapping (cf. [3]). It also provides a convenient expression for the computation of the optimal feedback control policy:

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

Algorithm 1 Fitted Q Iteration (FQI)

```
Inputs: Set of one-step system transitions $\mathcal{F} = \{\boldsymbol{n}_l, \boldsymbol{u}_l, \boldsymbol{n}_l^+\}_{l=1}^{\#\mathcal{F}}$, reward $\boldsymbol{r}(\cdot,\cdot)$, stopping criterion
Outputs: Policy $\hat{\mu}^*(\boldsymbol{n})$
  $k \leftarrow 0$ and $\hat{Q}_0(\cdot,\cdot) \leftarrow \boldsymbol{r}(\cdot,\cdot)$
  repeat
    $k \leftarrow k+1$ and 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)
```

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 FQI algorithm provides a solution to this problem from the sole knowledge of $\mathcal{F}$. The solution is obtained by computing a sequence of functions $\hat{Q}_1$, $\hat{Q}_2$, ... that approximates the sequence Q_1, Q_2, ... Indeed, let $\hat{Q}_0 = \boldsymbol{r}$ and for every $(\boldsymbol{n}_l, \boldsymbol{u}_l, \boldsymbol{n}_l^+)$ in $\mathcal{F}$ compute:

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

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 regression (e.g., EXTremly RAndomized or EXTRA Trees algorithm [13]). Now the iterative procedure is derived by generalising (4) as follows:

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

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

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

A major property of the FQI algorithm is convergence [8]. 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. A sufficient condition for convergence is the use of kernel-based approximators on the regression step [14]. Note that to ensure convergence using the EXTRA Trees algorithm, it is required to fix the structure of the trees, i.e., the kernel representation of the $\hat{Q}$ function approximation must not change from one iteration to another. However, a variable kernel representation of EXTRA Trees may provide better numerical results in comparison with the fixed one during the initial iterations [8]. Hence, one may fix the structure of the regression trees after a certain amount of iterations as a trade-off between a guarantee on convergence and good numerical performance. A stopping criterion can be also derived. Let $\hat{V}^*$ be defined as $\max\limits_{\boldsymbol{u} \in \boldsymbol{U}} \lim\limits_{N \to \infty} \hat{Q}_N$, and $\hat{V}^N = \max\limits_{\boldsymbol{u} \in \boldsymbol{U}} \hat{Q}_N$ be a value function after N iterations, then:

$$\|\hat{V}^N - \hat{V}^*\|_\infty \leq 2\frac{\gamma^N \|\boldsymbol{r}\|_\infty}{(1-\gamma)^2}. \tag{7}$$

We solve our control problem using one-step transitions; therefore, it can be assumed without loss of generality that the state-space is bounded and so is the reward function. Hence, the number of iterations can be bounded in advance based on the prescribed accuracy and the value of γ. The resulting iterative method is outlined in Algorithm 1.

III. Periodic Reference Tracking Using the Fitted Q Iteration Algorithm

Let $f(\cdot,\cdot)$ be an unknown function, and $g(\cdot)$ be a known function with a period T. Consider, a system:

$$\begin{aligned} \boldsymbol{n}_{t+1} &= f(\boldsymbol{n}_t, \boldsymbol{u}_t) \\ \boldsymbol{d}_{t+1} &= g(\boldsymbol{d}_t), \end{aligned} \tag{8}$$

where $\boldsymbol{d}$ takes only a finite number of values $\{\boldsymbol{v}_i\}_{i=1}^T$, $g(\boldsymbol{v}_i)$ 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:

$$\begin{gathered} \max_{\mu(\cdot,\cdot)} \sum_{i=t}^{\infty} \gamma^{i-t} \boldsymbol{r}(\boldsymbol{\rho}(\boldsymbol{n}_i, \boldsymbol{d}_i), \boldsymbol{n}_i, \boldsymbol{u}_i) \\ \text{subject to system dynamics (8), and} \\ \mu(\boldsymbol{n}_t, \boldsymbol{d}_t) = \boldsymbol{u}_t \in \boldsymbol{U}, \end{gathered} \tag{9}$$

where $\boldsymbol{r}$ is a given instantaneous reward function, $\boldsymbol{\rho}(\cdot,\cdot)$ is a function defining the distance between the current state $\boldsymbol{n}_i$ and reference $\boldsymbol{d}_i$, and $\mu(\cdot,\cdot)$ is a feedback control policy. In order to track the reference $\boldsymbol{d}$, increasing the value of the reward $\boldsymbol{r}$ must reduce the distance $\boldsymbol{\rho}$. The instantaneous reward 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 FQI, 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 $\boldsymbol{u}_l$.

The variable $\boldsymbol{d}$ can be seen as an additional state in the extended state-space $\{\boldsymbol{n}, \boldsymbol{d}\}$. 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:

$$\begin{aligned} Q_k(\boldsymbol{n}, \boldsymbol{d}, \boldsymbol{u}) = \boldsymbol{r}(\boldsymbol{\rho}(\boldsymbol{n}, \boldsymbol{d}), \boldsymbol{n}, \boldsymbol{u}) + \\ \max_{\boldsymbol{u}' \in \boldsymbol{U}} Q_{k-1}(\boldsymbol{n}^+, g(\boldsymbol{d}), \boldsymbol{u}') \quad \forall \boldsymbol{n}, \boldsymbol{d}, \boldsymbol{u}, \end{aligned} \tag{10}$$

where Q_0 is equal to $\boldsymbol{r}$. 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:

$$\mu(\boldsymbol{n}, \boldsymbol{d}) = \max_{\boldsymbol{u} \in \boldsymbol{U}} Q^*(\boldsymbol{n}, \boldsymbol{d}, \boldsymbol{u}).$$

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 $\boldsymbol{d}$, $\mathcal{F}$ should now include $\boldsymbol{d}$ as well. Since the time evolution of $\boldsymbol{d}$ 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 $i = 1, \dots, T$ and $l = 1, \dots, \#\mathcal{F}$. 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 $\boldsymbol{r}$ and the next iterates can be obtained as follows:

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

According to the FQI 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 time evolution of $\boldsymbol{d}$ is known in advance and $\boldsymbol{d}$ takes a finite number of values. To make these ideas more transparent, (11) is rewritten using a different notation as follows:

$$\begin{aligned} \hat{Q}_k^{\boldsymbol{v}_i}(\boldsymbol{n}_l, \boldsymbol{u}_l) = \boldsymbol{r}(\boldsymbol{\rho}(\boldsymbol{n}_l, \boldsymbol{v}_i), \boldsymbol{n}_l, \boldsymbol{u}_l) + \\ \max_{\boldsymbol{u} \in \boldsymbol{U}} \hat{Q}_{k-1}^{g(\boldsymbol{v}_i)}(\boldsymbol{n}_l^+, \boldsymbol{u}) \quad \forall l, \boldsymbol{v}_i, \end{aligned} \tag{12}$$

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 N, a near-optimal policy can be computed as follows:

$$\hat{\mu}_N^*(\boldsymbol{n}, \boldsymbol{d}) = \max_{\boldsymbol{u} \in \boldsymbol{U}} \hat{Q}_N^{\boldsymbol{d}}(\boldsymbol{n}, \boldsymbol{u}). \tag{13}$$

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 $\hat{Q}_k^{\boldsymbol{v}_i}$ 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 FQI algorithm in [8]. Moreover, the bound (7) is valid for Algorithm 2.

The proof is a straightforward adaptation of the convergence proof in [8] and therefore omitted. The remarks on FQI convergence are valid for this case, as well. Since only the

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 $\{\boldsymbol{v}_i\}_{i=1}^{T}$, reward $\boldsymbol{r}(\boldsymbol{\rho}(\cdot,\cdot),\cdot,\cdot)$, stopping criterion
Outputs: Policy $\hat{\mu}^*(\boldsymbol{n}, \boldsymbol{d})$
  $k \leftarrow 0$ and $\hat{Q}_0(\cdot,\cdot) \leftarrow \boldsymbol{r}(\boldsymbol{\rho}(\cdot,\cdot),\cdot,\cdot)$
  repeat
    $k \leftarrow k+1$ and compute (12) 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 (13)
```

regression step is modified, Algorithm 2 can be applied to the stochastic systems as well. If we assume that the reward function is time-independent, i.e., $\boldsymbol{d}$ is constant, and $g(\boldsymbol{v})$ is equal to $\boldsymbol{v}$, Algorithm 2 is reduced to Algorithm 1, since equation (12) becomes:

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

The regression can be applied directly to one-step transitions $\mathcal{TF} = \{\boldsymbol{n}_l, \boldsymbol{v}_i, \boldsymbol{u}_l, \boldsymbol{n}_l^+, \boldsymbol{v}^+\}_{i,l=1}^{l=\#\mathcal{F}, i=T}$. The computational cost of this large regression problem is $O(T \cdot \#\mathcal{F} \log(T \cdot \#\mathcal{F})$ [13]. If we follow the proposed algorithm and split this large regression into T smaller regression problems, we obtain $O(T \cdot \#\mathcal{F} \log(\#\mathcal{F}))$, which is a reduction of $O(T \cdot \#\mathcal{F} \log(T))$ floating point operations.

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 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:

$$\begin{aligned} \dot{m}^i &= \frac{c_1^i}{1+(p^{i-1})^2} - c_2^i m^i + \delta_{i1} b_1 u^1 + \delta_{i2} b_2 u^2 \\ \dot{p}^i &= c_3^i m^i - c_4^i p^i, \end{aligned} \tag{14}$$

where i is an integer from one to six, and δ_{ij} is equal to one, if i is equal to j, and equal to zero otherwise. We assume that the protein concentrations are given as readouts,

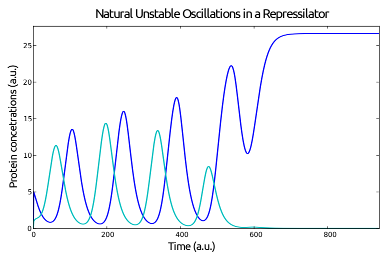

Fig. 2. Natural oscillatory behaviour of a generalised repressilator system consisting of 6 genes. The blue line represents the time evolution of the protein concentration produced by the expression of gene 1; the cyan line represents the time evolution of the protein concentration produced by the expression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

for instance via fluorescent markers [15], [16] (e.g., green fluorescent protein, GFP or red fluorescent protein, mCherry). On the other hand, we assume that the control inputs are implemented as light pulses of specific non-overlapping wavelengths activating a photo-sensitive promoter controlling the expression of genes 1 and 2 [17], [18]. When a photo-sensitive promoter is activated through a light pulse the concentration of the gene product 1 (or 2) is increased by a small amount through the expression of gene 1 (or 2). 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 [9], 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. We chose the parameter values according to [11]:

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

where it is shown that the dynamics of this system exhibit 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 time units.

B. Algorithm Parameters and Implementation

The instantaneous reward function is defined differently for each 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 two 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 reward function will depend on these two protein concentrations p^1 and p^2 and on the corresponding references d_t^1 and d_t^2. Note that here and in the sequel d_t^i stands for the i-th element of the vector $\boldsymbol{d}_t$. When tracking needs to be ensured for only one protein concentration (i.e., p^2), the reward function will only depend on p^2 and the scalar reference d_t^2. In both cases, the reward 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 reward is defined using a distance between the observed state p^i and the reference trajectories d_t^i:

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

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

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 $\mathcal{F}$. The discount factor γ 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. At every iteration every $Q^{\boldsymbol{v}_i}$ function is approximated using EXTRA Trees, which was shown to be an effective regression algorithm for the FQI framework [13]. The parameters for the algorithm are set to the default values from [8]. The algorithm is implemented in Python using the machine learning [19], parallelisation [20], graphics [21] and scientific computation [22] toolboxes.

C. Results

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. Here $\alpha = 0$ and the sinusoid is chosen to resemble the natural oscillations in terms of amplitude and offset from zero:

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

We test the algorithm for the following periods $T = 50$, 150, 250. We can increase the concentration of the protein 2 directly through 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 simulations are depicted in Figure 3. The natural oscillations have a period of approximately 150 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.

Two sinusoidal reference trajectories. For this simulation we chose two sinusoids (i.e., α is equal to one), where the second sinusoid lags behind the first one:

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

Genes, their protein products and the corresponding reference trajectories are colour coded in this example: the blue

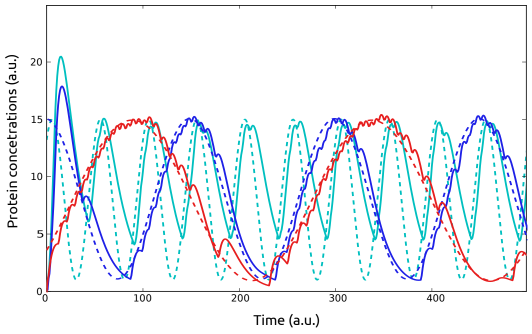

Fig. 3. Tracking a sinusoid in a six gene repressilator. The figure is colour coded: lines with the same colour correspond to the same simulation. Each solid curve represents the time evolution of the concentration protein 2, which attempts to follow the same colour reference trajectory, represented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

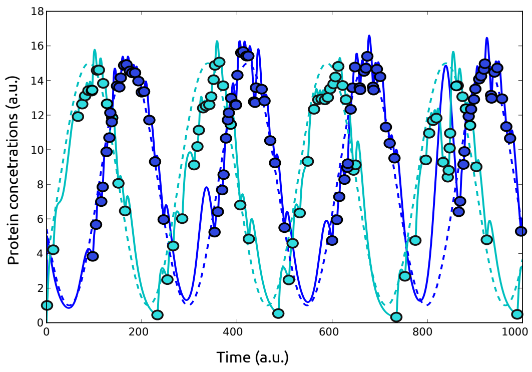

Fig. 4. Tracking two sinusoids in a six gene repressilator. The blue colour represents gene 1 in the repressilator, which represses gene 2 represented by the cyan colour. The dashed lines represent the reference trajectories; the solid lines represent the protein concentration tracking the reference 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.

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 simulations were conducted starting from different initial conditions, which yielded similar tracking results after the initial transient 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

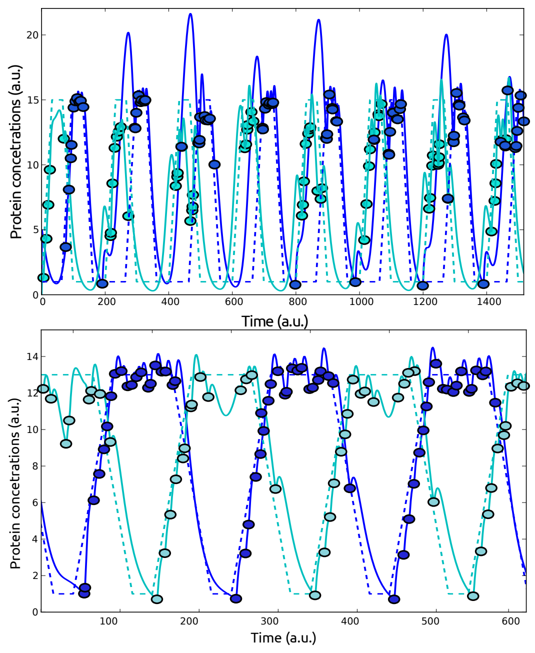

Fig. 5. Tracking the ramps in a six gene repressilator. Similar colour and line coding is used as in the figure above. In the upper panel, the algorithm finds it hard to keep both genes at low levels due to the inherent constraints imposed by the dynamics of the generalised repressilator; hence, the 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.

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.

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 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.

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 *a priori* knowledge of the reference trajectory, 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 communities 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] S. Hara, Y. Yamamoto, T. Omata, and M. Nakano, "Repetitive control system: a new type servo system for periodic exogenous signals," *IEEE Trans. Autom. Control*, vol. 33, no. 7, pp. 659–668, 1988.

[2] R. Fonteneau, S. Murphy, L. Wehenkel, and D. Ernst, "Batch mode reinforcement learning based on the synthesis of artificial trajectories," *Annals of Operations Research*, vol. 208, no. 1, pp. 383–416, 2013.

[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. Sutton and A. Barto, *Reinforcement Learning, an Introduction*. MIT Press, 1998.

[5] M. Riedmiller, "Neural fitted Q iteration – first experiences with a data efficient neural reinforcement learning method," in *Machine Learning: ECML 2005*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2005, vol. 3720, pp. 317–328.

[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*, vol. 2, no. 2, pp. 94–102, 2008.

[7] S. A. Murphy, "Optimal dynamic treatment regimes," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 65, no. 2, pp. 331–355, 2003.

[8] D. Ernst, P. Geurts, and L. Wehenkel, "Tree-based batch mode reinforcement learning," *J Mach Learn Res*, vol. 6, pp. 503–556, 2005.

[9] M. Elowitz and S. Leibler, "A synthetic oscillatory network of transcriptional regulators," *Nature*, vol. 403, no. 6767, pp. 335–338, 2000.

[10] H. Smith, "Oscillations and multiple steady states in a cyclic gene model with repression," *J Math Biol*, vol. 25, no. 15, pp. 169–190, Jul 1987.

[11] N. Strelkowa and M. Barahona, "Switchable genetic oscillator operating in quasi-stable mode," *J R Soc Interface*, vol. 7, no. 48, pp. 1071–1082, 2010.

[12] A. Sootla, N. Strelkowa, D. Ernst, M. Barahona, and G. Stan, "Toggling the genetic switch using reinforcement learning," March 2013, arXiv:1303.3183.

[13] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, no. 1, pp. 3–42, 2006.

[14] D. Ormoneit and S. Sen, "Kernel-based reinforcement learning," *Machine Learning*, vol. 49, no. 2-3, pp. 161–178, 2002.

[15] L. Cai, N. Friedman, and X. S. Xie, "Stochastic protein expression in individual cells at the single molecule level," *Nature*, vol. 440, pp. 358–362, 2006.

[16] M. R. Bennett and J. Hasty, "Microfluidic devices for measuring gene network dynamics in single cells," *Nat Rev Genet*, vol. 10, no. 9, pp. 628–638, September 2009.

[17] S. Shimizu-Sato, E. Huq, J. M. Tepperman, and P. H. Quail, "A light-switchable gene promoter system," *Nat Biotech*, vol. 20, no. 10, pp. 1041–1044, 2002.

[18] A. Levskaya, O. D. Weiner, W. A. Lim, and C. A. Voigt, "Spatiotemporal control of cell signalling using a light-switchable protein interaction," *Nature*, vol. 461, pp. 997–1001, 2009.

[19] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in python," *J Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[20] "Joblib: running python function as pipeline jobs." [Online]. Available: http://packages.python.org/joblib

[21] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing In Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[22] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," 2001–. [Online]. Available: http://www.scipy.org/