In: Proc. of the 10th International Conference on Simulation of Adaptive Behaviour (SAB '08), Osaka, Japan (2008)

# Adaptive Optimal Control for Redundantly Actuated Arms

Djordje Mitrovic, Stefan Klanke, and Sethu Vijayakumar

Institute of Perception, Action & Behavior, University of Edinburgh,
The King's Buildings, Edinburgh EH9 3JZ, United Kingdom
{d.mitrovic,s.klanke,sethu.vijayakumar}@ed.ac.uk

**Abstract.** Optimal feedback control has been proposed as an attractive movement generation strategy in goal reaching tasks for anthropomorphic manipulator systems. Recent developments, such as the iterative Linear Quadratic Gaussian (iLQG) algorithm, have focused on the case of non-linear, but still analytically available, dynamics. For realistic control systems, however, the dynamics may often be unknown, difficult to estimate, or subject to frequent systematic changes. In this paper, we combine the iLQG framework with learning the forward dynamics for a simulated arm with two limbs and six antagonistic muscles, and we demonstrate how our approach can compensate for complex dynamic perturbations in an online fashion.

**Key words:** Adaptive optimal control, learning dynamics, redundant actuation

## 1 Introduction

In this work, we focus on the issues related to planning and control of reaching movements for anthropomorphic manipulators with redundant actuation based on antagonistic muscles. While such systems are becoming more and more popular especially where compliance and interaction with humans is required, controlling these systems remains a big challenge: Apart from the problem of often highly non-linear and hard to model system dynamics, the controller has to make a choice between many different possible trajectories (kinematics) and a multitude of applicable motor commands (dynamics) for achieving a particular task. How do we resolve this redundancy?

Optimal control theory [1] answers this question by establishing a certain cost function, and selecting the solution with minimal cost (e.g., minimum jerk [2]). Quite often these control schemes are only concerned with trajectory *planning* and an "open loop" optimisation of the control commands, while the correction of errors during *execution* is left to simple PID controllers.

As an alternative, closed loop optimisation models are aimed at providing a control law which is explicitly based on feedback from the system. In the ideal case, the system state is directly mapped to control signals during execution,

and the form of this mapping is again governed by a cost function. A key property of such optimal feedback controllers (OFC) is that errors are only corrected if they adversely affect the task performance (minimum intervention principle [3]). This is important especially in systems that suffer from control dependent noise, since task-irrelevant correction could destabilise the system beside expending additional control effort. Empirically, OFC also accounts for many motion patterns that have been observed in natural, redundant systems and human experiments [4] including the confounding trial-to-trial variability in individual degrees of freedom that, remarkably, manages to not compromise task optimality [5, 6]. Therefore, this paradigm is potentially a very attractive control strategy for artificial anthropomorphic systems (i.e., many degrees of freedom, redundant actuation, flexible lightweight construction, variable stiffness).

Unfortunately, finding a globally valid optimal control law is a very hard problem especially for non-linear and high-dimensional systems. We therefore resort to hybrid algorithms that present a compromise between open loop and closed loop optimisation, that is, algorithms which iteratively compute an optimal trajectory together with a locally valid feedback law. Examples of these are differential dynamic programming (DDP) [7, 8], iterative linear-quadratic regulator designs [9], or the recent iterative Linear Quadratic Gaussian (iLQG) framework [10], which will form the basis of our work.

A major shortcoming of iLQG (and DDP) is the dependence on an analytic form of the system dynamics, which often may be unknown or subject to change. We overcome this limitation by learning an adaptive internal model of the system dynamics using an online, supervised learning method. We consequently use the learned model to derive an iLQG formulation that is computationally efficient, reacts optimally to transient perturbations, and most notably adapts to systematic changes in the plant dynamics.

The idea of learning the system dynamics in combination with iterative optimisations of trajectory or policy has been explored previously in the literature, e.g., for learning to swing up a pendulum [11] using some prior knowledge about the form of the dynamics. Similarly, Abeel et al. [12] proposed a hybrid reinforcement learning algorithm, where a policy and an internal model get subsequently updated from "real life" trials. In contrast to their method, however, we employ a second-order optimisation method, and we refine the control law solely from the internal model. To our knowledge, learning dynamics in conjunction with control optimisation has not been studied in the light of adaptability to changing plant dynamics. In this paper, we successfully apply our adaptive control formalism to a movement system with six antagonistic muscles, which exhibits large redundancies and complex non-linearities of the dynamics.

## 2 A Simulation Model of Redundant Actuation

We wish to study a two degrees of freedom (DoF) planar human arm model, which is actuated by four single-joint and two double-joint antagonistic muscles (Fig. 1, left). The arm model described in this section is based on [13]. Although

kinematically simple, the system is over-actuated and therefore an interesting testbed for our control scheme, because large redundancies in the dynamics have to be resolved. The dimensionality of the control signals makes adaptation processes (e.g., to external force fields) quite demanding.

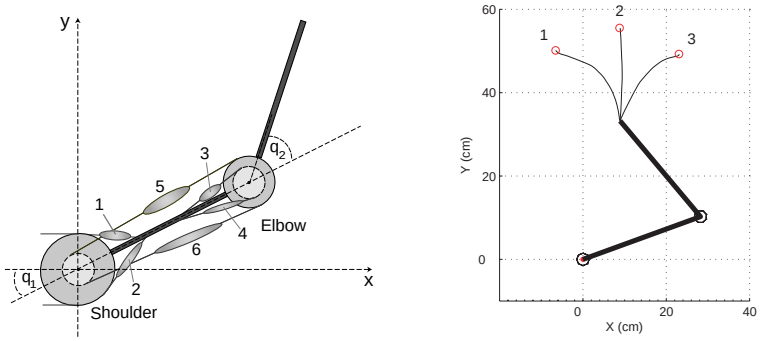


**Fig. 1.** Left: Human arm model with 6 muscles (adapted from [13]). Right: Same arm model with three selected targets (circles) and iLQG generated trajectories as benchmark data. The physics of the model is simulated using the Matlab Robotics Toolbox [14].

The dynamics of the arm is in part based on standard equations of motion. For our planar 2-DoF manipulator the joint torques $\boldsymbol{\tau}$ are given by

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}, \tag{1}$$

where $\mathbf{q}$ and $\dot{\mathbf{q}}$ are the joint angles and velocities, respectively; $\mathbf{M}(\mathbf{q})$ is the two-dimensional symmetric joint space inertia matrix and $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ accounts for Coriolis and centripetal forces.

Given the antagonistic muscle-based actuation, we can not command joint torques directly, but rather we have to calculate effective torques from the muscle activations $\mathbf{u}$. For the present model the corresponding transfer function is given by

$$\boldsymbol{\tau}(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{u}) = -\mathbf{A}(\mathbf{q})^T \mathbf{T}(\mathbf{l}, \dot{\mathbf{l}}, \mathbf{u}), \tag{2}$$

where $\mathbf{A}$ represents the moment arm. For simplicity, we assume $\mathbf{A}$ to be constant and independent of the joint angles $\mathbf{q}$:

$$\mathbf{A}(\mathbf{q}) = \mathbf{A} = \begin{pmatrix} a_1 & a_2 & 0 & 0 & a_5 & a_6 \\ 0 & 0 & a_3 & a_4 & a_7 & a_8 \end{pmatrix}^T. \tag{3}$$

The muscle lengths $\mathbf{l}$ depend on the joint angles $\mathbf{q}$ through the affine relationship $\mathbf{l} = \mathbf{l}_m - \mathbf{A}\mathbf{q}$, which also implies $\dot{\mathbf{l}} = -\mathbf{A}\dot{\mathbf{q}}$. The term $\mathbf{T}(\mathbf{l}, \dot{\mathbf{l}}, \mathbf{u})$ in (2) denotes the muscle tension, for which we follow the Kelvin-Voight model [15] and define:

$$\mathbf{T}(\mathbf{l}, \dot{\mathbf{l}}, \mathbf{u}) = \mathbf{K}(\mathbf{u})\big(\mathbf{l}_r(\mathbf{u}) - \mathbf{l}\big) - \mathbf{B}(\mathbf{u})\dot{\mathbf{l}}. \tag{4}$$

Here, $\mathbf{K}(\mathbf{u})$, $\mathbf{B}(\mathbf{u})$, and $\mathbf{l}_r(\mathbf{u})$ denote the muscle stiffness, the muscle viscosity and the muscle rest length, respectively. Each of these terms depends linearly on the motor commands $\mathbf{u}$, as given by

$$\mathbf{K}(\mathbf{u}) = diag(\mathbf{k}_0 + k\mathbf{u}), \qquad \mathbf{B}(\mathbf{u}) = diag(\mathbf{b}_0 + b\mathbf{u}), \qquad \mathbf{l}_r(\mathbf{u}) = \mathbf{l}_0 + r\mathbf{u}. \quad (5)$$

The elasticity coefficient $k$, the viscosity coefficient $b$, and the constant $r$ are given from the muscle model. The same holds true for $\mathbf{k}_0$, $\mathbf{b}_0$, and $\mathbf{l}_0$, which are the intrinsic elasticity, viscosity and rest length for $\mathbf{u} = \mathbf{0}$, respectively. For the exact values of these coefficients please refer to [13].

Please note that in contrast to standard torque-controlled robots, here the dynamics (1) is *not* linear in the control signals, since $\mathbf{u}$ enters (4) quadratically.

## 3 Locally-Optimal Feedback Control

Let $\mathbf{x}(t)$ denote the state of a plant and $\mathbf{u}(t)$ the applied control signal at time $t$. In this paper, the state consists of the joint angles $\mathbf{q}$ and velocities $\dot{\mathbf{q}}$ of the arm, and the control signals $\mathbf{u}$ are the muscle activations. If the system would be deterministic, we could express its dynamics as $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$, whereas in the presence of noise we write the dynamics as a stochastic differential equation

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, \mathbf{u})dt + \mathbf{F}(\mathbf{x}, \mathbf{u})d\boldsymbol{\omega}. \quad (6)$$

Here, $d\boldsymbol{\omega}$ is assumed to be Brownian motion noise, which is transformed by a possibly state- and control-dependent matrix $\mathbf{F}(\mathbf{x}, \mathbf{u})$. We state our problem as follows: Given an initial state $\mathbf{x}_0$ at time $t = 0$, we seek a control sequence $\mathbf{u}(t)$ such that the system's state is $\mathbf{x}^*$ at time $t = T$. Stochastic optimal control theory approaches the problem by first specifying a cost function which is composed of (i) some evaluation $h(\mathbf{x}(T))$ of the final state, usually penalising deviations from the desired state $\mathbf{x}^*$, and (ii) the accumulated cost $c(t, \mathbf{x}, \mathbf{u})$ of sending a control signal $\mathbf{u}$ at time $t$ in state $\mathbf{x}$, typically penalising large motor commands. Introducing a policy $\boldsymbol{\pi}(t, \mathbf{x})$ for selecting $\mathbf{u}(t)$, we can write the *expected cost* of following that policy from time $t$ as [10]

$$v^{\boldsymbol{\pi}}(t, \mathbf{x}(t)) = \left\langle h(\mathbf{x}(T)) + \int_t^T c(s, \mathbf{x}(s), \boldsymbol{\pi}(s, \mathbf{x}(s)))ds \right\rangle. \quad (7)$$

One then aims to find the policy $\boldsymbol{\pi}$ that minimises the total expected cost $v^{\boldsymbol{\pi}}(0, \mathbf{x}_0)$. Thus, in contrast to classical control, calculation of the trajectory (planning) and the control signal (execution) is not separated anymore, and for example, redundancy can actually be exploited in order to decrease the cost. If the dynamics $\mathbf{f}$ is linear in $\mathbf{x}$ and $\mathbf{u}$, the cost is quadratic, and the noise is Gaussian, the resulting so-called LQG problem is convex and can be solved analytically [1].

In our case of non-linear dynamics, global solutions can in theory still be found by applying dynamic programming methods [16] based on the Hamilton-Jacobi-Bellman equations. However, in their basic form these methods rely on a

discretisation of the state and action space, an approach that is not viable for large DoF systems. Some research has been carried out on random sampling in a continuous state and action space [17], and it has been suggested that sampling can avoid the curse of dimensionality if the underlying problem is simple enough [18], as is the case if the dynamics and cost functions are very smooth.

As an alternative, one can compute linear and quadratic approximations to the dynamics and the cost, respectively, and iteratively solve a "local" LQG problem to improve the control solution, until at least a local minimum of the cost function is found. The resulting iLQG algorithm has only recently been introduced [10], so we give a brief summary in the following[1].

One starts with an initial time-discretised control sequence $\bar{\mathbf{u}}_k \equiv \bar{\mathbf{u}}(k\Delta t)$ and applies the deterministic forward dynamics to retrieve an initial trajectory $\bar{\mathbf{x}}_k$, where

$$\bar{\mathbf{x}}_{k+1} = \bar{\mathbf{x}}_k + \Delta t\, \mathbf{f}(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k). \tag{8}$$

Linearising the discretised dynamics (6) around $\bar{\mathbf{x}}_k$ and $\bar{\mathbf{u}}_k$ and subtracting (8), one gets a dynamics equation for the deviations $\delta\mathbf{x}_k = \mathbf{x}_k - \bar{\mathbf{x}}_k$ and $\delta\mathbf{u}_k = \mathbf{u}_k - \bar{\mathbf{u}}_k$:

$$\delta\mathbf{x}_{k+1} = \left(\mathbf{I} + \Delta t \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\Big|_{\bar{\mathbf{x}}_k}\right)\delta\mathbf{x}_k + \Delta t \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\Big|_{\bar{\mathbf{u}}_k}\delta\mathbf{u}_k + \sqrt{\Delta t}\left(\mathbf{F}(\mathbf{u}_k) + \frac{\partial \mathbf{F}}{\partial \mathbf{u}}\Big|_{\bar{\mathbf{u}}_k}\delta\mathbf{u}_k\right)\boldsymbol{\xi}_k. \tag{9}$$

Similarly, one can derive an approximate cost function which is quadratic in $\delta\mathbf{u}$ and $\delta\mathbf{x}$. Thus, in the vicinity of the current trajectory $\bar{\mathbf{x}}$, the two approximations form a "local" LQG problem, which can be solved analytically and yields an affine control law $\delta\mathbf{u}_k = \mathbf{l}_k + \mathbf{L}_k\delta\mathbf{x}_k$ (for details please see [10]). This control law is fed into the linearised dynamics (eq. 9 without the noise term) and the resulting $\delta\mathbf{x}$ are used to update the trajectory $\bar{\mathbf{x}}$. In the same way, the control sequence $\bar{\mathbf{u}}$ is updated from $\delta\mathbf{u}$. This process is repeated until the total cost cannot be reduced anymore. The resultant control sequence $\bar{\mathbf{u}}$ can then be applied to the system, whereas the matrices $\mathbf{L}_k$ from the final iteration may serve as feedback gains.

In our current implementation we do not utilise an explicit noise model $\mathbf{F}$ for the sake of clarity of results; in any case, a matching feedback control law is only marginally superior to one that is optimised for a deterministic system [10].

## 4 iLQG with Learned Dynamics (iLQG–LD)

In order to eliminate the need for an analytic dynamics model and to make iLQG adaptive, we wish to learn an approximation $\tilde{\mathbf{f}}$ of the real plant forward dynamics $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$. Assuming our model $\tilde{\mathbf{f}}$ has been coarsely pre-trained, for example by motor babbling, we can refine that model in an online fashion as shown in Fig. 2.

For optimising and carrying out a movement, we have to define a cost function (where also the desired final state is encoded), the start state, and the number

[1] DDP works similarly, but requires quadratic approximations of both the dynamics and the cost function.

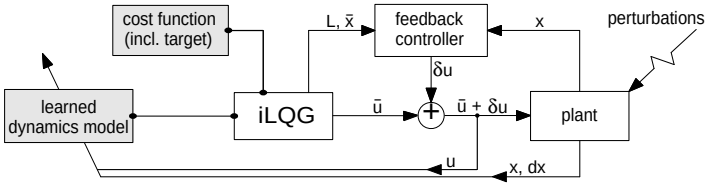


**Fig. 2.** Illustration of our iLQG–LD learning and control scheme.

of discrete time steps. Given an initial control sequence $\bar{\mathbf{u}}^0$, the iLQG iterations can be carried out as described in the previous section, but utilising the learned model $\tilde{\mathbf{f}}$. This yields a locally optimal control sequence $\bar{\mathbf{u}}_k$, a corresponding desired state sequence $\bar{\mathbf{x}}_k$, and feedback correction gain matrices $\mathbf{L}_k$. Denoting the plant's true state by $\mathbf{x}$, at each time step $k$, the feedback controller calculates the required correction to the control signal as $\delta\mathbf{u}_k = \mathbf{L}_k(\mathbf{x}_k - \bar{\mathbf{x}}_k)$. We then use the final control signal $\mathbf{u}_k = \bar{\mathbf{u}}_k + \delta\mathbf{u}_k$, the plant's state $\mathbf{x}_k$ and its change $d\mathbf{x}_k$ to update our internal forward model $\tilde{\mathbf{f}}$. As we show in Section 5, we can thus account for (systematic) perturbations and also bootstrap a dynamics model from scratch.

The domain of real-time control demands certain properties of a learning algorithm, namely fast learning rates, high prediction speeds at run-time, and robustness towards negative interference if the model is trained incrementally. Locally Weighted Projection Regression (LWPR) has been shown to exhibit these properties, and to be very efficient for incremental learning of non-linear models in high dimensions [19]. In LWPR, the regression function is constructed by blending local linear models, each of which is endowed with a locality kernel that defines the area of its validity (also termed its receptive field). During training, the parameters of the local models (locality and fit) are updated using incremental Partial Least Squares, and models can be pruned or added on an as-need basis, for example, when training data is generated in previously unexplored regions.

LWPR learning has the desirable property that it can be carried out online, and moreover, the learned model can be adapted to changes in the dynamics in real-time. A forgetting factor $\lambda$ [19], which balances the trade-off between preserving what has been learned and quickly adapting to the non-stationarity, can be tuned to the expected rate of external changes.

## 5 Experiments

We study movements of our arm model (Section 2) for a fixed motion duration of one second, which we discretise into $K = 50$ steps ($\Delta t = 0.02$s). The manipulator starts at an initial position $\mathbf{q}_0$ and reaches towards a target $\mathbf{q}_{tar}$. During movement we wish to minimise the amount of muscle activation ($\approx$ energy con-

sumption) of the system. We therefore use the cost function

$$v = w_p \,|\, \mathbf{q}_K - \mathbf{q}_{tar} \,|^2 + w_v \,|\, \dot{\mathbf{q}}_K \,|^2 + w_e \sum_{k=0}^{K} |\, \mathbf{u}_k \,|^2 \Delta t, \tag{10}$$

where the factors for the target position accuracy ($w_p$), the final target velocity accuracy ($w_v$), and for the energy term ($w_e$) weight the importance of each component.

### 5.1 Stationary Dynamics

In order to make iLQG–LD work for our three reference targets (see Fig. 1, right) we coarsely pre-trained our LWPR model with a focus on a wide coverage of the workspace. For the arm model we use in this paper, the training data are given as tuples consisting of $(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{u})$ as inputs (10 dimensions in total), and the observed joint accelerations $\ddot{\mathbf{q}}$ as the desired two-dimensional output. We stopped training once the normalised mean squared error (nMSE) in the predictions reached $\leq 0.005$. At this point LWPR had seen $1.2 \cdot 10^6$ training data points and had acquired 852 receptive fields, which is in accordance with the previously discussed high non-linearity of the plant dynamics.

We carried out a reaching task to the three reference targets using the feedback controller (feedback gain matrix $\mathbf{L}$) that falls out of iLQG(-LD). To compare the stability of the control solution, we simulated control dependent noise by contaminating the muscle commands $\mathbf{u}$ just before feeding them into the plant. We applied Gaussian noise with 50% of the variance of the signal $\mathbf{u}$.

Figure 3 depicts the generated control signals and the resulting performance of iLQG–LD and iLQG over 20 reaching trials per target. Both methods show similar endpoint variances and trajectories which are in close match. As can be seen from the visualisation of the control sequences, antagonistic muscles (i.e., muscle pairs 1/2, 3/4, and 5/6) are never activated at the same time. This is a direct consequence of the cost function, which penalises co-contraction as a waste of energy. Table 1 quantifies the control results of iLQG–LD and iLQG for each target with respect to the number of iterations, the generated running costs and the end point accuracy.

**Table 1.** Comparison of the performance of iLQG–LD and iLQG with respect to the number of iterations required to compute the control law, the average running cost, and the average Euclidean distance to the three reference targets (left, center, right).

| | **iLQG** | | | **iLQG–LD** | | |
|---|---|---|---|---|---|---|
| **Targets** | **Iter.** | **Run. cost** | **d (cm)** | **Iter.** | **Run. cost** | **d (cm)** |
| Center | 19 | **0.0345** $\pm$ 0.0060 | **0.11** $\pm$ 0.07 | 14 | **0.0427** $\pm$ 0.0069 | **0.38** $\pm$ 0.22 |
| Left | 40 | **0.1873** $\pm$ 0.0204 | **0.10** $\pm$ 0.06 | 36 | **0.1670** $\pm$ 0.0136 | **0.21** $\pm$ 0.16 |
| Right | 41 | **0.1858** $\pm$ 0.0202 | **0.57** $\pm$ 0.49 | 36 | **0.1534** $\pm$ 0.0273 | **0.19** $\pm$ 0.12 |

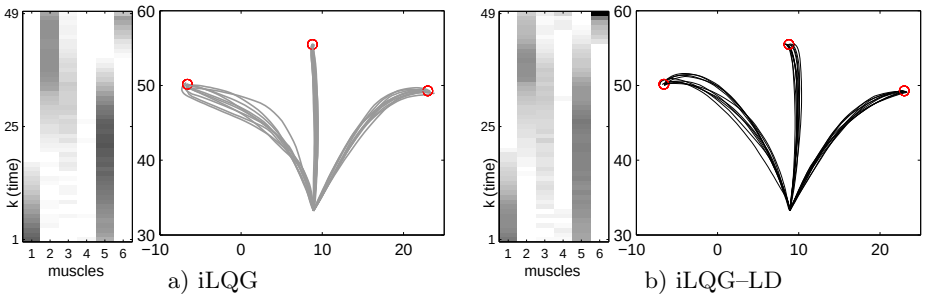


**Fig. 3.** Illustration of an optimised control sequence (left) and resulting trajectories (right) when using a) the known analytic dynamics model and b) the LWPR model learned from data. The control sequences (left target only) for each muscle (1–6) are drawn from bottom to top, with darker grey levels indicating stronger muscle activation.

### 5.2 Adaptation Results

A major advantage of iLQG–LD is that it does not rely on an accurate analytic dynamics model; consequently, it can adapt 'on-the-fly' to external perturbations and to changes in the plant dynamics that may result from altered morphology or wear and tear. We carried out adaptive reaching experiments (towards the center target) in our simulation similar to the human manipulandum experiments in [20]. We generated a constant unidirectional force field (FF) acting perpendicular to the reaching movement (see Fig. 4). Using the iLQG–LD model from the previous experiment, the manipulator gets strongly deflected when reaching for the target because the learned dynamics model cannot yet account for the "spurious" forces. However, using the resultant deflected trajectory as training data, updating the dynamics model online brings the manipulator nearer to the target with each new trial. In order to produce enough training data, as is required for a successful adaptation, we generated 20 slightly jittered versions of the optimised control sequences, each with length $K = 50$. We then ran those 20 trajectories on the plant, and trained the LWPR model with a total of $K \times 20 = 1000$ samples. We repeated this procedure until the iLQG–LD performance converged successfully, which was the case after 27000 training samples. At that point, the internal model successfully accounted for the change in dynamics caused by the FF. Then, we switched off the FF while continuing to use the adapted LWPR model. This resulted in an overshooting of the manipulator to the other side, trying to compensate for non-existing forces. Just as before, we re-adapted the dynamics online over repeated trials. The arm reached the target again after 7000 training points.

For accelerating the adaptation process, we set LWPR's forgetting factor to $\lambda = 0.95$ (instead of the default 0.999), which allows the learner to weight the importance of new data more strongly [19]. It is interesting to note that since the iLQG–LD control scheme always tries to correct the system towards

the target, it produces relevant dynamics training data in a way that could be termed "active learning".

Figure 4 summarises the results of the sequential adaptation process just described. Please note how the optimised "adapted" control sequence contains considerably stronger activations of the extensor muscles responsible for pulling the arm to the right (denoted by "2" and "6" in Fig. 1, left), while still exhibiting practically no co-contraction.

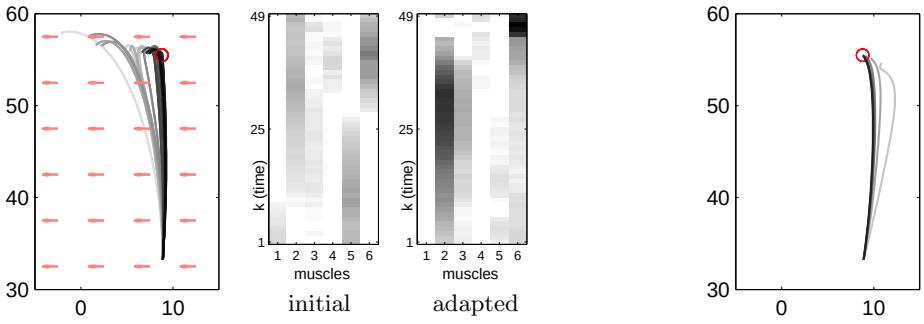


**Fig. 4.** Left: Adaptation to a unidirectional constant force field (indicated by the arrows). Darker lines indicate better trained models. In particular, the left-most trajectory corresponds to the "initial" control sequence, which was calculated using the LWPR model *before* the adaptation process. The fully "adapted" control sequence results in a nearly straight line reaching movement. Right: Resulting trajectories during re-adaptation after the force field has been switched off.

## 6 Conclusion

In this work we introduced iLQG–LD, a method that realises adaptive optimal feedback control by incorporating a learned dynamics model into the iLQG framework. Most importantly, we carried over the favourable properties of iLQG to more realistic control problems where the analytic dynamics model is often unknown, difficult to estimate accurately or subject to changes. As with iLQG control, redundancies are implicitly resolved by the OFC framework through a cost function, eliminating the need for a separate trajectory planner and inverse kinematics/dynamics computation.

Using a non-linear arm model actuated by six antagonistic muscles, we empirically showed that iLQG–LD performs reliably in the presence of noise and that it is adaptive with respect to systematic changes in the dynamics; hence, the framework has the potential to provide a unifying tool for modelling (and informing) non-linear sensorimotor adaptation experiments even under complex dynamic perturbations.

**Acknowledgements.** This work has been carried out within the SENSOPAC project which is supported by the European Commission through the Sixth Framework Programme for Research and Development.

## References


1. Stengel, R.F.: Optimal control and estimation. Dover Publications, New York (1994)
2. Flash, T., Hogan, N.: The coordination of arm movements: an experimentally confirmed mathematical model. Journal of Neuroscience **5** (1985) 1688–1703
3. Todorov, E., Jordan, M.: A minimal intervention principle for coordinated movement. In: Advances in Neural Information Processing Systems. Volume 15., MIT Press (2003) 27–34
4. Shadmehr, R., Wise, S.P.: The Computational Neurobiology of Reaching and Ponting. MIT Press (2005)
5. Li, W.: Optimal Control for Biological Movement Systems. PhD dissertation, University of California, San Diego (2006)
6. Scott, S.H.: Optimal feedback control and the neural basis of volitional motor control. Nature Reviews Neuroscience **5** (2004) 532–546
7. Dyer, P., McReynolds, S.: The Computational Theory of Optimal Control. Academic Press, New York (1970)
8. Jacobson, D.H., Mayne, D.Q.: Differential Dynamic Programming. Elsevier, New York (1970)
9. Li, W., Todorov, E.: Iterative linear-quadratic regulator design for nonlinear biological movement systems. In: Proc. 1st Int. Conf. Informatics in Control, Automation and Robotics. (2004)
10. Todorov, E., Li, W.: A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. In: Proc. of the American Control Conference. (2005)
11. Atkeson, C.G., Schaal, S.: Learning tasks from a single demonstration. In: Proc. Int. Conf. on Robotics and Automation (ICRA). Volume 2., Albuquerque, New Mexico (1997) 1706–1712
12. Abbeel, P., Quigley, M., Ng, A.Y.: Using inaccurate models in reinforcement learning. In: Proc. Int. Conf. on Machine Learning. (2006) 1–8
13. Katayama, M., Kawato, M.: Virtual trajectory and stiffness ellipse during multijoint arm movement predicted by neural inverse model. Biol. Cybern. **69** (1993) 353–362
14. Corke, P.I.: A robotics toolbox for MATLAB. IEEE Robotics and Automation Magazine **3**(1) (1996) 24–32
15. Özkaya, N., Nordin, M.: Fundamentals of biomechanics: equilibrium, motion, and deformation. Van Nostrand Reinhold, New York (1991)
16. Bertsekas, D.P.: Dynamic programming and optimal control. Athena Scientific, Belmont, Mass. (1995)
17. Thrun, S.: Monte carlo POMDPs. In: Advances in Neural Information Processing Systems 12, MIT Press (2000) 1064–1070
18. Atkeson, C.G.: Randomly sampling actions in dynamic programming. In: Proc. Int. Symp. on Approximate Dynamic Programming and Reinforcement Learning. (2007) 185–192
19. Vijayakumar, S., D'Souza, A., Schaal, S.: Incremental online learning in high dimensions. Neural Computation **17** (2005) 2602–2634
20. Shadmehr, R., Mussa-Ivaldi, F.A.: Adaptive representation of dynamics during learning of a motor task. The Journal of Neurosciene **14**(5) (1994) 3208–3224