Reinforcement Learning of Impedance Control in Stochastic Force Fields

Freek Stulp*, Jonas Buchli*†, Alice Ellmer*, Michael Mistry‡, Evangelos Theodorou* and Stefan Schaal*
*Computational Learning and Motor Control Lab, University of Southern California, Los Angeles, CA 90089, USA
†Dept. of Advanced Robotics, Italian Institute of Technology, Via Morego 30, 16163 Genova, Italy
‡Disney Research, Pittsburgh, PA 15213, USA

***Abstract*—Variable impedance control is essential for ensuring robust and safe physical interaction with the environment. As demonstrated in numerous force field experiments, humans combine two strategies to adapt their impedance to external perturbations: 1) if perturbations are unpredictable, subjects increase their impedance through co-contraction; 2) if perturbations are predictable, subjects learn a feed-forward command to counter the known perturbation.**

In this paper, we apply the force field paradigm to a simulated 7-DOF robot, by exerting stochastic forces on the robot's end-effector. The robot 'subject' uses our model-free reinforcement learning algorithm PI^2 to simultaneously learn the end-effector trajectories and variable impedance schedules. We demonstrate how the robot learns the same two-fold strategy to perturbation rejection as humans do, resulting in qualitatively similar behavior. Our results provide a biologically plausible approach to learning appropriate impedances purely from experience, without requiring a model of either body or environment dynamics.

I. Introduction

To ensure robust and safe physical interaction with the environment, humans adapt the impedance of their biomechanical system to different task requirements and stochastic disturbances. Numerous force field experiments have demonstrated that humans combine two strategies to deal with perturbations [15], [14], [2], [11], [3], [10]. Deterministic, and thus predictable, perturbations are countered by learning a feed-forward term, whereas stochastic perturbations lead to an increase in stiffness through muscle co-contraction, as illustrated in Fig. 1 and 2. The general rule of thumb thus seems to be *"be compliant when possible; stiffen up only when the task requires it"*. Task-specific adaptation of impedance allows humans to combine the advantages of high stiffness (accurate tracking, stable under unforeseen perturbations) and compliance (lower energy consumption, safer interaction with the environment, decoupling from perturbations).

In contrast, robots have traditionally been controlled with constant high gain negative error feedback control [16]. Especially for industrial robots, the rule of thumb is rather *"be stiff"*. Achieving high position accuracy has thus come at the cost of high energy consumption, and the necessity to build cages around robots to avoid human-robot contact. For autonomous mobile robots operating in human environments, safety and energy efficiency requirements are very different, and low-gain variable impedance control will be an essential characteristic of such robots.

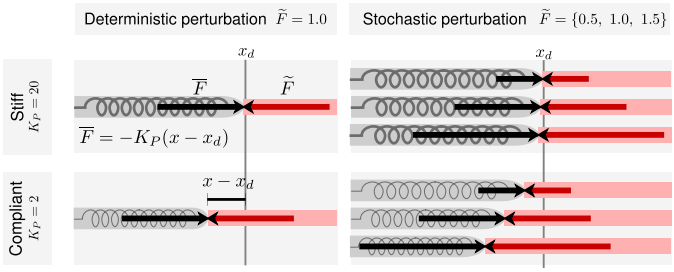


Fig. 1. Illustration of using different impedances (gains) in deterministic and stochastic perturbations. The perturbation pushes from the right; its force $\widetilde{F}$ is indicated with a red arrow. In the left column, perturbations are deterministic $\widetilde{F} = 1.0$, and in the right column they are stochastic $\widetilde{F} = \{0.5,\ 1.0,\ 1.5\}$. The 'robot manipulator' is depicted as a gray bar pushing from the left; its force $\overline{F}$ is indicated by a black arrow. It is computed with $\overline{F} = -K_P(x - x_d)$, i.e. a proportional controller based on the position error between the desired position x_d and actual position x. The desired position x_d is indicated by the vertical lines, and the gain $K_P = \{2, 20\}$ is represented by the thickness of the spring in the robot manipulator. In this illustration, all forces are in equilibrium, i.e. $\overline{F} = -\widetilde{F}$ and $\dot{x} = 0$. This illustration highlights that higher impedance (upper row) leads to higher forces for small position error, and position errors are therefore low with both deterministic and stochastic perturbations. Low impedance (lower row) leads to higher overall position errors in the face of perturbations.

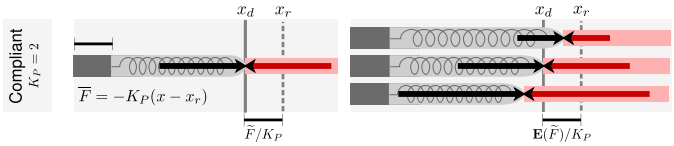


Fig. 2. With a deterministic perturbation (left), low error $(x - x_d)$ can be achieved despite low impedance if a feed-forward term is learned, as humans do [15]. In our simple example, this can be achieved analytically by providing a reference position x_r to the proportional controller $(\overline{F} = -K_P(x - x_r))$ that is different from the desired position x_d, and contains a fixed offset $x_r = x_d - \widetilde{F}/K_P$ based on the strength of the force field $\widetilde{F}$. With stochastic perturbations, $\widetilde{F}$ is not known in advance, but the average position error $\mathbf{E}(x - x_d)$ can still be reduced by using the average force of the field $x_r = x_d - \mathbf{E}(\widetilde{F})/K_P$. However, this offset term does not reduce the *variance* in the position error due to the stochasticity in the perturbations; this can only be reduced by increasing the gain, as in the upper row of Fig. 1. In humans, adaptation to the average of a stochastic force field whilst increasing the impedance was observed in [2], [11].

Although variable impedance control for robots is desirable, "[T]he selection of good impedance parameters [...] is not an easy task" [16]. The main reason is that deriving useful impedance controllers usually involves models of both the environment and the robot, as well as deep knowledge about

designing and parameterizing such controllers [5]. Recently, Mitrovic et al. [12] proposed to learn these models, which are then used in a model-based stochastic optimal controller to control a 1-DOF antagonistic actuator. As an alternative to model-based methods, we recently proposed the use of *model-free reinforcement learning* to simultaneously learn reference trajectories *and* variable impedance controllers [1]. We have implemented this approach on real robots with the PI^2 algorithm, which is derived from stochastic optimal control, and does not require a model of the robot or the environment.

In this paper, we use PI^2 to learn variable impedance controllers in deterministic and stochastic force fields, and show that the robot learns behavioral adaptations similar to those of humans, i.e. the two-fold strategy of combining a feed-forward term for deterministic perturbations with increased impedance for stochastic perturbations. In contrast to model-based optimal control, which is frequently used to simulate human behavior in computational motor control [8], PI^2 does not require a model and scales to high-dimensional tasks [17], thus making it a biologically more plausible learning algorithm. Furthermore, as PI^2 converges to local optimal solutions [18], our results lends support to the idea that learning based on stochastic optimality is a good model for human learning.

Most closely related to our work is the model described by Franklin et al. [3], which is used to simulate human data, and has also been implemented the context of a 1-DOF robot [4]. In [4], co-contraction of muscles is modelled by increasing the gains of the robot, and the robot learns to adapt its impedance, the forces it applies and reference trajectories. In contrast to our method, which is a generic policy learning algorithm, their learning method is specific to gain scheduling, and has not been demonstrated on more high-dimensional systems. Other reinforcement learning approaches to impedance control include [7], which has not been applied to higher-dimensional systems, and [9], where impedance is learned, but the reference trajectory is kept fixed.

In summary, we have previously shown that PI^2 learns variable impedance control on real robots in high-dimensional tasks, and in this contribution we demonstrate that it is also able to simulate human behavior based on stochastic optimality, without requiring a model.

II. Methods

Our robot 'subject' is a 7-DOF Barret arm, depicted in Fig. 3. We use an accurate physical simulation of the robot with the SL software package [13].

A. Experimental Protocol

In this paper, we consider the learning of reaching movements to a discrete, specified goal, and follow the experimental protocols in [3], [8], [11]. In particular, the experimental parameters below are taken from Experiment 1 in [11], and Experiment 3 in [8].

Initially the robot makes a straight movement with minimum-jerk velocity profile along the x-axis (distance $0.2m$, duration $1s$), away from its body [8]. This movement is

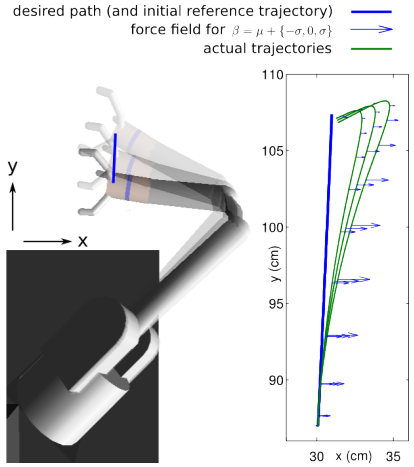


Fig. 3. The 7-DOF robotic arm used in this paper, simulated in SL. The reaching trajectory and force fields are depicted in the right graph (for $\sigma = 0.2886$). Note that although the visualizations in this paper are in 2D for ease of interpretation, the robot is simulated in full 3D space.

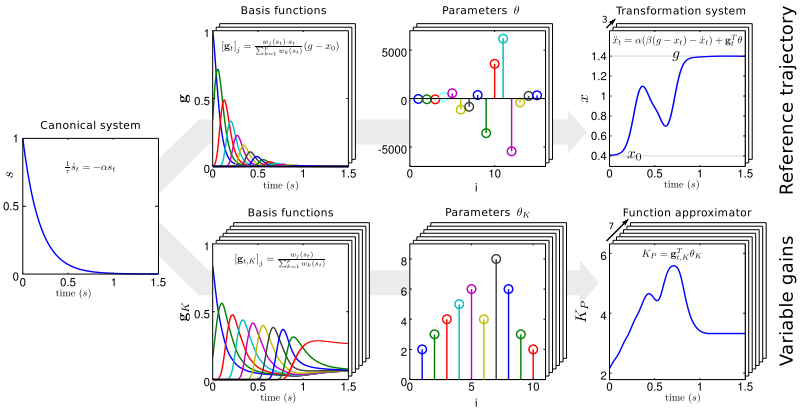
depicted in Fig. 3. We use a velocity dependent force field $\begin{bmatrix} F_x \\ F_y \end{bmatrix} = \beta \begin{bmatrix} 0 & 10 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}$, where F_x, F_y is the force applied to the subject's end-effector along the x/y-axis respectively, and $\dot{x}/\dot{y}$ is the velocity of the end-effector along the x/y-axis [11]. The strength of the force field β is sampled at the beginning of each trial from a Gaussian distribution $\mathcal{N}(1, \sigma)$. We apply four different force fields with $\sigma = \{0.0000, 0.1442, 0.2886, 0.4330\}$ [11]. The effect of the force field with $\sigma = 0.2886$ is depicted in Fig. 3.

The robot receives feedback about joint positions, end-effector position, and joint torques. After each trial, feedback on task achievement is given by the cost function:

$$J(\boldsymbol{\tau}_i) = \int_{t_i}^{t_N} 10^3 d(\mathbf{x}_t) + 10^{-2} \sum_{j=1}^{7} (K_{P,t}^j - K_{P,t}^{j,min}) + 10^{-3} |\ddot{\mathbf{x}}_t| \tag{1}$$

where • $d(\mathbf{x})$ is the distance in meters from the end-effector to the line connecting start and end-point of the movement, i.e. similar to Eq. 2 in [3]; this cost expresses that we do not want large errors in position from the straight desired path. Note that this desired path is invariant, and is not the same as the reference trajectory which is adapted through learning (i.e. compare x_r and x_d in Fig. 2). Using such an invariant desired path is also frequently assumed in modelling of human behavior [4], [3], [19]. • $\sum_{j=1}^{7} (K_P^j - K_P^{j,min})$ is the sum of the proportional gains (minus their minimum values) over all joints j; this cost expresses that we prefer low gains, as they lead to lower torque commands and safer human-robot interaction. In principle, we would expect similar results when penalizing motor commands directly. Penalizing the gains stems from our explicit goal of achieving compliant robots with low-gain control [1]. • $|\ddot{\mathbf{x}}|$ is the end-effector acceleration in m/s^2; this cost expresses that we do not want motions with high accelerations.

We now describe how the reference trajectory (Section II-B)



Dynamic Movement Primitives

$$\frac{1}{\tau}\ddot{x}_t = \alpha(\beta(g - x_t) - \dot{x}_t) + \mathbf{g}_t^T\boldsymbol{\theta} \quad \text{Transform. system} \quad (2)$$

$$[\mathbf{g}_t]_j = \frac{w_j(s_t)\cdot s_t}{\sum_{k=1}^{p} w_k(s_t)}(g - x_0) \quad \text{Basis functions} \quad (3)$$

$$w_j = \exp\left(-0.5h_j(s_t - c_j)^2\right) \quad \text{Gaussian kernel} \quad (4)$$

$$\frac{1}{\tau}\dot{s}_t = -\alpha s_t \quad \text{Canonical. system} \quad (5)$$

Gain schedules

$$K_{P,t} = \mathbf{g}_{t,K}^T\boldsymbol{\theta}_K \quad \text{Function approximator} \quad (6)$$

$$[\mathbf{g}_{t,K}]_j = \frac{w_j(s_t)}{\sum_{k=1}^{p} w_k(s_t)} \quad \text{Basis functions} \quad (7)$$

Fig. 4. Dynamic Movement Primitives (DMPs). The core idea behind DMPs is to perturb a simple linear dynamical system (the first part of Eq. 2) with a non-linear component ($\mathbf{g}_t^T\boldsymbol{\theta}$) to acquire smooth movements of arbitrary shape. The non-linear component consists of basis functions $\mathbf{g}_t$, multiplied with a parameter vector $\boldsymbol{\theta}$. The canonical system s_t represents the phase variable, which is 1 at the beginning of the movement, and 0 at the end. The movement in the left figure (upper row) has a duration of 1 second, after which x has reached the goal, i.e. $x_{t>1.0} = g$. Proportional gain schedules $K_{P,t}$ (lower row) are not transformation systems, but rather represented directly with the function approximator $\mathbf{g}_{t,K}^T\boldsymbol{\theta}_K$.

and variable gain schedules (Section II-C) are represented. In Section II-D we present the model-free reinforcement algorithm that learns to minimize the cost function in Eq. 1, by adapting the reference trajectories and gain schedules.

B. Dynamic Movement Primitives

The reaching trajectory is represented by a Dynamic Movement Primitive (DMP) [6], which consists of a set of dynamic system equations listed, visualized, and explained in Fig. 4. We leave the details to [6], [18]. For this paper, the important features of DMPs are that • When integrating a DMP over time, it generates a 1-dimensional output trajectory $[x_t\ \dot{x}_t\ \ddot{x}_t]$ • DMPs converge from the initial value x_0 towards the goal parameter g. • The general shape of the trajectory is determined by the parameters $\boldsymbol{\theta}$.

Multi-dimensional DMPs are represented by coupling several transformation systems as in Eq. 2 with one shared phase variable s. In this paper, the DMP has three transformation systems, which represent the 3-dimensional reference trajectory of the robot's end-effector in Cartesian space $[\mathbf{x}_{r,t}\ \dot{\mathbf{x}}_{r,t}\ \ddot{\mathbf{x}}_{r,t}]$.

The initial parameters $\boldsymbol{\theta}$ are trained with supervised learning [6], so that the reference trajectory has a minimum-jerk velocity profile, and generates the trajectory depicted in Fig. 3. The reference end-effector positions are converted into joint space using the Jacobian pseudo-inverse. The resulting joint velocities $\dot{\mathbf{q}}_{r,t}$ are integrated and differentiated, to get joint positions $\mathbf{q}_{r,t}$ and accelerations $\ddot{\mathbf{q}}_{r,t}$ respectively.

C. Representing Variable Gain Schedules in DMPs

Given the reference joint trajectory $[\mathbf{q}_{r,t}\ \dot{\mathbf{q}}_{r,t}\ \ddot{\mathbf{q}}_{r,t}]$ generated above, the motor command torques $\mathbf{u}$ for our robot are calculated via a PD/feed-forward control law:

$$\mathbf{u} = -\mathbf{K}_P(\mathbf{q} - \mathbf{q}_r) - \mathbf{K}_D(\dot{\mathbf{q}} - \dot{\mathbf{q}}_r) + \mathbf{u}_{ID} \quad (8)$$

$$\mathbf{K}_D = C\sqrt{\mathbf{K}_P} \quad (9)$$

where $\mathbf{K}_P$, $\mathbf{K}_D$ are the positive definite position and velocity gain matrices, and C is a constant scale factor set manually for each joint. The feed-forward control term $\mathbf{u}_{ID}$ is computed with an inverse dynamics (ID) controller[1] based on a Newton-Euler algorithm[2]. Thus, the impedance of a joint is parameterized by the choice of the gains $\mathbf{K}_P$ (stiffness) and $\mathbf{K}_D$ (damping).

The key to variable impedance control is to allow $\mathbf{K}_P$ to vary as the movement is executed. As introduced in [1], this can be achieved by representing the gain schedules as extra dimensions in the DMP. Since the gains do not have a specific goal value, the proportional gains are not represented as transformation systems that converge to g, but computed directly as a function approximator $K_{P,t} = \mathbf{g}_{t,K}^T\boldsymbol{\theta}_K$, as depicted in Fig. 4.

In our experiments, we use supervised learning to initialize $\boldsymbol{\theta}_K$ such that the proportional gains of the 7 joints are constant over time, and have the values $\mathbf{K}_P^{min} = \{60, 60, 16, 16, 6, 6, 1.6\}$. These are 0.4 times the default gains we use for this robot, and the minimum gains we allow during learning, as too low gains lead to poor tracking such that the robot frequently runs into its joint limits. Although we start out with a gain schedule that is constant over time, we shall see in the next section that varying $\boldsymbol{\theta}_K$ leads to varying gain schedules, which are adapted to external perturbations through reinforcement learning.

In summary, in the Dynamic Movement Primitive used in this paper, there is one canonical system (representing the phase variable s_t) which drives 3 transformation systems (representing the 3-D end-effector position over time) and 7 function approximators (representing the gain schedules of the 7 joints). We chose to specify the reference trajectory in end-

[1]The inverse dynamics feed-forward torques $\mathbf{u}_{ID}$ only compensate for forces due to gravity, inertia and Coriolis effects, but *not* for the force fields we generate. Therefore, the feed-forward term which is learned to compensate for the force field perturbation $\widetilde{F}$ is completely independent of $\mathbf{u}_{ID}$.

[2]Although our learning method is model-free, our inverse dynamics controller *does* require a model of the robot. This general model is not used to learn the specific task of compensating for the force fields.

effector space since this is the element the 'subject' has to regulate to fulfill the task and receives feedback on. In contrast we chose to regulate gains in joint space to avoid to have to specify and additional, arbitrary null-space behavior.

D. Policy Improvement with Path Integrals – PI²

Given the DMP representation above, the goal is to learn the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\theta}_K$ which minimize the cost function in Eq. 1. To do so, we use the policy improvement algorithm PI² [18]. Since PI² learns $\boldsymbol{\theta}$ and $\boldsymbol{\theta}_K$ simultaneously with the same method, from now on we simply denote these parameters for the end-effector trajectory and gain schedules as one parameter vector $\boldsymbol{\theta}$. Cost functions for PI² take the generic form:

$$J(\boldsymbol{\tau}_i) = \phi_{t_N} + \int_{t_i}^{t_N} (r_t + \frac{1}{2}\boldsymbol{\theta}^T \mathbf{R}\boldsymbol{\theta})\ dt \quad \text{Traj. cost} \quad (10)$$

where J is the finite horizon cost over a trajectory $\boldsymbol{\tau}_i$ starting at time t_i and ending at time t_N. This cost consists of a terminal cost ϕ_{t_N}, an immediate cost r_t, and an immediate control cost $\frac{1}{2}\boldsymbol{\theta}^T\mathbf{R}\boldsymbol{\theta}$. The specific cost function for the task considered in this paper was given in Eq. 1, and adheres to this generic format.

Policy improvement methods minimize cost functions through an iterative process of exploration and parameter updating, which we explain using Fig. 5. Exploration is done by executing a Dynamic Movement Primitive M times, each time with slightly different policy parameters $\boldsymbol{\theta} + \boldsymbol{\epsilon}^{\boldsymbol{\theta}}{}_{t,k}$ which is added to explore the parameter space, as in Eq. 11. This noise is sampled from a Gaussian distribution with variance $\boldsymbol{\Sigma}^{\boldsymbol{\theta}}$. A similar Gaussian exploration is applied to the gain schedules as in Eq. 12.

$$\frac{1}{\tau}\ddot{x}_t = \alpha(\beta(g - x_t) - \dot{x}_t) + \mathbf{g}_t^T(\underbrace{\boldsymbol{\theta} + \boldsymbol{\epsilon}^{\boldsymbol{\theta}}{}_{t,m}}_{\text{Shape exploration}}) \quad (11)$$

$$K_{P,t} = \mathbf{g}_{t,K}^T(\underbrace{\boldsymbol{\theta}_K + \epsilon_{t,m}^{\theta_K}}_{\text{Gain exploration}}) \quad (12)$$

These 'noisy' DMP parameters generate slightly different reference trajectories $\{\ddot{\mathbf{x}}_{r,t}, \dot{\mathbf{x}}_{r,t}, \mathbf{x}_{r,t}\}_{m=1\ldots M}$ and gain schedules $\{\mathbf{K}_{P,t}\}_{m=1\ldots M}$, which each lead to different costs. Given the costs and noisy parameters of the M DMP executions, called *roll-outs*, policy improvement methods then update the parameter vector $\boldsymbol{\theta}$ such that it is expected to generate movements that lead to lower costs in the future. The process then continues with the new $\boldsymbol{\theta}$ as the basis for exploration.

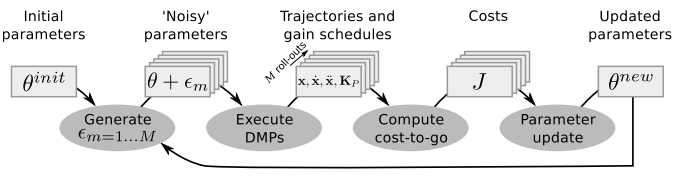


Fig. 5. Generic loop of policy improvement algorithms.

The most crucial part of the policy improvement loop in Fig. 5 is the parameter update; it is here that the key differences between PI² and other policy improvement methods lie. Rather than focussing on its derivation from first principles of stochastic optimal control, which is presented extensively in [18], we provide a post-hoc interpretation of the resulting update rule in Fig. 6.

PI² Parameter Update Rule

$$S(\boldsymbol{\tau}_{i,m}) = \phi_{t_N,m} + \sum_{j=i}^{N-1} r_{t_j,m} + \frac{1}{2}\sum_{j=i+1}^{N-1} (\boldsymbol{\theta} + \mathbf{M}_{t_j,m}\boldsymbol{\epsilon}^{\boldsymbol{\theta}}{}_{t_j,m})^T \mathbf{R}(\boldsymbol{\theta} + \mathbf{M}_{t_j,m}\boldsymbol{\epsilon}^{\boldsymbol{\theta}}{}_{t_j,m}) \quad (13)$$

$$\mathbf{M}_{t_j,m} = \frac{\mathbf{R}^{-1}\mathbf{g}_{t_j}\ \mathbf{g}_{t_j}^T}{\mathbf{g}_{t_j}^T\mathbf{R}^{-1}\mathbf{g}_{t_j}} \quad (14)$$

$$P(\boldsymbol{\tau}_{i,m}) = \frac{e^{-\frac{1}{\lambda}S(\boldsymbol{\tau}_{i,m})}}{\sum_{l=1}^{M}[e^{-\frac{1}{\lambda}S(\boldsymbol{\tau}_{i,l})}]} \quad (15)$$

$$\delta\boldsymbol{\theta}_{t_i} = \sum_{m=1}^{M}\left[P(\boldsymbol{\tau}_{i,m})\ \mathbf{M}_{t_i,m}\ \boldsymbol{\epsilon}^{\boldsymbol{\theta}}{}_{t_i,m}\right] \quad (16)$$

$$[\delta\boldsymbol{\theta}]_j = \frac{\sum_{i=0}^{N-1}(N-i)\ w_{j,t_i}\ [\delta\boldsymbol{\theta}_{t_i}]_j}{\sum_{i=0}^{N-1} w_{j,t_i}(N-i)} \quad (17)$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \delta\boldsymbol{\theta} \quad (18)$$

Fig. 6. PI² parameter update rule:
Eq. 13 – Determine cost-to-go of each roll-out $S(\boldsymbol{\tau}_{i,m})$ at each time step i. This is an evaluation of the cost function $J(\boldsymbol{\tau}_i)$ in Eq. 10 , which is task-dependent and provided by the user. The matrix $\mathbf{M}_{t_j,m}$ (Eq. 14) is needed to project the exploration noise onto the parameter space.
Eq. 15 – Compute probability of each roll-out $P(\boldsymbol{\tau}_{i,m})$ at each time step i by exponentiating the cost-to-go. The intuition behind this step is that trajectories of lower cost should have higher probabilities.
Eq. 16 – Average over roll-outs. Compute the parameter update $\delta\boldsymbol{\theta}$ for each time step i through probability weighted averaging over the exploration $\boldsymbol{\epsilon}^{\boldsymbol{\theta}}$ of all M roll-outs. Trajectories with higher probability, and thus lower cost, therefore contribute more to the parameter update. Again, $\mathbf{M}_{t_j,m}$ projects the exploration noise onto the parameter space.
Eq. 17 – Average over time-steps. In the final step, we average the parameter update $\delta\boldsymbol{\theta}_{t_i}$ per time step i over all time steps. Each parameter update is weighted according to the number of steps left in the trajectory. This is to give earlier points in the trajectory higher weights, as they influence a larger part of the trajectory. They are also weighted with the activation of the corresponding basis function w_j at time t_i, as the influence of parameter θ_j is highest when w_j is highest. Finally, the actual parameter update is performed with Eq. 18.

In our experiments, we performed 100 PI² updates with $M = 10$ roll-outs per update for each of the four force fields. The exploration noise was $\boldsymbol{\Sigma}^{\boldsymbol{\theta}} = 10^2$ for the Cartesian positions, and $\boldsymbol{\Sigma}^{\boldsymbol{\theta}_K} = 10^{-3} \cdot \mathbf{K}_P^{min}$ for the 7 gain schedules[3], where $\mathbf{K}_P^{min}$ are the minimum gains as listed in Section II-C.

III. Results

After each PI² update, three roll-outs were executed *without* exploration noise for evaluation purposes. For these three roll-outs, force fields with strength $\beta = 1 - \sigma,\ 1,\ 1 + \sigma$ were used. For all four forcefields (each with a different level of stochasticity determined by σ), the reference trajectories, force fields and actual trajectories at various stages of learning are

[3]The relatively low exploration noise for the gains does not express less exploration per se, but is rather due to numerical differences in using the function approximator to model the gains directly (Eq. 6) rather than as the non-linear component of a transformation system (Eq. 2).

depicted on the last page in Fig. 10. For comparison, the reference trajectories after 100 updates for the four force fields are depicted together in Fig. 7. These correspond to the after-effects that occur when the force field is turned off, similar to the graphs in [15], [11].

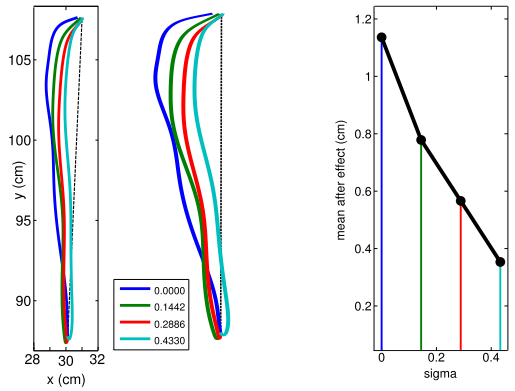


Fig. 7. Left: After-effects (i.e. reference trajectories) after 100 updates for each of the force fields. Center: Same, with x axis scaled $\times 2$ for visualization purposes. Right: Average after-effect (i.e. mean distance to the trajectory before learning) as a function of force field stochasticity.

The learning curves for the four force fields are depicted in Fig. 8. The sum over all gain schedules at various stages during learning are depicted in Fig. 9.

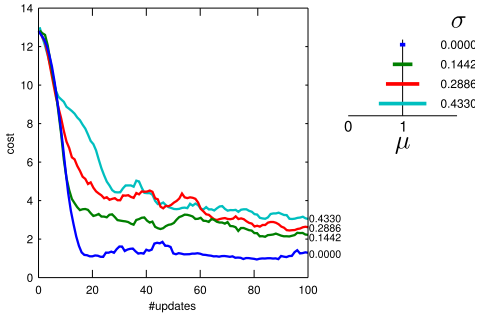


Fig. 8. Learning curves for the different force fields. The y-axis represents the average costs over the three evaluation roll-outs performed after each update. Curves are smoothed with a moving average filter of window size 7.

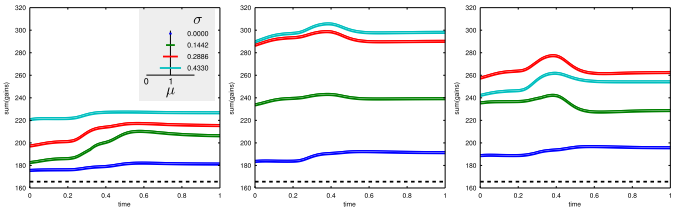


Fig. 9. Sum of the individual gain schedules of the 7 joints after 10, 34, 100 updates. The gains before learning (which are also the minimum gains) are depicted as dashed lines.

A. Discussion

A closer inspection of the individual cost components, reference trajectories and gain schedules leads to the following observations:

• The main adaptation of the reference trajectory happens in the first 10 updates. As can be seen in Fig. 10, the reference trajectories after 10 updates are already close to the shape they have at the end of learning after 100 updates. This suggests that the robot first learns the feed-forward term to compensate for the average force field with $\beta = 1$. Since the force field pushes from left to right, the reference trajectory is placed to the left, and is approximately a mirror image of the perturbed trajectory before learning. This is consistent with the observations on human adaptation in [15], [2], [11].

• Between 10 and 30 updates, gains go up quickly (reaching their maximum value after 34 updates) to compensate for stochasticity in the force field. Higher stochasticity σ leads to higher peak values for the gains (center graph Fig. 9) and smaller after-effects (right graph Fig. 7), which is consistent with human adaptations observed in [2], [11].

• After 30 updates, gain schedules are fine-tuned to achieve low position error with low gains. This lowering of gains slowly continues even after 100 updates. Quick increases in gains followed by slow decreases is consistent with our observations in several robotic manipulation tasks [1], [17].

In summary, the robot adapts to the mean of perturbation by moving the reference trajectory in the opposite direction of the force field, and adapts to stochasticity in the perturbations by increasing the impedance. These are qualitatively similar results to those observed biophysics experiments [11].

For now, the results could not be modelled quantitively, as there are some clear differences between humans and our robotic platform. In particular, the kinematics and dynamics of the robot are not the same as the human body; this alone may explain many of the quantitive differences. Also, humans learn in 'muscle space', and higher impedance is caused by co-contraction of the muscles. In contrast, our robot learns in gain space, as in [4]. Finally, the robot learns much slower than humans (1000 trials vs. 175 in [11]). We believe the main reason for this is that in contrast to humans, our controller and PI^2 have no built-in reflexes, and start without any initial knowledge about the domain. Also, humans learn continually during and after each trial, whereas PI^2 requires M trials ($M = 10$ in this paper) to be performed before updating the parameters. Our current research focuses mainly on applying our methods to more human-like kinematics, and biologically plausible muscle models. We are also developing a version of PI^2 in which the last M trials are kept in a FIFO buffer, allowing updates to be performed after each trial, which would enable continual learning.

As in [3], [4], we assume a straight desired path to calculate the task reward, i.e. deviations from this (invariant) desired path are penalized. Without this position error penalty, we are not able to simulate the human movement data. The plausibility of a desired trajectory in biology [19] is still under debate, and the role of the desired trajectory in our system also deserves further investigation.

An important part of the robot's adaptation to the force field is achieved by changing the reference trajectory, i.e. in the analytical example in Fig. 2 this was $\overline{F} = -K_P(x - x_r)$,

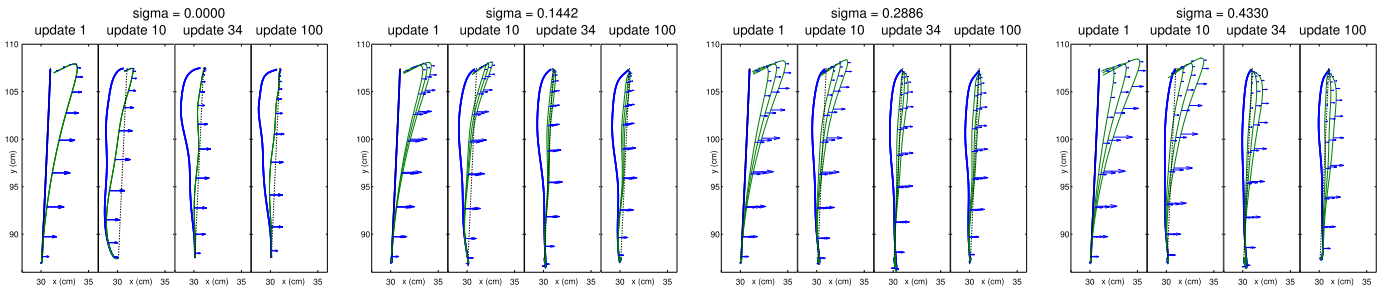


Fig. 10. Reference (blue) and actual (green) hand trajectories as learning progresses, for the different force fields. Each row represents a force field with different σ. The actual trajectories represent the perturbed motions when executing the reference trajectories for force fields with strengths $\beta = \{1-\sigma, \quad 1, \quad 1+\sigma\}$.

with $x_r = x_d - \widetilde{F}/K_P$. The position offset $\widetilde{F}/K_P$ leads to larger errors between the reference path and actual trajectory $(x_d - x)$ and thus larger forces, which counteract the force field. Changing the reference position to exert a force is known as *indirect force control* [16]. An alternative would be to directly learn a reference force F_r to compensate for the force field, and perform *direct force control*, i.e. $\overline{F} = -K_P(x - x_d) + F_r$. Since both direct and indirect force control will lead to similar after-effects, it is not clear which approach humans use. In our future work, we will compare the results of learning with direct and indirect force control on our robot platform. Any differences observed on the robot could assist us in designing experiments to verify which form of control humans use to compensate for stochastic perturbations.

IV. Conclusion

Dynamic Movement Primitives and the PI^2 algorithm have previously been applied to learning reference trajectories and gain schedules for complex high-dimensional robotic tasks [17], [1]. Our approach offers a pragmatic solution for robotics applications, where exploiting the power of adaptive impedance is desirable, but where it is difficult to tune impedance controllers, even for deterministic problems.

In this paper we demonstrate how PI^2 is able to find motor policies that qualitatively replicate human movement data in stochastic force field experiments. As PI^2 is a model-free algorithm based on stochastic optimal control, these results 1) support the hypothesis that human learning in such situations is based on stochastic optimality; 2) demonstrate that locally optimal policies can be learned from experience without requiring a model of the force field or the (bio)mechanical system, thus making it a more biologically plausible alternative to model-based optimal control methods.

Acknowledgments

This research was supported in part by National Science Foundation grants ECS-0325383, IIS-0312802, IIS-0082995, IIS-9988642, ECS-0326095, ANI-0224419, and the ATR Computational Neuroscience Laboratories. F.S. was supported by a Research Fellowship from the German Research Foundation (DFG). J.B. was supported by an advanced researcher fellowship from the Swiss National Science Foundation. E.T. was supported by a Myronis Fellowship.

References

[1] J. Buchli, F. Stulp, E. Theodorou, and S. Schaal. Learning variable impedance control. *International Journal of Robotics Research*, 2011.

[2] C. Takahashi, D. Scheidt, and R. Reinkensmeyer. Impedance control and internal model formation when reach in a randomly varying dynamical environment. *Journal of Neurophysiology*, 86:1047–51, 2001.

[3] D. Franklin, E. Burdet, K. Tee, R. Osu, C. Chew, T. Milner, and M. Kawato. CNS learns stable, accurate, and efficient movements using a simple algorithm. *Journal of Neuroscience*, 28(44):11165–73, 2008.

[4] G. Ganesh, A. Albu-Schäffer, M. Haruno, M. Kawato, and E. Burdet. Biomimetic motor behavior for simultaneous adaptation of force, impedance and trajectory in interaction tasks. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 2705–2711, 2010.

[5] N. Hogan. Impedance control - an approach to manipulation. I - theory. II - implementation. III - applications. *ASME, Transactions, Journal of Dynamic Systems, Measurement, and Control*, 107:1–24, 1985.

[6] A. J. Ijspeert, J. Nakanishi, and S. Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2002.

[7] J. Izawa, T. Kondo, and K. Ito. Biological arm motion through reinforcement learning. *Biological Cybernetics*, 91(1):10–22, 2004.

[8] J. Izawa, T. Rane, O. Donchin, and R. Shadmehr. Motor adaptation as a process of reoptimization. *Journal of Neuroscience*, 2008.

[9] B. Kim, J. Park, S. Park, and S. Kang. Impedance learning for robotic contact tasks using natural actor-critic algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 40(2), 2009.

[10] L. Selen, D. Franklin, and D. Wolpert. Impedance control reduces instability that arises from motor noise. *Journal of Neuroscience*, 7(40):12606–16, 2009.

[11] M. Mistry, E. Theodorou, H. Hoffmann, and S. Schaal. the dual role of uncertainty in force field learning. In *Abstracts of the 18th Annual Meeting of Neural Control of Movement (NCM)*, 2008.

[12] D. Mitrovic, S. Klanke, M. Howard, and S. Vijayakumar. Exploiting sensorimotor stochasticity for learning control of variable impedance actuators. In *Proc. IEEE-RAS International Conference on Humanoid Robots*, 2010.

[13] S. Schaal. The SL simulation and real-time control software package. Technical report, University of Southern California, 2009.

[14] R. Scheidt, B. Dingwell, and F. Mussa-Ivaldi. Learning to move amid uncertainty. *Journal of Neurophysiology*, 86:971–85, 2001.

[15] R. Shadmehr and F. A. Mussa-Ivaldi. Adaptive representation of dynamics during learning of a motor task. *Journal of Neuroscience*, 14(5):3208–3224, 1994.

[16] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics – Modelling, Planning and Control*. Springer, London, 2009.

[17] F. Stulp, J. Buchli, E. Theodorou, and S. Schaal. Reinforcement learning of full-body humanoid motor skills. In *10th IEEE-RAS International Conference on Humanoid Robots*, 2010.

[18] E. Theodorou, J. Buchli, and S. Schaal. A generalized path integral approach to reinforcement learning. *Journal of Machine Learning Research*, 11(Nov):3137–3181, 2010.

[19] D. Wolpert, Z. Ghahramani, and M. Jordan. Are arm trajectories planned in kinematic or dynamic coordinates? An adaptation study. *Experimental Brain Research*, 103:460–470, 1995.