2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Locally Weighted Least Squares Policy Iteration for Model-free Learning in Uncertain Environments

Matthew Howard and Yoshihiko Nakamura

Abstract— **This paper introduces Locally Weighted Least Squares Policy Iteration for learning approximate optimal control in settings where models of the dynamics and cost function are either unavailable or hard to obtain. Building on recent advances in Least Squares Temporal Difference Learning, the proposed approach is able to learn from data collected from interactions with a system, in order to build a global control policy based on localised models of the state-action value function. Evaluations are reported characterising learning performance for non-linear control problems including an under-powered pendulum swing-up task, and a robotic door-opening problem under different dynamical conditions.**

I. Introduction

In recent years, a number of methods for the approximate solution of optimal control problems have been proposed. Examples include variants on Differential Dynamic Programming (DDP) [1], [2], Iterative Linear Quadratic Regulator design (ILQR) [3], [4] and Path Integral Policy Iteration (PI^2) [5]. A common strategy in these approaches is their use of local approximations of the system dynamics and the cost, in order to compute local optimal control laws along a nominal trajectory. This simplifies potentially high-dimensional, non-linear control problems within local regions of the state space, enabling their approximate solution.

While such approaches have been shown to be effective in many robotic problems, their applicability in many situations may be limited by their requirement for (i) an accurate model of the system dynamics, (ii) a closed-form expression for the cost function, and (iii) the requirement for accurate state estimation from sensor measurements. These are necessary for constructing the local approximations to the dynamics and cost, and thereby the local solution to the control problem considered.

In many practical situations, however, such information is either unavailable or difficult to obtain. For example, in tasks where robots are asked to interact with objects in an unstructured environment (such as in a disaster response scenario), models of the dynamics are unlikely to be unavailable, and may change with time (e.g., as the disaster unfolds). Furthermore, in absence of such models of environmental dynamics, it is usually not possible to specify cost functions in closed form, since this implies knowledge of how the desired states and control inputs affect the desired outcomes of the behaviour. For instance, consider a cost function specifying maximum distance throwing of an object with unknown aerodynamics: if the mass, air-resistance, wind-speed etc. are unknown, the cost (i.e., distance thrown) for a given controller cannot be predicted prior to its execution.

The approach taken in this paper continues in the spirit of applying local approximation techniques to simplify non-linear optimal control problems. However, rather than relying on model knowledge, we explore the use of model-free reinforcement learning techniques, based on recent advances in Least Squares Temporal Difference (LSTD) learning [6], [7], [8]. The proposed approach is inspired by local learning techniques [9], [10] that have been used for learning models of the dynamics of robotic systems. In contrast to these prior works, however, here control policies are directly learnt from data, avoiding sources of error from inaccuracies in the dynamics model. Empirical evaluations are reported characterising performance of learnt controllers for non-linear problems, including an under-powered pendulum swing-up task [11] and a simulated robotic door-opening task.

II. Problem Definition

In this paper, we focus on approximate optimal control in settings where model information (i.e., the form and/or parameter values) about (i) the dynamics and (ii) the cost, is not explicitly available.

Formally, it is assumed that the deterministic, discrete-time approximation of the (unknown) dynamics has the form

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

where $\mathbf{u}_t \in \mathbb{R}^{\mathcal{Q}}$ is a vector of control inputs with restriction on their admissible values $\mathbf{u} \in [\mathbf{u}^{min}, \mathbf{u}^{max}]$, and $\mathbf{x}_t \in \mathbb{R}^{\mathcal{P}}$ is the state at time step t. Access to the latter is assumed to be restricted to observations $\mathbf{y}_t \in \mathbb{R}^{\mathcal{R}}$ through some bijective sensor function[1]

$$\mathbf{y}_t = \mathbf{h}(\mathbf{x}_t) \tag{2}$$

the functional form and/or parameters of which are also assumed unknown.

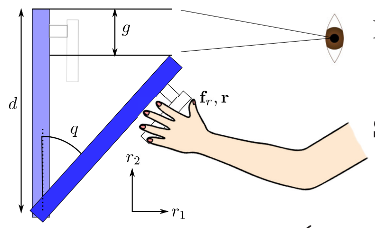
The goal is to find a controller (policy) $\mathbf{u}_t = \boldsymbol{\pi}(\mathbf{y}_t)$ that minimises the long-term cost

$$J = \sum_{t=0}^{\infty} \iota(\mathbf{x}_t, \mathbf{u}_t) \tag{3}$$

where $\iota(\mathbf{x}_t, \mathbf{u}_t)$ (or, equivalently, $j(\mathbf{y}_t, \mathbf{u}_t) := \iota(\mathbf{h}^{-1}(\mathbf{y}_t), \mathbf{u}_t)$ when expressed in terms of state observations) is the instantaneous cost incurred in applying command $\mathbf{u}_t$ in state

M. Howard (matthew.j.howard@kcl.ac.uk) is at the Dept. Informatics, Kings College London, UK. Y. Nakamura is at the Dept. Mechano-informatics, University of Tokyo, Japan. This research was supported by a JSPS Postdoctoral Fellowship (MH) and Grant-in-Aid for Scientific Research (MH,YN).

[1]Note that, the sensor function $\mathbf{h}$ represents a *transformation of the state space* of the original problem. This commonly arises in robotic systems where the minimal/most natural representation of the system state $\mathbf{x}$ may not be directly sensed. It is assumed that no *perceptual aliasing* [12] occurs in this transformation.



Dynamics:

$$I\ddot{q} = \mathbf{J}(q)^\top \mathbf{f}_r + \tau_e$$

Sensing:

$$\mathbf{y} = (\mathbf{r}^\top, \dot{\mathbf{r}}^\top)^\top$$

Cost: $$g(q) = \begin{cases} d(1-\cos(q)) & -\frac{\pi}{2} < q < \frac{\pi}{2} \\ d & otherwise \end{cases}$$

Fig. 1. Simplified model of door opening, formulated as a hidden state optimal control problem. Here, I denotes the door inertia, $\mathbf{J}(q)$ the Jacobian from the door angle to the hand contact point and τ_e the external torques (e.g., from friction, obstacles, door closing mechanisms, etc.). The terms $\mathbf{r}$ and $\dot{\mathbf{r}}$ are the hand position and velocity (that may be sensed, e.g., through the arm kinematics) and $\mathbf{f}_r$ is the Cartesian force applied by the hand. Other quantities are marked in the diagram.

$\mathbf{x}_t$. The latter is assumed to be unknown or, equivalently, incomputable due to lack of knowledge of (1)-(2). It is, however, assumed that observations, j_t, of the instantaneous cost can be made through sensory observation. For example, if the goal is to minimise energy in system powered by electric motors, j_t may correspond to the instantaneous power consumption, as measured by sensing the current drawn.

A. Motivating Example: Uncertainties in Opening a Door

To motivate these assumptions, consider the situation in which a robot is required to open a door in an unstructured and unfamiliar environment, as illustrated in Fig. 1. Even for this seemingly simple task, there are numerous uncertainties that make it difficult to determine a control strategy for this task. For example, there are uncertainties in

- *Dynamics parameters* such as the dimensions (height, width, thickness), inertial properties (total mass, mass distribution) and friction (at the hinges, around the frame). Estimation of these properties is challenging based purely on sensor information.
- *External dynamic effects*, e.g., from a closing mechanism, unseen obstacles blocking the door, 'jamming' of old, ill-maintained doors, other forms of damage to the door.
- *State estimation*, since direct feedback about the state of the door (e.g., hinge angular position and velocity) is unlikely to be available. The robot may be able to sense (e.g., through vision) or compute (through its forward kinematics) the location of an end-effector, but uncertainty about the door parameters (e.g., width, hinge location), makes this insufficient to compute the door state.
- *Cost predictions* due to uncertainties in the dependencies between the cost and the state/control parameters. For example, measuring performance by the clearance afforded when passing through the door, (i.e., the gap between door and frame, g, see Fig. 1), uncertainties exist as to whether the door opens inwards or outwards, or the gap size at a certain hinge angle (e.g., due to unknown door geometry).

One approach to tackling these problems is to use data, collected from interactions with the system, to design the optimal controller. Ideally, to maximise the autonomy of the robotic system, it should (i) work directly with available sensing with minimal assumptions on the form of the (unknown) dynamics and cost, and (ii) learn quickly and efficiently from limited data to minimise exploration (testing of different control strategies). In the following, an approach based on LSTD learning is proposed.

III. METHOD

The approach proposed in this paper is an extension of the Least Squares Policy Iteration (LSPI) algorithm [13], a model-free, off-policy reinforcement learning approach in the family of temporal difference (TD) learning methods.

A. Least Squares Policy Iteration

In its original form, LSPI learns an optimal control policy based on a global model of the state-action value function

$$Q^{\boldsymbol{\pi}}(\mathbf{y}_t, \mathbf{u}_t) = \sum_{s=0}^{\infty} \gamma^s j(\mathbf{y}_{s+t}, \mathbf{u}_{s+t}) \tag{4}$$

that predicts the future (discounted) cost incurred when starting in state $\mathbf{x}_t$ (where $\mathbf{y}_t = \mathbf{h}(\mathbf{x}_t)$), applying command $\mathbf{u}_t$ and acting according to policy $\mathbf{u} = \boldsymbol{\pi}(\mathbf{h}(\mathbf{x}))$ thereafter [14]. Here, $\gamma \in [0, 1)$ is a discount factor, and state transitions are dictated by the system dynamics (1). Learning proceeds by iterating between: (i) policy evaluation, in which an estimate of (4) is formed, and (ii) policy improvement, in which the control policy is updated according to the estimate.

1) Policy Evaluation: Policy evaluation consists of estimating (4) for a given control policy $\boldsymbol{\pi}(\mathbf{y})$. In the standard formulation of LSPI [13], the approximation of $Q^{\boldsymbol{\pi}}$ uses a global linear model

$$\tilde{Q}^{\boldsymbol{\pi}}(\mathbf{y}, \mathbf{u}) = \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{y}, \mathbf{u}) \tag{5}$$

where $\boldsymbol{\theta} \in \mathbb{R}^{\mathcal{M}}$ are the model parameters and $\boldsymbol{\phi}(\mathbf{y}, \mathbf{u}) \in \mathbb{R}^{\mathcal{M}}$ is a vector of features. The latter may be hand-selected for a given application, or consist of generic features (e.g., a set of polynomials, radial basis functions, etc.).

The parameters $\boldsymbol{\theta}$ are learnt from data through LSTD learning [13]. Specifically, samples $\{\mathbf{y}_n, \mathbf{u}_n, \bar{\mathbf{y}}_n, j_n\}_{n=1}^{\mathcal{N}}$ are assumed, where $\bar{\mathbf{y}}_n$ is the sensed state after the transition, when command $\mathbf{u}_n$ is applied in state $\mathbf{x}_n$ (i.e., $\bar{\mathbf{y}}_n = \mathbf{h}(\bar{\mathbf{x}}_n)$ with $\bar{\mathbf{x}}_n = \mathbf{f}(\mathbf{x}_n, \mathbf{u}_n)$ and $\mathbf{y}_n = \mathbf{h}(\mathbf{x}_n)$), and j_n is the instantaneous cost of making that transition. To form an estimate from these data, self-consistency with respect to the Bellman equation [7], [8] is sought, as measured by the temporal difference error

$$\boldsymbol{\delta} = \tilde{\mathbf{Q}}^{\boldsymbol{\pi}} - T^{\boldsymbol{\pi}}[\tilde{\mathbf{Q}}^{\boldsymbol{\pi}}] \tag{6}$$

where $\tilde{\mathbf{Q}}^{\boldsymbol{\pi}} \in \mathbb{R}^{\mathcal{N}}$ is the vector of model predictions (i.e., $\tilde{Q}^{\boldsymbol{\pi}}_n = \tilde{Q}^{\boldsymbol{\pi}}(\mathbf{y}_n, \mathbf{u}_n)$) and $T^{\boldsymbol{\pi}}[\tilde{\mathbf{Q}}^{\boldsymbol{\pi}}] \in \mathbb{R}^{\mathcal{N}}$ are the model predictions under the Bellman operator

$$T^{\boldsymbol{\pi}}[Q^{\boldsymbol{\pi}}(\mathbf{y}, \mathbf{u})] = j(\mathbf{y}, \mathbf{u}) + \gamma Q^{\boldsymbol{\pi}}(\bar{\mathbf{y}}, \boldsymbol{\pi}(\bar{\mathbf{y}})). \tag{7}$$

Assuming the linear model (5), the predictions $\tilde{\mathbf{Q}}^{\boldsymbol{\pi}}$ lie on the manifold corresponding to the column space of $\boldsymbol{\Phi} \in \mathbb{R}^{\mathcal{N} \times \mathcal{M}}$, where $\Phi_{mn} := \phi_m(\mathbf{y}_n, \mathbf{u}_n)$. This is not true of the vector $T^{\boldsymbol{\pi}}[\tilde{\mathbf{Q}}^{\boldsymbol{\pi}}]$ (due to the action of the Bellman operator) and so the least squares fixed point that minimises (6) within the column space of $\mathbf{\Phi}$ is sought instead, i.e., the point for which

$$\mathbf{\Phi}^\top(\tilde{\mathbf{Q}}^{\boldsymbol{\pi}} - T^{\boldsymbol{\pi}}[\tilde{\mathbf{Q}}^{\boldsymbol{\pi}}]) = \mathbf{0}. \tag{8}$$

Substituting $\tilde{\mathbf{Q}}^{\boldsymbol{\pi}} = \mathbf{\Phi}\boldsymbol{\theta}$, expanding the Bellman operator $T^{\boldsymbol{\pi}}[\tilde{\mathbf{Q}}^{\boldsymbol{\pi}}] = \mathbf{j} + \gamma\bar{\mathbf{\Phi}}\boldsymbol{\theta}$ (where $\bar{\Phi}_{mn} := \phi_m(\bar{\mathbf{y}}_n, \boldsymbol{\pi}(\bar{\mathbf{y}}_n))$ and $\mathbf{j} = (j_1, \cdots, j_{\mathcal{N}})^\top$) and rearranging, yields the optimal estimator

$$\boldsymbol{\theta}^* = (\mathbf{\Phi}^\top(\mathbf{\Phi} - \gamma\bar{\mathbf{\Phi}}))^{-1}\mathbf{\Phi}^\top\mathbf{j} = \mathbf{A}^{-1}\mathbf{b}. \tag{9}$$

where[2] $\mathbf{A} := \mathbf{\Phi}^\top(\mathbf{\Phi} - \gamma\bar{\mathbf{\Phi}}) \in \mathbb{R}^{\mathcal{M}\times\mathcal{M}}$ and $\mathbf{b} := \mathbf{\Phi}^\top\mathbf{j} \in \mathbb{R}^{\mathcal{M}}$. In a standard implementation, solving (9) has computational complexity of order $\mathcal{O}(\mathcal{M}^3)$.

2) Policy Improvement: Using the estimate (9), policy improvement is achieved by computing the optimal controls

$$\boldsymbol{\pi}^*(\mathbf{y}) = \arg\min_{\mathbf{u}} \tilde{Q}^{\boldsymbol{\pi}}(\mathbf{y}, \mathbf{u}) \tag{10}$$

i.e., the policy that obtains the minimum cost according to the value function. Depending on the choice of feature vectors $\boldsymbol{\phi}(\mathbf{y}, \mathbf{u})$, this may be done analytically or numerically[3]. The value function of the updated policy (10) is then estimated according to the evaluation step, and iterations between the policy evaluation and improvement steps continue, until convergence is met (measured, for example, according to a minimum change in the parameter vector $\boldsymbol{\theta}$) [13].

3) Selection of Basis Functions: The simplicity of LSPI, and its potential for off-policy, model-free learning make it an appealing candidate for the problems considered in Sec. II. However, one of the major difficulties in its direct application is the selection of an appropriate set of features $\boldsymbol{\phi}$ for forming the estimate of $Q^{\boldsymbol{\pi}}$. While hand-selection of features using domain knowledge may be possible in some cases, this is usually non-trivial since it is often not clear how the dynamics and cost in a given problem will impact on $Q^{\boldsymbol{\pi}}$. In the setting considered here, this is further exacerbated by the assumption that prior knowledge of the environmental dynamics, and their impact on the cost, are highly limited or absent.

An alternative, is to use basis functions forming a generic function approximator, such as polynomials, or a set of radial basis functions. However, difficulties are then encountered in terms of scalability: due to the very large number of basis functions required for adequate coverage of the state-action space and/or sufficient representational power for non-linear $Q^{\boldsymbol{\pi}}$. This causes a sharp increase in the computation of the optimal parameters through (9) due to the cubic dependence on $\mathcal{M}$, a demand for larger quantities of data [15], and can also cause difficulties in the policy optimisation step (10).

B. Locally Weighted Least Squares Policy Iteration

In this paper, instead of constructing a single, global estimator (5), the use of local learning techniques is investigated. In the spirit of fast, local regression techniques [9], [10], the estimate of $Q^{\boldsymbol{\pi}}$ is composed through a set of simple, but generic local models [6]. This avoids the need for prior domain knowledge in selecting features and also improves the computational efficiency, as outlined below.

It is proposed to learn a set of $\mathcal{K}$ local models,

$$\tilde{Q}^{\boldsymbol{\pi}}_k(\mathbf{y}, \mathbf{u}) = \boldsymbol{\theta}_k^\top\boldsymbol{\phi}(\mathbf{y}, \mathbf{u}) \tag{11}$$

that are combined into a global prediction through the weighted combination

$$\tilde{Q}^{\boldsymbol{\pi}}(\mathbf{y}, \mathbf{u}) = \sum_k^{\mathcal{K}} \hat{w}_k(\mathbf{y})\tilde{Q}^{\boldsymbol{\pi}}_k(\mathbf{y}, \mathbf{u}). \tag{12}$$

Here, $\hat{w}_k(\mathbf{y})$ is a normalised weighting function that assigns responsibility to the kth local model for predictions according to its distance in $\mathbf{y}$ to the query point. Several choices of weighting function exist, for example the tricube function [16] or normalised Gaussians [9], [10].

Note that, the global model (12) can potentially represent any non-linear $Q^{\boldsymbol{\pi}}$ with appropriate choice of the local models (or, more specifically, the basis functions $\boldsymbol{\phi}(\mathbf{y}, \mathbf{u})$). An efficient choice of $\boldsymbol{\phi}(\mathbf{y}, \mathbf{u})$ is discussed in Sec. III-B.2.

1) Policy Evaluation through Locally Weighted LSTD: For training the models (11), a weighted version of LSTD may be employed to localise the model fit. As noted in [6], [13], the relative importance of the temporal difference errors in (6) can be adjusted by re-weighing the elements of the vector of temporal difference errors $\boldsymbol{\delta} := (\delta_1, \ldots, \delta_{\mathcal{N}})^\top$.

In the proposed approach, the parameters $\boldsymbol{\theta}_k$ of the kth local model are estimated according to the weighted temporal difference error

$$\boldsymbol{\delta}_k = \mathbf{W}_k(\tilde{\mathbf{Q}}^{\boldsymbol{\pi}}_k - T^{\boldsymbol{\pi}}[\tilde{\mathbf{Q}}^{\boldsymbol{\pi}}_k]) \tag{13}$$

where $\mathbf{W}_k \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$ is a diagonal weighting matrix, with elements selected in order to give greater priority to fitting nearby data according to that model's weighting function[4], i.e., $(W_k)_{nn} = \hat{w}_k(\mathbf{y}_n)$.

Through a similar derivation as described in Sec. III-A.1, the solution to (13) can be computed in closed form (for each local model) as

$$\boldsymbol{\theta}^*_k = (\mathbf{\Phi}^\top\mathbf{W}_k(\mathbf{\Phi} - \gamma\bar{\mathbf{\Phi}}))^{-1}\mathbf{\Phi}^\top\mathbf{W}_k\mathbf{j} = \mathbf{A}_k^{-1}\mathbf{b}_k \tag{14}$$

where $\mathbf{A}_k \in \mathbb{R}^{\mathcal{M}\times\mathcal{M}}$ and $\mathbf{b}_k \in \mathbb{R}^{\mathcal{M}}$.

At this stage, it is possible to see a computational advantage of the proposed approach for fitting (potentially non-linear) $Q^{\boldsymbol{\pi}}$ in large state spaces. The computational complexity of solving (14) independently for each of the local models is $\mathcal{O}(\mathcal{M}^3)$, causing computation of the global fit (12) to have $\mathcal{O}(\mathcal{M}^3\mathcal{K})$ complexity. At first glance, this seems more expensive than the original version of LSPI. Note, however, here $\mathcal{M}$ is kept small through use of simple, generic local models. Increases to the model's representational power (e.g., if $Q^{\boldsymbol{\pi}}$ is strongly non-linear) are achieved not by increases to the number of basis functions $\mathcal{M}$, but by addition of local models $\mathcal{K}$, resulting in a linear increase in complexity[5].

[2]Note that, a unique solution to (9) is ensured provided that the basis functions (columns of $\mathbf{\Phi}$) are linearly independent, [13].

[3]In [13], the commands are assumed to be discrete such that solving (10) amounts to simply looking up the command with the minimum value. Here, the primary interest is in robotic applications with continuous commands.

[4]A similar strategy of using local weighting for weighted least squares fitting is employed in locally weighted regression (e.g., [9], [10]), but note that, here learning has no access to target data (i.e., samples of $Q^{\boldsymbol{\pi}}$).

[5]Note also that, since the learning of each local model is independent, this computation can easily be parallelised to yield further speed-up.

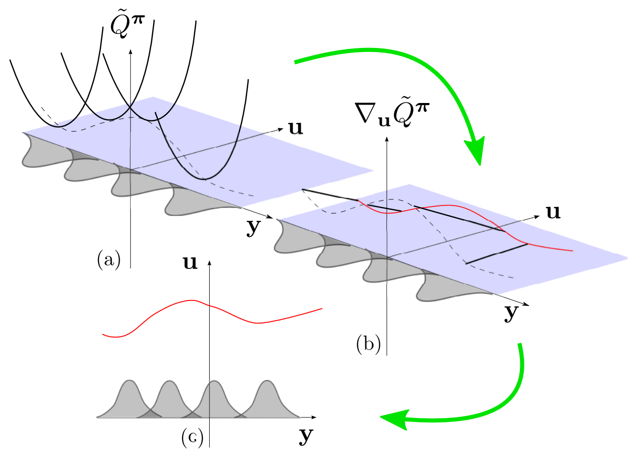


Fig. 2. Visualisation of locally weighted LSPI with the basis (15). (a) Estimate $\tilde{Q}^{\boldsymbol{\pi}}$ composed from a weighted combination of local models that are globally quadratic in $\mathbf{u}$. (b) Derivatives of the models with respect to $\mathbf{u}$ are linear in $\mathbf{u}$. (c) Solving for $\nabla_{\mathbf{u}}[\tilde{Q}^{\boldsymbol{\pi}}] = \mathbf{0}$ (intersection with the blue area) yields the (non-linear) policy (18).

2) Local Quadratic Models for Efficient Policy Improvement: Considering the selection of $\boldsymbol{\phi}(\mathbf{y},\mathbf{u})$ for the local models (11), in principle, any linearly independent set of features may be used. In this paper, the choice is guided by efficiency, at the cost of introducing bias in the approximation (12). Specifically, the local models use a quadratic basis of the form

$$\boldsymbol{\phi}(\mathbf{y},\mathbf{u}) = \mathrm{UT}[(\mathbf{y}^\top,\mathbf{u}^\top,1)^\top(\mathbf{y}^\top,\mathbf{u}^\top,1)] \quad \in \mathbb{R}^{\mathcal{M}} \qquad (15)$$

where $\mathrm{UT}[\cdot]$ is an operator that forms a column vector from the upper triangular part of its matrix argument. With this basis, the dimensionality of $\boldsymbol{\phi}(\mathbf{y},\mathbf{u})$ is $\mathcal{M} = 2(\mathcal{R}+\mathcal{Q}) + {}^{(\mathcal{R}+\mathcal{Q})}\mathrm{C}_2 + 1$. Using this basis, it is straightforward to retrieve the optimal policy from the estimate (12).

Taking derivatives with respect to $\mathbf{u}$, the minimum is the solution of

$$\boldsymbol{\nabla}_{\mathbf{u}}[\tilde{Q}^{\boldsymbol{\pi}}(\mathbf{y},\mathbf{u})] = \sum_{k}^{\mathcal{K}} \hat{w}_k(\mathbf{y})\boldsymbol{\nabla}_{\mathbf{u}}[\boldsymbol{\theta}_k^\top\boldsymbol{\phi}(\mathbf{y},\mathbf{u})] = \mathbf{0}. \qquad (16)$$

Since $\boldsymbol{\theta}_k^\top\boldsymbol{\phi}(\mathbf{y},\mathbf{u})$ is quadratic, its derivative is linear:

$$\boldsymbol{\nabla}_{\mathbf{u}}[\boldsymbol{\theta}_k^\top\boldsymbol{\phi}(\mathbf{y},\mathbf{u})] = \mathbf{L}_k^{\mathbf{y}}\mathbf{y} + \mathbf{L}_k^{\mathbf{u}}\mathbf{u} + \mathbf{l}_k^{\mathbf{1}} \qquad (17)$$

where $\mathbf{L}_k^{\mathbf{y}} \in \mathbb{R}^{\mathcal{Q}\times\mathcal{R}}$, $\mathbf{L}_k^{\mathbf{u}} \in \mathbb{R}^{\mathcal{Q}\times\mathcal{Q}}$ and $\mathbf{l}_k^{\mathbf{1}} \in \mathbb{R}^{\mathcal{Q}}$ are matrices with entries selected from $\boldsymbol{\theta}_k$ (see appendix). Substituting into (16) and rearranging, yields

$$\boldsymbol{\pi}^*(\mathbf{y}) = -\left(\textstyle\sum_k \hat{w}_k(\mathbf{y})\mathbf{L}_k^{\mathbf{u}}\right)^{-1} \textstyle\sum_k \hat{w}_k(\mathbf{y})(\mathbf{L}_k^{\mathbf{y}}\mathbf{y} + \mathbf{l}_k^{\mathbf{1}}). \qquad (18)$$

The policy $\boldsymbol{\pi}(\mathbf{y})$ is non-linear in $\mathbf{y}$ due to the weighed contributions of each of the local models: See Fig. 2.

An implication of the use of the quadratic basis (15), localised in $\mathbf{y}$ according to (12), is that the set of value functions $Q^{\boldsymbol{\pi}}$ that may be represented without error is restricted to those that are globally quadratic in $\mathbf{u}$. In general, this cannot be guaranteed. However, even with this approximation, reasonable performance can be obtained (as evidenced in the evaluations, see Sec. IV).

Algorithm 1 LWLSPI

```
Require: Data $\{\mathbf{y}_n,\mathbf{u}_n,\bar{\mathbf{y}}_n,j_n\}_{n=1}^{\mathcal{N}}$, discount factor $\gamma$,
    initial policy $\boldsymbol{\pi}_0$.
 1: Evaluate $\boldsymbol{\phi}_n = \boldsymbol{\phi}(\mathbf{y}_n,\mathbf{u}_n), n = 1,\cdots,\mathcal{N}$.
 2: Set $\boldsymbol{\pi} \leftarrow \boldsymbol{\pi}_0$.
 3: while $\boldsymbol{\theta}_k, k = 1,\cdots,\mathcal{K}$ changing do
 4:   Evaluate $\bar{\boldsymbol{\phi}}_n = \boldsymbol{\phi}(\bar{\mathbf{y}}_n,\boldsymbol{\pi}(\bar{\mathbf{y}}_n)), n = 1,\cdots,\mathcal{N}$.
 5:   for $k = 1,\cdots,\mathcal{K}$ do
 6:     Compute statistics:
          $\mathbf{A}_k = \sum_n^{\mathcal{N}} \hat{w}_{k,n}\boldsymbol{\phi}_n(\boldsymbol{\phi}_n - \gamma\bar{\boldsymbol{\phi}}_n)^\top$
          $\mathbf{b}_k = \sum_n^{\mathcal{N}} \hat{w}_{k,n}\boldsymbol{\phi}_n j_n$.
 7:     Retrieve parameters $\boldsymbol{\theta}_k = \mathbf{A}_k^{-1}\mathbf{b}_k$.
 8:   end for
 9:   Update policy
        $\boldsymbol{\pi}(\mathbf{y}) \leftarrow -(\sum_k \hat{w}_k(\mathbf{y})\mathbf{L}_k^{\mathbf{u}})^{-1}\sum_k \hat{w}_k(\mathbf{y})(\mathbf{L}_k^{\mathbf{y}}\mathbf{y} + \mathbf{l}_k^{\mathbf{1}})$.
10: end while
11: return Policy $\boldsymbol{\pi}(\mathbf{y})$ [optionally: $\tilde{Q}^{\boldsymbol{\pi}}(\mathbf{y},\mathbf{u})$]
```

Combining the policy evaluation steps (ref. Sec. III-B) and the policy improvement step (18), the complete algorithm, Locally Weighted LSPI (LWLSPI), is provided in Algorithm 1.

IV. Evaluation

In this section, the performance of LWLSPI is evaluated for learning in two non-linear control tasks, including the pendulum swing up problem [11] and a simulated robotic door opening task[6].

A. Pendulum Swing Up

The goal of the first evaluation, is to characterise learning performance in a simple, non-linear control problem. As an illustrative task, performance in the pendulum swing-up problem [11] is tested, in which the task is to control a pendulum, under the influence of gravity, to stabilise at the upright position (see Fig. 3(a)).

The dynamics of the pendulum are described by

$$ml^2\ddot{q} + \mu\dot{q} - mgl\sin q = \tau \qquad (19)$$

where $m = 1\,kg$ is the mass, $l = 1\,m$ is the length of the pendulum and $\mu = 0.01\,Nms/rad$ is the viscous friction coefficient. The system state (assumed fully observable, i.e., $\mathbf{y} = \mathbf{x}$) is described by the angular position and velocity $\mathbf{x} = (q,\dot{q})^\top$ and the control input is the torque around the pivot $u = \tau$, with limits $u \in [-5,5]\,Nm$. The latter cause the pendulum to be underpowered (i.e., unable to lift its own weight) making swing-up and stabilisation non-trivial.

Task performance is measured by the height reached by the pendulum bob $h(q) = l\cos(q)$ [11], with an additional penalty for high torque use

$$j = -h(q) + w_\tau\tau^2 \qquad (20)$$

where $w_\tau = 7.5\times10^{-4}$. Note that, (20) is not explicitly provided to the learner: Feedback about performance is provided indirectly through data samples j_n collected during

[6]Data used in these evaluations is freely available for download (DOI: 10.6084/m9.figshare.748806). Software may be requested from the corresponding author.

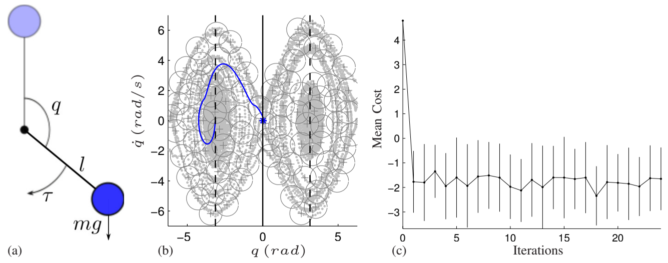


Fig. 3. Pendulum swing up problem. (a) Diagram of pendulum and physical parameters. (b) Phase space plot indicating training data (light grey dots), local model centres (+) with $\hat{w}_k = 0.1$ activation isolines (grey circles) and a sample test trajectory generated by the learnt policy from the downward-hanging (i.e., $q = -\pi\, rad$) position (blue line). (c) Average cost (mean±s.d. over 20 trials) versus number of iterations of LWLSPI on the 10 test trajectories (refer to main text).

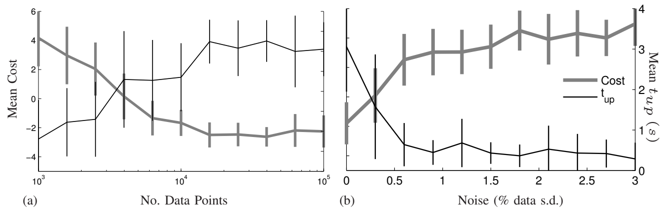


Fig. 4. Swing up task performance with respect to the average cost and balancing time t_{up} for the test trajectories (mean±s.d. over 20 trials) as (a) the number of data and (b) the noise increases.

exploration. In the setting where model information, such as knowledge of l, is unavailable, such data may be obtained, for example, by visual estimates of the bob height.

As training data, 8000 data points are collected in the form of trajectories recorded from the pendulum during motor babbling. In particular, each trajectory is initiated at rest, at a uniform-randomly selected joint angle, $q \sim \mathcal{U}[-2\pi, 2\pi]\, rad$, and a trajectory is generated by applying uniform-random commands $\tau \sim \mathcal{U}[-5, 5]\, Nm$ at each time step for $10\, s$ of operation. During each trajectory, the joint angular position, velocity, command and pendulum bob height are recorded at sampling rate $40\, Hz$.

Using this data, LWLSPI is then applied to learn a controller. The estimate $\tilde{Q}^{\boldsymbol{\pi}}$, is formed from a set of local models with Gaussian weighting functions, $w_k(\mathbf{y}) = e^{-|\mathbf{y}-\mathbf{c}_k|^2/2\sigma^2}$ with fixed width $\sigma^2 = 0.125$, and with the number and placement of the centres $\mathbf{c}_k$ chosen automatically according to the data. Briefly, the latter is done by iterating through the data, evaluating the activation of all existing local models for every datum, and placing a new model whenever a datum is found for which no model is activated above a minimum activation threshold $\hat{w}^{min} = 0.03$. In the evaluations reported here, this yielded 88 models, on average. A typical data set, and the associated local models, is illustrated in Fig. 3(b). The results reported here use discount factor $\gamma = 0.85$ and an initial policy that simply predicts zeros for all states, i.e., $\boldsymbol{\pi}_0(\mathbf{y}) = 0$.

Learning performance is evaluated by computing (i) the average cost incurred and (ii) the average time t_{up} that the pendulum spends upright ($|q| < \pi/4\, rad$) [11] for a set of 10 test trajectories. The latter are recorded for $10\, s$ of operation, starting from rest at a set of evenly-spaced joint angles $q \in [-\pi, \pi]\, rad$. In the following, results are reported for 20 trials repeated on different data sets.

Fig. 3(b) illustrates a sample test trajectory for an example policy learnt by LWLSPI. As can be seen, the movement starts with a phase of oscillation, in which the controller pumps energy into the system in order to overcome gravity. The pendulum is then stabilised, as the trajectory converges to the upright position ($q = 0\, rad$). Evaluated on the 10 test trajectories over 20 trials, the average cost incurred by the learnt policy is -1.79 ± 1.43 with a mean balancing time of $t_{up} = 3.06 \pm 1.11\, s$. In Fig. 3(c), the cost of the intermediate policies generated at each iteration of LWLSPI is plotted. As can be seen, after only one iteration, a near-optimal policy is reached, with only minor improvements from there on.

To test the robustness of learning, the evaluation is also repeated for data sets containing (i) different quantities of data and (ii) different levels of noise. For the latter, the state $\mathbf{y}_n$ and cost j_n measurements[7] are corrupted with zero-mean additive Gaussian noise with variance proportional to the scale of the data. The results are plotted in Fig. 4.

As can be seen, as the quantity of data increases the cost gradually decreases, and a corresponding increase, in the average balancing time is seen. Conversely, as the noise level increases the opposite trend can be observed, in keeping with expectations.

B. Door Opening

The focus of the next evaluation, is to test the robustness of LWLSPI in adaptively learning controllers for different systems with unknown dynamics, in the absence of direct state measurements. As an example of this, the task of learning to open unfamiliar doors, as discussed in Sec. II-A, is examined.

The doors are modelled as rigid bodies, attached to a rotational hinge, to which the robot may apply forces $\mathbf{u} = \mathbf{f}_r \in \mathbb{R}^2$ in Cartesian space at the handle (see Fig. 1). The state $\mathbf{x} = (q, \dot{q})^\top \in \mathbb{R}^2$ is described by the hinge angle and angular velocity, however, since the latter is assumed not to be directly observable (ref. Sec. II-A), observations are given in the form of end-effector position and velocity measurements $\mathbf{y} = (\mathbf{r}^\top, \dot{\mathbf{r}}^\top)^\top \in \mathbb{R}^4$. The latter may be provided from the robot kinematics, or visual estimates.

The 5 doors considered include:

1) *Pull-door:* opens *inwards*, with hinge angle restricted to admissible range $-\frac{2}{3}\pi < q < 0\, rad$.
2) *Push-door:* opens *outwards*, with hinge angle restricted to $0 < q < \frac{2}{3}\pi\, rad$.
3) *Swing-door:* opens in either direction, with hinge angle restricted to $-\frac{2}{3}\pi < q < \frac{2}{3}\pi\, rad$.
4) *Pull-door with closer:* identical to the *pull-door*, except for the presence of a 'closer' that applies a restorative force to bring the door back to the closed position. The closer is simulated as a spring-damper system that applies torque $\tau_c = -k_c q - b_c \dot{q}$ with $k_c = 1\, Nm/rad$ and $b_c = 0.5\, Nms/rad$.

[7]Note that, this matches real learning situations in which information about the state and cost come from noisy sensor data, while the u_n, are exactly known, since these are given from the controller.

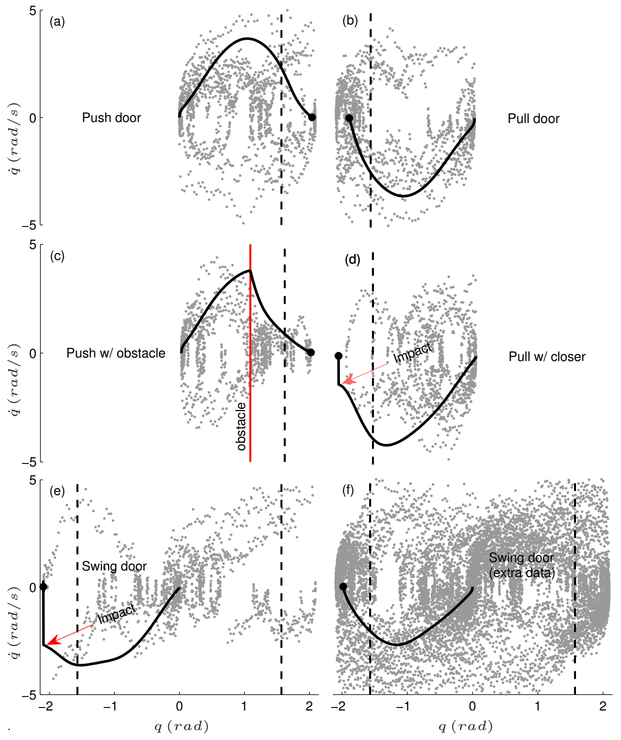


Fig. 5. Typical opening behaviours learnt for the different doors, visualised in the state space (phase space of the hinge) $\mathbf{x} = (q, \dot{q})^\top$. Solid black lines show the trajectories generated from the learnt controllers (final state indicated by ●). Grey dots indicate locations of training samples. Black dashed lines mark the angle for which the door is fully open ($q = \pm\pi/2$). In (c), the red line indicates the position of the obstacle.

5) *Push-door with obstacle:* identical to the push-door, except for the presence of an unseen object behind the door, causing it to 'jam' when hit. The obstacle is assumed to be placed such that jamming occurs at $q = \pi/3\,rad$, and 'jamming' is modelled as a 40-fold increase in the viscous friction after impact.

The doors have a common mass, $15\,kg$, width, $1\,m$, and viscous friction $0.1\,Nms/rad$. Any impacts between the door and its frame are modelled as inelastic collisions, with coefficient of restitution $c_r = 0.1$. Note that, (i) the differences in the doors are such that qualitatively different controllers (pushing, pulling, etc.) are required for successful opening, (ii) none of the parameters are explicitly available to the learner: The appropriate strategy must be determined from interactions with the system.

The primary performance measure in this task is to maximise the width of the gap $g(q)$ (see Fig. 1), in the shortest possible time. In addition, penalties are imposed for (i) excessively high forces and (ii) very high velocities, in order to reduce the risk of large internal forces or high velocity impacts that could potentially damage the door or robot. The resultant cost function is

$$j = -g(q) + w_v \dot{\mathbf{r}}^\top \dot{\mathbf{r}} + w_f \mathbf{f}_r^\top \mathbf{f}_r \qquad (21)$$

where $w_v = 5 \times 10^{-3}$ and $w_f = 10^{-4}$. Note that, the quantities in (21) are only provided through sensor measurements (g, $\dot{\mathbf{r}}$) or directly from the controller ($\mathbf{u} = \mathbf{f}_r$), as per the assumptions described in Sec. II-A.

Door	**Success** (%)	**Time** (s)	**Cost**
Pull door	100	1.04 ± 0.08	-2.13 ± 0.08
Push door	100	1.02 ± 0.11	-2.16 ± 0.11
Pull w/ closer	45	1.35 ± 0.30	-0.79 ± 0.92
Push w/ obstacle	75	1.10 ± 0.16	-1.58 ± 0.95
Swing door	75	1.20 ± 0.61	-1.95 ± 0.54
Swing door (extra data)	100	1.39 ± 0.29	-1.86 ± 0.25

TABLE I

SUCCESS RATE, OPENING TIME AND COST (MEAN±S.D.) FOR LEARNING TO OPEN DIFFERENT DOORS OVER 20 TRIALS.

As training data, 5 trajectories (3000 data points) are collected through motor babbling, with commands drawn uniform-randomly from $u_i \sim \mathcal{U}[-25, 25]\,N$, $i \in \{1, 2\}$ at every time step. In each trajectory, the door starts at rest, in the closed position, i.e., $\mathbf{x} = (0, 0)^\top$, and the robot end-effector position, velocity, force and gap width are recorded for $3\,s$. The model of Q^π is constructed in the same manner as described in Sec. IV-A, using Gaussian weighting functions, with fixed widths $\sigma^2 = 1.5$ and minimum activation threshold $\hat{w}^{min} = 0.3$ for allocating models. The discount factor is $\gamma = 0.9$ and the initial policy predicts zeros for all states, $\boldsymbol{\pi}_0(\mathbf{y}) = \mathbf{0}$.

Performance is evaluated on the (i) rate of success in opening the door (across trials), (ii) average opening time of successful trials, and (iii) cost incurred, when using the learnt policy to generate a trajectory from rest at the closed position. Here, 'success' is defined as the door reaching the fully opened position ($|q| \geq \pi/2\,rad$), and coming to rest ($|\dot{q}| < 0.05\,rad/s$) by the end of the trajectory. The results, summarised in Table I and Fig. 5, are for 20 repeated trials on different data sets.

Looking at the success rates (Table I), it can be seen that LWLSPI has no difficulty in learning to open the *pull-* or *push-* doors, and produces smooth trajectories (ref. Fig. 5) that incur a similar level of cost. For the *swing* door, the success rate is a little lower at 75%, with higher variance. A consequence is somewhat unstable behaviour, with, for instance, occasional high velocity impacts (see Fig. 5(e)). Upon closer examination, the cause appears to be sparsity of the data: the swing door has a larger state-space than the push or pull doors, and therefore requires greater exploration to gather sufficient data to learn a good policy. Repeating the experiment with with a 7-fold increase in the number of data confirms this: the success rate draws closer to that of the push/pull doors (ref. Table I), with lower variance in the cost.

The worst performance is seen in the case of the *pull door with closer* and the *push-door with obstacle*, with lower success rate and higher average cost. Again, this can be attributed to the data distribution: The effect of the closer on the motor-babbling exploration strategy, for instance, is a biasing of samples toward low-velocity movements around the closed position leaving little data around the target (open) position (the effect is similar for the obstacle), see Fig. 5(c)&(d)). It is speculated that a more directed exploration strategy (e.g., systematic testing of pushing and pulling behaviour with different levels of force) would improve the success rate for these cases too.

V. Conclusion

This study presents an extension to Least Squares Policy Iteration that enables learning of optimal feedback controllers for non-linear problems in which the dynamics, sensing and cost functions are unknown. By exploiting locally weighted learning techniques, the proposed approach allows for gains in computational efficiency compared to global methods, by focusing learning only on regions in which data has been seen. Numerical evaluations show the efficacy of the approach in solving non-linear control problems, characterise the data requirements (in terms of quantity and distribution), noise susceptibility, and show the robustness in learning controllers for systems with a range of different dynamics.

In future work, the adaptation of the size and shape of the weighting functions will be investigated in order to achieve greater accuracy and improve generalisation of the learnt controllers. It is also intended that the empirical evaluations be extended to include (i) more complex tasks in higher dimensions (e.g., by increasing the sensor space to provide richer information about the task), (ii) further investigation of suitable exploration strategies to enable fast learning in settings where the dynamics make random exploration unsuitable.

Appendix: Computation of $\mathbf{L}^{\mathbf{y}}$, $\mathbf{L}^{\mathbf{u}}$ and $\mathbf{l}^{\mathbf{1}}$

Define the symmetric matrices

$$\mathbf{\Theta}_k := \begin{pmatrix} (\boldsymbol{\theta}_k)_1 & \frac{1}{2}(\boldsymbol{\theta}_k)_2 & \cdots & \frac{1}{2}(\boldsymbol{\theta}_k)_{\mathcal{R}+\mathcal{Q}+1} \\ \cdot & (\boldsymbol{\theta}_k)_{\mathcal{R}+\mathcal{Q}+2} & \cdots & \frac{1}{2}(\boldsymbol{\theta}_k)_{2(\mathcal{R}+\mathcal{Q})+1} \\ \vdots & \vdots & \ddots & \vdots \\ \cdot & \cdot & \cdots & (\boldsymbol{\theta}_k)_{2(\mathcal{R}+\mathcal{Q})+{}^{(\mathcal{R}+\mathcal{Q})}\mathrm{C}_2+1} \end{pmatrix} \tag{22}$$

and

$$\mathbf{\Psi} := \begin{pmatrix} \mathbf{y} \\ \mathbf{u} \\ 1 \end{pmatrix} (\mathbf{y}^\top \mathbf{u}^\top 1) = \begin{pmatrix} \mathbf{y}\mathbf{y}^\top & \mathbf{y}\mathbf{u}^\top & \mathbf{y} \\ \mathbf{u}\mathbf{y}^\top & \mathbf{u}\mathbf{u}^\top & \mathbf{u} \\ \mathbf{y}^\top & \mathbf{u}^\top & 1 \end{pmatrix} \tag{23}$$

where $\mathbf{\Theta}_k, \mathbf{\Psi} \in \mathbb{R}^{(\mathcal{R}+\mathcal{Q}+1)\times(\mathcal{R}+\mathcal{Q}+1)}$. With these definitions, the local model of the value function (11) can be written as

$$\tilde{Q}_k^{\boldsymbol{\pi}}(\mathbf{y},\mathbf{u}) = \mathrm{vec}[\mathbf{\Theta}_k]^\top \mathrm{vec}[\mathbf{\Psi}] = \mathrm{Tr}[\mathbf{\Theta}_k^\top \mathbf{\Psi}]. \tag{24}$$

It is desired to solve

$$\boldsymbol{\nabla}_{\mathbf{u}}[\tilde{Q}_k^{\boldsymbol{\pi}}(\mathbf{y},\mathbf{u})] = \boldsymbol{\nabla}_{\mathbf{u}}[\boldsymbol{\theta}_k^\top \boldsymbol{\phi}(\mathbf{y},\mathbf{u})] = \mathbf{0}. \tag{25}$$

Note that

$$\frac{\partial \tilde{Q}_k^{\boldsymbol{\pi}}}{\partial u_i} = \mathrm{Tr}\left[\mathbf{\Theta}_k^\top \frac{\partial \mathbf{\Psi}}{\partial u_i}\right] \tag{26}$$

where

$$\mathbf{D} := \frac{\partial \mathbf{\Psi}}{\partial u_i} = \begin{pmatrix} \mathbf{0} & \mathbf{y}\mathbf{e}_i^\top & \mathbf{0} \\ \mathbf{e}_i\mathbf{y}^\top & \mathbf{e}_i\mathbf{u}^\top + \mathbf{u}\mathbf{e}_i^\top & \mathbf{e}_i \\ \mathbf{0} & \mathbf{e}_i^\top & 0 \end{pmatrix} \tag{27}$$

where $\mathbf{e}_i$ is the ith basis vector (i.e., a vector with 1 as the ith element and zeros elsewhere).

Define the block-wise partition[8] of $\mathbf{\Theta}$

[8]The k index is omitted in the following for readability.

$$\mathbf{\Theta} = \left(\begin{array}{c|c|c} \mathbf{\Theta}_{11}^{\mathcal{R}\times\mathcal{R}} & \mathbf{\Theta}_{12}^{\mathcal{R}\times\mathcal{Q}} & \mathbf{\Theta}_{13}^{\mathcal{R}\times 1} \\ \hline \mathbf{\Theta}_{21}^{\mathcal{Q}\times\mathcal{R}} & \mathbf{\Theta}_{22}^{\mathcal{Q}\times\mathcal{Q}} & \mathbf{\Theta}_{23}^{\mathcal{Q}\times 1} \\ \hline \mathbf{\Theta}_{31}^{1\times\mathcal{R}} & \mathbf{\Theta}_{32}^{1\times\mathcal{Q}} & \mathbf{\Theta}_{33}^{1\times 1} \end{array}\right) \tag{28}$$

and likewise for $\mathbf{D}$, where superscripts denote the block dimensions. Noting that the matrix product $\mathbf{C}=\mathbf{AB}$ can be formed blockwise according to $\mathbf{C}_{\alpha\beta} = \sum_\gamma \mathbf{A}_{\alpha\gamma}\mathbf{B}_{\gamma\beta}$, (26) can be re-written as

$$\frac{\partial \tilde{Q}^{\boldsymbol{\pi}}}{\partial u_i} = \mathrm{Tr}\left[\begin{pmatrix} \mathbf{\Theta}_{12}\mathbf{D}_{21} & \cdot & \cdot \\ \cdot & \sum_{j=1}^3 \mathbf{\Theta}_{2j}\mathbf{D}_{j2} & \cdot \\ \cdot & \cdot & \mathbf{\Theta}_{32}\mathbf{D}_{23} \end{pmatrix}\right] \tag{29}$$

where off-diagonal terms (irrelevant for computing the trace) are omitted. Substituting for $\mathbf{D}$ and manipulating, yields

$$\begin{aligned} \frac{\partial \tilde{Q}^{\boldsymbol{\pi}}}{\partial u_i} &= \mathrm{vec}[(\mathbf{\Theta}_{12}\mathbf{e}_i + \mathbf{\Theta}_{21}^\top\mathbf{e}_i)]^\top \mathbf{y} \\ &\quad + 2\mathrm{vec}[\mathbf{\Theta}_{22}^\top\mathbf{e}_i]^\top \mathbf{u} + \mathrm{vec}[(\mathbf{\Theta}_{23} + \mathbf{\Theta}_{32}^\top)]^\top \mathbf{e}_i \end{aligned} \tag{30}$$

The matrices $\mathbf{L}^{\mathbf{y}}$, $\mathbf{L}^{\mathbf{u}}$ and the vector $\mathbf{l}^{\mathbf{1}}$ are thus defined as

$$\mathbf{L}^{\mathbf{y}} = \begin{pmatrix} (\boldsymbol{\lambda}_1^{\mathbf{y}})^\top \\ \vdots \\ (\boldsymbol{\lambda}_{\mathcal{Q}}^{\mathbf{y}})^\top \end{pmatrix}, \quad \mathbf{L}^{\mathbf{u}} = \begin{pmatrix} (\boldsymbol{\lambda}_1^{\mathbf{u}})^\top \\ \vdots \\ (\boldsymbol{\lambda}_{\mathcal{Q}}^{\mathbf{u}})^\top \end{pmatrix}, \quad \mathbf{l}^{\mathbf{1}} = \begin{pmatrix} \lambda_1^{\mathbf{1}} \\ \vdots \\ \lambda_{\mathcal{Q}}^{\mathbf{1}} \end{pmatrix} \tag{31}$$

where $\boldsymbol{\lambda}_i^{\mathbf{y}} := \mathrm{vec}[(\mathbf{\Theta}_{12}\mathbf{e}_i + \mathbf{\Theta}_{21}^\top\mathbf{e}_i)] \in \mathbb{R}^{\mathcal{R}}$, $\boldsymbol{\lambda}_i^{\mathbf{u}} := 2\mathrm{vec}[\mathbf{\Theta}_{22}^\top\mathbf{e}_i] \in \mathbb{R}^{\mathcal{Q}}$ and $\lambda_i^{\mathbf{1}} := \mathrm{vec}[(\mathbf{\Theta}_{23} + \mathbf{\Theta}_{32}^\top)]^\top\mathbf{e}_i \in \mathbb{R}$.

References

[1] C. Atkeson and B. Stephens, "Random sampling of states in dynamic programming," *IEEE Trans. Sys., Man and Cybernetics*, vol. 38, no. 4, pp. 924 –929, 2008.

[2] D. Jacobson and D. Mayne, *Differential Dynamic Programming.* Elsevier, 1970.

[3] D. Mitrovic, S. Klanke, and S. Vijayakumar, "Adaptive optimal feedback control with learned internal dynamics models," in *From Motor Learning to Interaction Learning in Robots*. Springer, 2010.

[4] W. Li and E. Todorov, "Iterative linear-quadratic regulator design for nonlinear biological movement systems," in *Int. Conf. Informatics in Control, Automation & Robotics*, 2004.

[5] E. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning," *J. Machine Learning Research*, vol. 11, pp. 3137–3181, 2010.

[6] M. Howard and Y. Nakamura, "Locally weighted least squares temporal difference learning," in *European Symp. Artificial Neural Networks*, 2013.

[7] J. Boyan, "Technical update: Least-squares temporal difference learning," *Machine Learning*, vol. 49, pp. 233–246, 2002.

[8] S. Bradtke and A. Barto, "Linear least-squares algorithms for temporal difference learning," *Machine Learning*, vol. 22, pp. 33–57, 1996.

[9] S. Vijayakumar, A. D'Souza, and S. Schaal, "Incremental online learning in high dimensions," *Neural Computation*, vol. 17, no. 12, pp. 2602–2634, 2005.

[10] S. Schaal and C. Atkeson, "Constructive incremental learning from only local information," *Neural Computation*, vol. 10, pp. 2047–2084, 1998.

[11] K. Doya, "Reinforcement learning in continuous time and space," *Neural Computation*, vol. 21, pp. 219–245, 2000.

[12] S. Whitehead and D. Ballard, "Learning to perceive and act by trial and error," *Machine Learning*, vol. 7, pp. 45–83, 1991.

[13] M. Lagoudakis and R. Parr, "Least-squares policy iteration," *J. Machine Learning Research*, vol. 4, pp. 1107–1149, 2003.

[14] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction.* MIT Press, 1998.

[15] A. Lazaric, M. Ghavamzadeh, and R. Munos, "Finite-sample analysis of LSTD," in *Int. Conf. Machine Learning*, 2010.

[16] W. Cleveland and S. Devlin, "Locally weighted regression: An approach to regression analysis by local fitting," *J. American Stat. Assoc.*, vol. 83, no. 403, pp. 596–610, 1988.