Auton Robot
DOI 10.1007/s10514-009-9118-y

On-line learning and modulation of periodic movements with nonlinear dynamical systems

Andrej Gams · Auke J. Ijspeert · Stefan Schaal · Jadran Lenarčič

Received: 11 September 2008 / Accepted: 29 April 2009

Abstract The paper presents a two-layered system for (1) learning and encoding a periodic signal without any knowledge on its frequency and waveform, and (2) modulating the learned periodic trajectory in response to external events. The system is used to learn periodic tasks on a humanoid HOAP-2 robot. The first layer of the system is a dynamical system responsible for extracting the fundamental frequency of the input signal, based on adaptive frequency oscillators. The second layer is a dynamical system responsible for learning of the waveform based on a built-in learning algorithm. By combining the two dynamical systems into one system we can rapidly teach new trajectories to robots without any knowledge of the frequency of the demonstration signal. The system extracts and learns only one period of the demonstration signal. Furthermore, the trajectories are robust to perturbations and can be modulated to cope with a dynamic environment. The system is computationally inexpensive, works on-line for any periodic signal, requires no additional signal processing to determine the frequency of the input signal and can be applied in parallel to multiple dimensions. Additionally, it can adapt to changes in frequency and shape, e.g. to non-stationary signals, such as hand-generated signals and human demonstrations.

Keywords Learning by imitation · Adaptive frequency oscillators · Dynamic movement primitives · Modulating learned trajectory

A. Gams (✉) · J. Lenarčič
“Jožef Stefan” Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia
e-mail: andrej.gams@ijs.si

A.J. Ijspeert
School of Computer and Communication Sciences, EPFL—École Polytechnique Fédérale de Lausanne, Station 14, 1015 Lausanne, Switzerland
e-mail: auke.ijspeert@epfl.ch

S. Schaal
Computational Learning and Motor Control Laboratory, University of Southern California, Los Angeles, CA 90089-2520, USA

1 Introduction

One of the central issues in robotics and animal motor control is the problem of trajectory generation and modulation. Since in many cases trajectories have to be modified on-line when goals are changed, obstacles are encountered, or when external perturbations occur, the notions of trajectory generation and trajectory modulation are tightly coupled.

In this article we address some of the issues related to trajectory generation and modulation, with an emphasis on the supervised learning of periodic trajectories, including the learning of the frequency. Other addressed issues include robust movement execution despite external perturbations, and modulation of the trajectory to reuse it under modified conditions.

For the learning of a periodic trajectory without specifying the period and without using traditional off-line signal processing methods, our approach suggests splitting the task into two sub-tasks: (1) frequency extraction, and (2) the supervised learning of the waveform. This is done using two ingredients: nonlinear oscillators for the frequency adaptation, and nonparametric[1] regression techniques for shaping

[1]The term “nonparametric” is to indicate that the data to be modelled stem from very large families of distributions which cannot be indexed by a finite dimensional parameter vector in a natural way. It does not mean that there are no parameters.

the attractor landscapes according to the demonstrated trajectories. The systems are designed such that after having learned the trajectory, simple changes of parameters allow modulations in terms of, for instance, frequency, amplitude and oscillation offset, while keeping the general features of the original trajectory.

The system we propose in this paper is based on the motion imitation approach described in Ijspeert et al. (2002a, 2002b, 2002c), Schaal et al. (2007). That approach uses two dynamical systems like the system presented here, but with a simple nonlinear oscillator to generate the phase and the amplitude of the periodic movements. A major drawback of that approach is that it requires the frequency of the demonstration signal to be explicitly specified. This means that the frequency has to be either known or extracted from the recorded signal by signal processing methods, e.g. Fourier analysis. The main difference of our new approach is that we use an adaptive frequency oscillator (Buchli and Ijspeert 2004; Righetti et al. 2006), which has the process of frequency extraction and adaptation totally embedded into its dynamics. The frequency does not need to be known or extracted, nor do we need to perform any transformations (Righetti and Ijspeert 2006). This simplifies the process of teaching a new task/trajectory to the robot. To our knowledge, this is the first system of this kind. Additionally, the system can work incrementally in on-line settings. The system also generalizes an approach that proposes the use of feedback instead of an oscillatory dynamical system (Gams et al. 2007), but thus imposes great restrictions on the generality and the possibility of modulation.

Our approach is loosely inspired from dynamical systems observed in vertebrate central nervous systems, in particular central pattern generators (Ijspeert 2008). Additionally, our work fits in the view that biological movements are constructed out of the combination of "motor primitives" (Schaal 1999; Mataric 1998), and the system we develop could be used as blocks or motor primitives for generating more complex trajectories.

In the next sections, we first provide a review of different approaches for trajectory generation and modulation. We define five desirable properties that a trajectory generation system should posses, which helps to structure the review. We also give a short overview of different biological concepts related to our work (Sect. 2). We then present the structure of the system and the learning algorithm used (Sect. 3). Properties and possibilities of modulation are given next (Sect. 4), followed by experimental evaluation of the system (Sect. 5). We conclude the paper by discussing the results and presenting some of possible improvements of the system (Sect. 6).

2 Different approaches to learning trajectories

There are numerous studies on the learning and modulation of trajectories in a variety of fields including robotics, machine learning, artificial neural networks, computational neuroscience, machine vision and graphics animation. In this paper, we are mainly interested in human periodic trajectory demonstration, where *trajectory* is a variable that evolves over time. The variable can be multi-dimensional, and can encode different quantities such as a position in Cartesian space, joint angle, or a torque.

Desirable characteristics

A system for encoding movement trajectories should ideally possess at least five desirable characteristics:

1. The ease of learning and representing a desired trajectory, where learning is ideally one-shot and computationally inexpensive.
2. Compactness of representation: Multidimensional trajectories can quickly fill large data arrays; therefore the representation should compress the data into a compact form.
3. Ability to deal with noise, perturbations and changes in a dynamic environment. When used in a control task, the trajectory generation system should be able to make on-line modifications to the trajectory, e.g., for obstacle avoidance.
4. Ease of re-use for related tasks and modification for new tasks. Ideally, the system should be able to generate a family of similar trajectories with modifying only few parameters, instead of having to learn each trajectory individually.
5. Categorization for trajectory recognition. Representations for encoding a trajectory should serve for classification of trajectories, e.g. to measure their similarities and dissimilarities.

The possibility of sequencing trajectories, as one of the possible characteristics, was intentionally omitted from the list, as it requires a higher level of task-oriented control system, which is not the topic of this paper.

As we will see, very few approaches fulfill these different desirable characteristics.

Representation. Choosing the representation or encoding of the trajectory is a central issue. Among the simplest representations is simple storing of large time-indexed vectors (Kawamura and Fukao 1994). Alternatively, more compact representations can be constructed by using interpolation algorithms such as spline fitting and only storing key via-points (Schaal 1999). Other approaches include function-approximators, such as neural networks (Maass et al. 2002; Zegers and Sundareshan 2003) and recurrent neural networks (RNN) (Paine and Tani 2004; Tani and Ito 2003).

Lately, probabilistic models, such as Hidden Markov Models (HMM) (Inamura et al. 2004), Bayesian Networks (Grimes et al. 2006), or Gaussian Mixture Models (GMM) (Calinon et al. 2007), have also been used. Probabilistic models can also be used in combination with knowledge-based systems (Hersch et al. 2008). Rhythmic motions require particular types of encoding to encapsulate their cyclic nature. Different approaches include cyclically reading vectors, cyclic vector fields (Li and Horowitz 1999; Okada et al. 2002), and encoding trajectories into the limit cycle behavior of nonlinear oscillators (Nishii 1998; Ijspeert et al. 2002b).

Some representations use hierarchical structures in which a trajectory is encoded as a superposition and/or sequence of simpler trajectories (Mussa-Ivaldi 1997; Tsuji et al. 2002; Rohrer and Hogan 2003; Fod et al. 2002; Drumwright et al. 2004). Such approaches are often inspired from the concept of motor primitives in vertebrate motor control (Schaal 1999; Mataric 1998), and are interesting ways of making the encoding of multiple trajectories more compact.

Learning. The learning algorithms are closely related to the chosen type of representation. When time-indexed vectors (Kawamura and Fukao 1994) are used, no learning algorithm is needed. Representations based on spline-fitting typically use well-established fitting algorithms (Miyamoto et al. 1996). The via-points used by these algorithms can be either assigned automatically or by the user.

When the chosen representation is linear in the parameters, regression can be sufficient (Maass et al. 2002; Ijspeert et al. 2002b), and has the advantage of fast, one-shot learning. Gradient-descent algorithms are extensively used, particularly for neural networks, such as variants of the back-propagation algorithm (Simard and LeCun 1991). Alternatively, evolutionary algorithms can be used for instantiating the network's parameters given a cost function describing the desired trajectories (Ijspeert et al. 1999). Both gradient-descent and evolutionary algorithms tend to be slow.

Other learning algorithms also include reinforcement learning, as in, for example, Guenter et al. (2007).

Trajectory generation and modulation. Many researchers are not only interested in correctly reproducing a learned trajectory, but also in adapting a learned trajectory to new conditions. Simple modulations include repeating trajectories at different speeds and/or different amplitudes. More complex modulations might be time or space dependent such as those needed to reproduce a trajectory while avoiding an obstacle along the original path. Other modulations might be required when the robot is subjected to external forces. One should notice that some representations are more suitable than others for modulation and dealing with perturbations. Reproducing trajectories at different speeds and/or different amplitudes can be obtained using simple scaling laws with time-indexed vector representations (Kawamura and Fukao 1994). Representations using spline fitting and via-points can also be modulated using dynamic optimization methods (Miyamoto et al. 1996). Such approaches, however, present the disadvantage that the control policy requires time as an explicit variable, which makes them highly sensitive toward unforeseen perturbations in the environment that would disrupt the normal time flow. Neural networks can be trained to generalize and produce a range of different motions from a fixed set of training examples, and this without requiring explicit time indexing (Zegers and Sundareshan 2003).

Specific features of human-like movements cannot be accounted for by simple scaling. These features can be encapsulated by optimization criteria, such as minimum torque-change (Kawato 1996) or minimum variance of end position (Harris and Wolpert 1998). Alternatively, dynamical systems can be designed to replicate human movement features, and to generate arbitrary movements by the modulation of attractor points (Bullock and Grossberg 1989). Ad-hoc kinematic planning models can also replicate some features of human movement generalization. Different interpolation and extrapolation techniques have also been developed for motion synthesis in the field of animation and humanoid robotics. Methods synthesizing a trajectory from a given set of learned motions are another option (Ude et al. 2007 or Mezger and Giese 2005).

Reproducing and modulating trajectories is also possible using a system of coupled nonlinear oscillators (Righetti and Ijspeert 2006), where adaptive frequency oscillators are used to learn separate frequency components of the demonstration trajectory, and added to recreate the signal. The drawback of this approach is that the system can only be modulated in speed and amplitude. Also the return to the limit cycle regime after a perturbation is relatively slow because of the time needed to converge to the right phase lags between the multiple oscillators used to encode a specific one-dimensional signal. For complex signals and for scaling into multi-dimensionality, the number of oscillators needed grows quickly, leading to a complex system structure.

Dealing with perturbations. Dealing with perturbations, e.g. when performing a trajectory with a robot in the presence of obstacles and/or external forces, is a complex and to a great extent task-dependent problem. For instance, hitting an obstacle with a limb will require different trajectory modulation during a walking sequence compared to a reaching movement. In some situations, trajectories might not need to be modulated if the perturbations are small and/or short. If we assume that the robot is provided with an on-line tracking controller (e.g. a PID feedback controller, potentially extended with a feed-forward controller), the feedback control loop of the tracking controller could be sufficient to rapidly overcome the perturbation by adjusting torques and diminishing the error between desired and actual positions. Large perturbations (e.g. due to contacts with an obstacle) however require on-line modulations of the desired trajectories in order to prevent risks of damage or of being permanently stuck. For example, a criterion can be used to modulate the learned trajectories (see for example Hersch et al. 2008). Another approach in robotics relies on encoding desired trajectories in terms of vector fields, e.g. velocity fields (Li and Horowitz 1999), or potential fields (Khatib 1986). The vector fields essentially represent an attractor landscape, with the desired trajectory as attractor trajectory. This offers the opportunity to introduce repulsive forces for avoiding obstacles. This is also one of the features of our system.

In view of the presented review, our system is based on encoding trajectories as a limit cycle produced by a two layered dynamical system, with one layer for frequency adaptation and for producing the phase of the signal, and the second layer for learning the waveform as a non-linear filter. As will be shown in the next sections, the complete system is low-dimensional, learns rapidly, allows modulation, and is robust against perturbations.

3 System for learning the frequency and waveform of an unknown signal

In this section we first present the structure of the proposed system, followed by the detailed explanation of the two building blocks: (1) The *Canonical Dynamical System* for the frequency adaptation, and (2) the *Output Dynamical System* for the learning of the waveform.

3.1 System structure

Figure 1 shows the structure of the proposed system for the learning of the frequency and the waveform of the input signal. The input into the system $y_{demo}(t)$ is an arbitrary periodic signal of one or more degrees of freedom (DOF). For the clarity of the presentation, we assume a single DOF for the input signal (multiple dimensions will be discussed in Sect. 4.3).

The task of frequency and waveform learning is split into two separate tasks, each performed by a separate dynamical system. The frequency adaptation is performed by the *Canonical Dynamical System*, which consists of several adaptive frequency oscillators in a feedback structure. Its purpose is to extract the fundamental frequency Ω of the input signal, and to provide the phase Φ of the oscillator at this frequency.

These quantities are fed into the *Output Dynamical System*, whose goal is to adapt the shape of the limit cycle of the Canonical Dynamical System, and to learn the waveform of the input signal. The resulting output signal of the

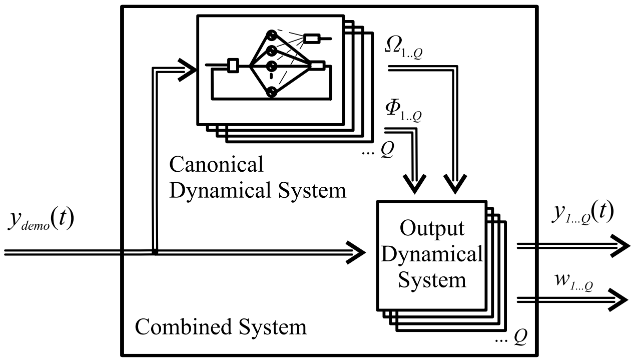

Fig. 1 Proposed structure of the system. The two-layer system is composed of the *Canonical Dynamical System* as the first layer for the frequency adaptation, and the *Output Dynamical System* for the learning as the second layer. The input signal $y_{demo}(t)$ is an arbitrary Q-dimensional periodic signal. The Canonical Dynamical System outputs the fundamental frequency Ω and phase of the oscillator at that frequency, Φ, for each of the Q DOF, and the Output Dynamical System learns the waveform

Output Dynamical System is not explicitly encoded but generated during the time evolution of the Canonical Dynamical System, by using a set of weights learned by Incremental Locally Weighted Regression (ILWR) (Schaal and Atkeson 1998).

The Output Dynamical System encapsulates several interesting properties, such as the reproduction of the trajectories, their modulation, and dealing with perturbations in a single set of differential equations.

Both frequency adaptation and waveform learning work in parallel, thus accelerating the process. The output of the combined system can be, for example, joint coordinates of the robot, position in task space, joint torques, etc., depending on what the input signal represents. Another output of the system is the weight vector w_i, which we can use to reproduce the learned trajectories at a desired frequency (see Sect. 4), or to classify the trajectories (see Sect. 5.5).

3.2 Canonical dynamical system

The task of the Canonical Dynamical System is two-fold. Firstly, it has to extract the fundamental frequency Ω of the input signal, and secondly, it has to exhibit stable limit cycle behavior in order to provide a phase signal Φ, that is used to anchor the waveform of the output signal.

As the basis of our canonical dynamical system we use a set of phase oscillators, see e.g. Buchli et al. (2006), to which we apply the adaptive frequency learning rule as introduced in Buchli and Ijspeert (2004) and Righetti et al. (2006), and combine it with a feedback structure (Righetti and Ijspeert 2006) shown in Fig. 2. The basic idea of the structure is that each of the oscillators will adapt its frequency to one of the frequency components of the input signal, essentially "populating" the frequency spectrum.

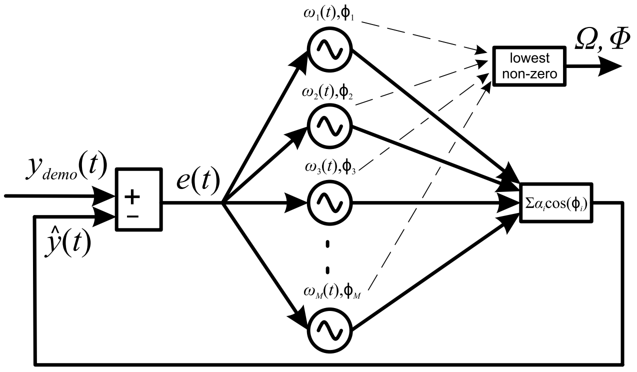

Fig. 2 Feedback structure of a network of adaptive frequency phase oscillators, that form the Canonical Dynamical System. All oscillators receive the same input and have to be at different starting frequencies to converge to different final frequencies. Refer also to text and (1–5)

We use several oscillators, but are interested only in the fundamental or lowest non-zero frequency of the input signal, denoted by Ω, and the phase of the oscillator at this frequency, denoted by Φ. Therefore the feedback structure is followed by a small logical block, which chooses the correct, lowest non-zero, frequency. Determining Ω and Φ is important because with them we can formulate a supervised learning problem in the second stage—the Output Dynamical System, and learn the waveform of the full period of the input signal.

The feedback structure of M adaptive frequency phase oscillators is governed by the following equations:

$$\dot{\phi}_i = \omega_i - Ke(t)\sin(\phi_i), \tag{1}$$

$$\dot{\omega}_i = -Ke(t)\sin(\phi_i), \tag{2}$$

$$e(t) = y_{demo}(t) - \hat{y}(t), \tag{3}$$

$$\hat{y}(t) = \sum_{i=1}^{M} \alpha_i \cos(\phi_i), \tag{4}$$

$$\dot{\alpha}_i = \eta\cos(\phi_i)e(t), \tag{5}$$

where K is the coupling strength, ϕ_i is the phase of oscillator i, $e(t)$ is the input into the oscillators, $y_{demo}(t)$ is the input signal, $\hat{y}(t)$ is the weighted sum of the oscillators' outputs, M is the number of oscillators, α_i is the amplitude associated to the i-th oscillator, and η is a learning constant.

Equations (1) and (2) present the core of the Canonical Dynamical System—the adaptive frequency phase oscillator. Several (M) such oscillators are used in a feedback loop to extract separate frequency components. Equations (3) and (4) specify the feedback loop, which needs also amplitude adaptation for each of the frequency components (5).

As we can see in Fig. 2, each of the oscillators of the structure receives the same input signal, which is the difference between the signal to be learned and the signal already learned by the feedback loop, as in (3). Since a negative feedback loop is used, this difference approaches zero as the weighted sum of separate frequency components, (4), approaches the learned signal, and therefore the frequencies of the oscillators stabilize. Equation (5) ensures amplitude adaptation and thus the stabilization of the learned frequency. Such a feedback structure performs a kind of dynamic Fourier analysis. It can learn several frequency components of the input signal (Righetti and Ijspeert 2006) and enables the frequency of a given oscillator to converge as $t \to \infty$, because once the frequency of a separate oscillator is set, it is deducted from the demonstration signal $y_{demo}(t)$, and disappears from $e(t)$ (due to the negative feedback loop). Other oscillators can thus adapt to other remaining frequency components.

The populating of the frequency spectrum is therefore done without any signal processing, as the whole process of frequency extraction and adaptation is totally embedded into the dynamics of the adaptive frequency oscillators.

Frequency adaptation results for a time-varying signal are illustrated in Fig. 3. The top plot shows the signal and the bottom plot shows the frequency adaptation to the signal. The signal itself is of three parts, a non-stationary signal (presented by a chirp signal), followed by a step change in the frequency of the signal, and in the end a stationary signal. We can see that the output frequency stabilizes very quickly at the (changing) target frequency. In general the speed of convergence depends on the coupling strength K (see Fig. 5) (Righetti and Ijspeert 2006). Besides the use for non-stationary signals, such as chirp signals, coping with the change in frequency of the input signal proves especially useful when adapting to the frequency of hand-generated signals, which are never stationary.

In this particular example, a single adaptive frequency oscillator in a feedback loop was enough, because the input signal was purely sinusoidal. In the next sections we will use a common fixed number of oscillators M for all the signals.

Figure 4 shows the results of the Canonical Dynamical System adapting to an input signal with five frequency components; five oscillators were used. The oscillators tend to adapt to the components they are closest to, as they fall into their basins of attraction. These are different in size, depending on the power of separate components and the coupling strength K. To accurately learn the frequencies we have to include enough oscillators in a feedback loop to "eliminate" all the signals with (3). If we use fewer oscillators than there are frequency components of the signal, their frequency will oscillate with an amplitude of oscillations dependent on K (Righetti and Ijspeert 2006). We can still extract the exact frequency by using a mean value of the chosen frequency.

The number of adaptive frequency oscillators in a feedback loop is therefore a matter of design. There should be enough oscillators to avoid missing the fundamental frequency and to limit the variation of frequencies described

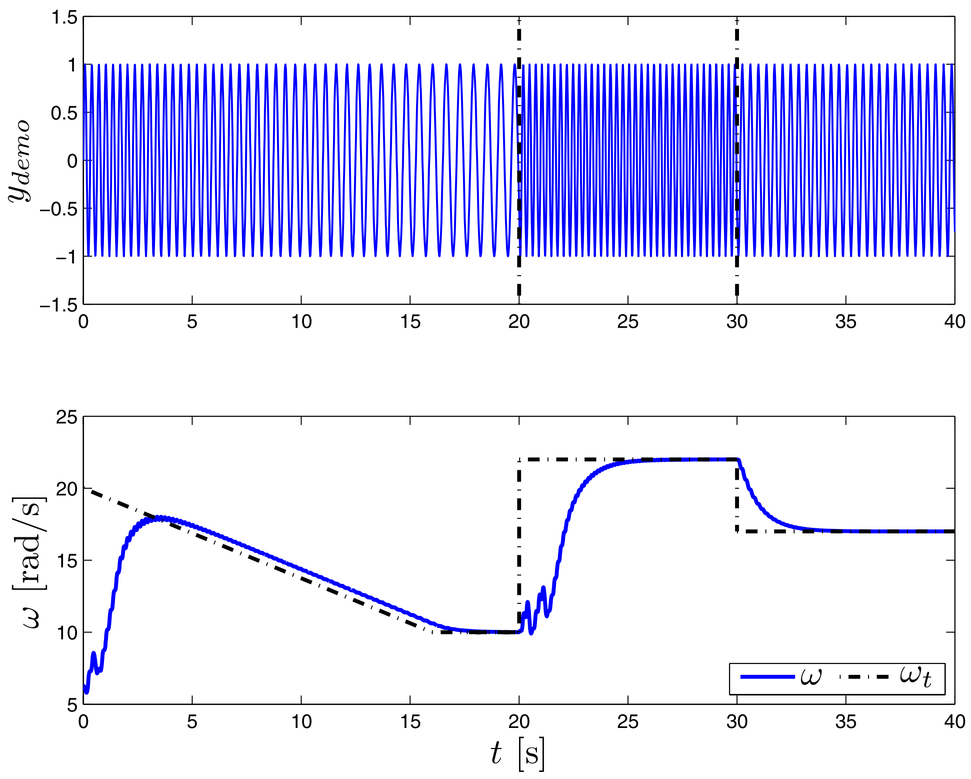

Fig. 3 *Top*: Non-stationary input signal: chirp signal $y_{demo} = \sin(\omega_t t)$ with $\omega_t = (20 - 0.625t)$ rad/s for $t < 16$ s, and $\omega_t = 10$ rad/s for $t > 16$ s. Step changes follow with $\omega_t = 22$ rad/s and 17 rad/s, occurring at $t = 20$ s and $t = 30$ s. Input signal stabilizes at 17 rad/s. *Bottom*: frequency adaptation of a feedback structure with one adaptive frequency oscillator, starting at $\omega_0 = 2\pi$ rad/s. ω is the *solid line* and ω_t is the *dash-dot line*. As we can see, the adaptation is successful for non-stationary signals, step changes and stationary signals

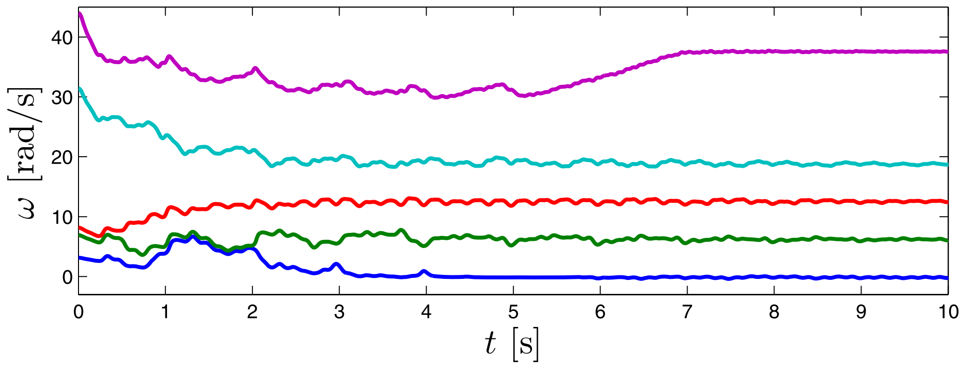

Fig. 4 Frequency adaptation of the feedback structure with five adaptive phase oscillators to the input signal $y_{demo}(t) = 0.5 + 0.8\sin(2\pi t) + 2\sin(4\pi t) + \sin(6\pi t) + \sin(12\pi t)$, with the initial conditions of the adaptive frequency oscillators at $\omega_0 = [3.14, 6.59, 8.17, 31.41, 43.98]$ rad/s. The canonical system successfully adapts to all five frequency components—the offset (or 0 rad/s), 2π rad/s, 4π rad/s, 6π rad/s and 12π rad/s

above when the input signal has many frequencies components. A high number of oscillators can easily be used, as this does not affect the speed of convergence. Beside the almost negligible computational costs, using too many oscillators does not affect the solution. A practical problem that arises is that the oscillators' frequencies might come too close together, and then lock onto the same frequency component. To solve this we separate their initial frequencies ω_0 in a manner that suggests that (preferably only) one oscillator will go for the offset, one will go for the highest frequency, and the others will "stay between" (see Fig. 4).

With a high number of oscillators, many of them want to lock to the offset (0 Hz). With the target frequency under 1 rad/s the oscillations of the estimated frequency tend to be higher, which results in longer adaptation times. This makes choosing the fundamental frequency without introducing complex decision-making logic difficult. We decided on using $M = 5$ oscillators (unless specified otherwise), keeping the number of oscillators low and allowing us to learn frequencies of five frequency components, one being the offset of the signal. The problem of choosing the correct frequency during normal usage of the system is thus kept at a minimum complexity level.

Other than using logic to choose the lowest non-zero frequency (absolute), we can also solve the problem of having frequency components in a ratio that cannot be described with an integer number, such as 4:3. In this case the fundamental frequency Ω is 1, which is the lowest common denominator of 3 and 4 (this will be discussed in more detail in Sect. 4.3).

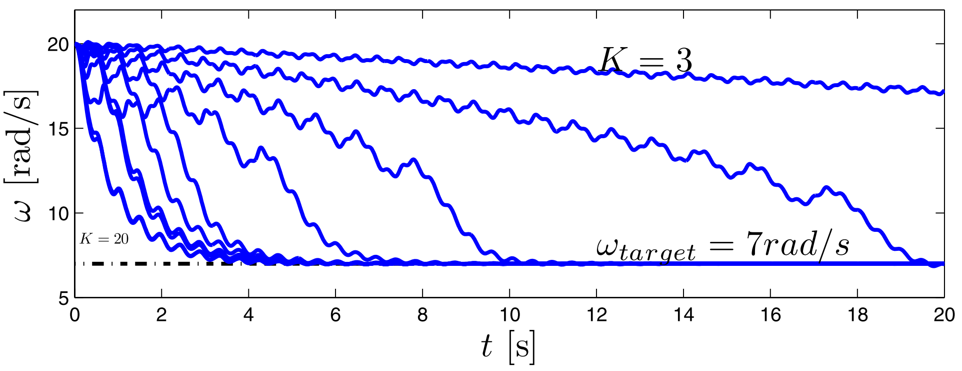

Fig. 5 Adaptation of a single adaptive frequency oscillator from the initial frequency $\omega_0 = 20$ rad/s to the target frequency $\omega_t = 7$ rad/s with the changing of the coupling strength K from $K = 3$ (highest or most right plot) to $K = 20$ (lowest or most left plot) with a step of 2 (final step is 1)

Besides learning, we can also use the system to repeat already learned signals. It this case, we cut feedback to the adaptive frequency oscillators by setting $e(t) = 0$. This way the oscillators continue to oscillate at the frequency to which they adapted. We are only interested in the fundamental frequency, determined by

$$\dot{\Phi} = \Omega, \tag{6}$$

$$\dot{\Omega} = 0, \tag{7}$$

which is derived from (1 and 2). This is also the equation of a normal phase oscillator.

Parameters of the Canonical Dynamical System
We want the system to adapt to the frequency of the input signal as fast as possible, or at least very fast. The speed of convergence depends on the parameter K in (1 and 2), see (Righetti et al. 2006) for details. Figure 5 presents the results for the coupling strength increasing from 3 to 20.

Another issue that has to be taken into consideration is the ratio between the coupling strength K and the learning constant η, which currently requires manual tuning. Throughout the paper we use $K = 20$ and $\eta = 1$, unless specified otherwise.

3.3 Output dynamical system

The output dynamical system is used to learn the waveform of the input signal. Again the explanation is for a 1 DOF signal. For multiple DOF, the algorithm works in parallel for all the degrees of freedom.

The following dynamics specify the attractor landscape of a trajectory y towards the anchor point g, with the Canonical Dynamical System providing the phase Φ to the function Ψ_i of the control policy:

$$\dot{z} = \Omega\left(\alpha_z\left(\beta_z\left(g - y\right) - z\right) + \frac{\sum_{i=1}^{N}\Psi_i w_i r}{\sum_{i=1}^{N}\Psi_i}\right), \tag{8}$$

$$\dot{y} = \Omega z, \tag{9}$$

$$\Psi_i = \exp\left(h\left(\cos(\Phi - c_i) - 1\right)\right). \tag{10}$$

Here Ω is the frequency given by Canonical Dynamical System, (2), α_Z and β_z are positive constants, set to $\alpha_z = 8$ and $\beta_z = 2$ for all the results; the ratio 4:1 ensures critical damping so that the system monotonically varies to the trajectory oscillating around g—an anchor point for the oscillatory trajectory. N is the number of Gaussian-like periodic kernel functions Ψ_i, which are given by (10). w_i is the learned weight parameter (below) and r is the amplitude control parameter, maintaining the amplitude of the demonstration signal with $r = 1$. The system given by (8) without the nonlinear term is a second-order linear system with a unique globally stable point attractor (Ijspeert et al. 2002b). But because of the periodic nonlinear term, this system produces stable periodic trajectories whose frequency is Ω and whose waveform is determined by the weight parameters w_i.

In (10), which determines the Gaussian-like kernel functions Ψ_i, h determines their width, which is set to $h = 2.5N$ for all the results presented in the paper unless stated otherwise, and c_i are equally spaced between 0 and 2π in N steps.

As the input into the learning algorithm we use triplets of position, velocity and acceleration $y_{demo}(t)$, $\dot{y}_{demo}(t)$, and $\ddot{y}_{demo}(t)$ with *demo* marking the input or demonstration trajectory we are trying to learn. With this (8) can be rewritten as

$$\frac{1}{\Omega}\dot{z} - \alpha_z\left(\beta_z\left(g - y\right) - z\right) = \frac{\sum_{i=1}^{N}\Psi_i w_i r}{\sum_{i=1}^{N}\Psi_i} \tag{11}$$

and formulated as a supervised learning problem with on the right hand side a set of local models $w_i r$ that are weighted by the kernel functions Ψ_i, and on the left hand side the target function f_{targ} given by $f_{targ} = \frac{1}{\Omega^2}\ddot{y}_{demo} - \alpha_z(\beta_z(g - y_{demo}) - \frac{1}{\Omega}\dot{y}_{demo})$, which is obtained by matching y to y_{demo}, z to $\frac{\dot{y}_{demo}}{\Omega}$, and $\dot{z}$ to $\frac{\ddot{y}_{demo}}{\Omega}$.

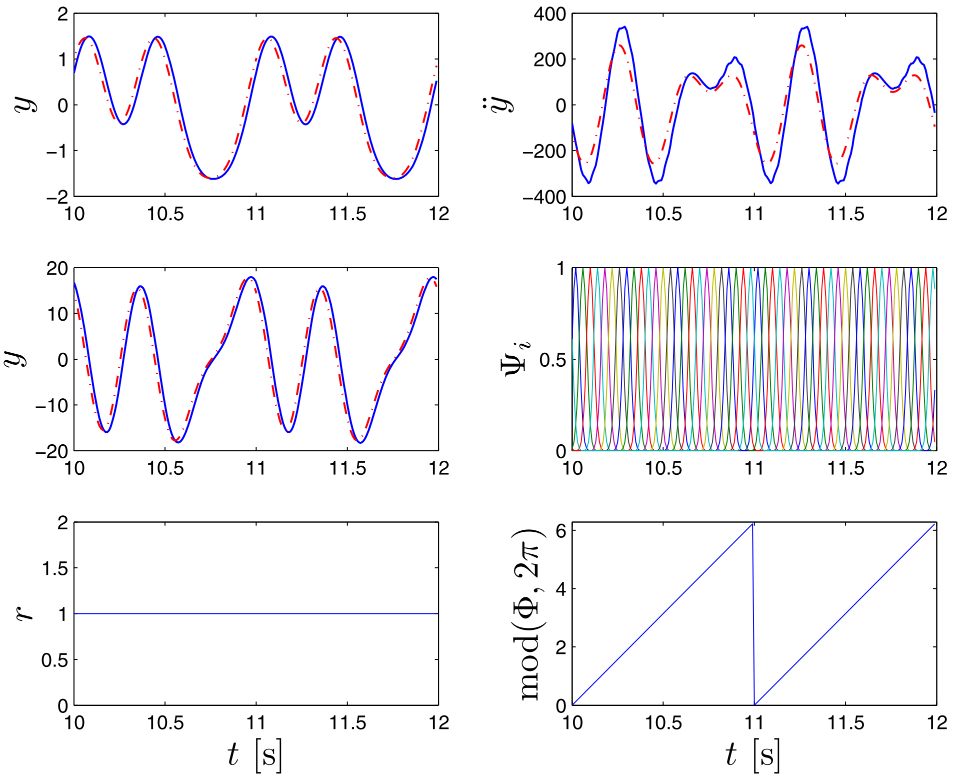

Fig. 6 The result of Output Dynamical System with a constant frequency input and with continuous learning of the weights. In *all the plots* the input signal is the *dash-dot line* while the learned signal is the *solid line*. In the *middle-right plot* we can see the evolution of the kernel functions. The kernel functions are a function of Φ and do not necessarily change uniformly (see also Fig. 13). In the *bottom right plot* the phase of the oscillator is shown. The amplitude is here $r = 1$, as shown *bottom-left*

Locally weighted regression corresponds to finding, for each kernel function Ψ_i, the weight vector w_i, which minimizes the quadratic error criterion[2]

$$J_i = \sum_{t=1}^{P} \Psi_i(t) \left(f_{targ}(t) - w_i r(t) \right)^2 \tag{12}$$

where t is an index corresponding to discrete time steps (of the integration). The regression can be performed as a *batch* regression, or alternatively, we can perform the minimization of the J_i cost function incrementally, while the target data points $f_{targ}(t)$ arrive. As we want continuous learning of the demonstration signal, we use the latter. Incremental regression is done with the use of recursive least squares with a forgetting factor of λ, to determine the parameters (or weights) w_i. Given the target data $f_{targ}(t)$ and $r(t)$, w_i is updated by

$$w_i(t+1) = w_i(t) + \Psi_i P_i(t+1) r(t) e_r(t), \tag{13}$$

$$P_i(t+1) = \frac{1}{\lambda} \left(P_i(t) - \frac{P_i(t)^2 r(t)^2}{\frac{\lambda}{\Psi_i} + P_i(t) r(t)^2} \right), \tag{14}$$

$$e_r(t) = f_{targ}(t) - w_i(t) r(t). \tag{15}$$

P, in general, is the inverse covariance matrix (Ljung and Söderström 1986). The recursion is started with $w_i = 0$ and $P_i = 1$. Batch and incremental learning regressions provide identical weights w_i for the same training sets when the forgetting factor λ is set to one. Differences appear when the forgetting factor is less than one, in which case the incremental regression gives more weight to recent data (i.e. tends to forget older ones). The error of weight learning e_r (15) is not "related" to e when extracting frequency components (3). This allows for complete separation of frequency adaptation and waveform learning.

Figure 6 shows the time evolution of the Output Dynamical System anchored to a Canonical Dynamical System with the frequency set at $\Omega = 2\pi$ rad/s, and the weight parameters w_i adjusted to fit the trajectory $y_{demo}(t) = \sin(2\pi t) + \cos(4\pi t) + 0.4\sin(6\pi t)$. As we can see in the top-left plot, the input signal and the reconstructed signal match closely. The matching between the reconstructed signal and the input signal can be improved by increasing the number of Gaussian-like functions, as is explained in the next paragraphs.

Parameters of the Output Dynamical System

When tuning the parameters of the Output Dynamical System, we have to determine the number of Gaussian-like Kernel functions N, and specially the forgetting factor λ.

[2] LWR is derived from a piecewise linear function approximation approach (Schaal and Atkeson 1998), which decouples a nonlinear least-squares learning problem into several locally linear learning problems, each characterized by the local cost function J_i. These local problems can be solved with standard weighted least squares approaches.

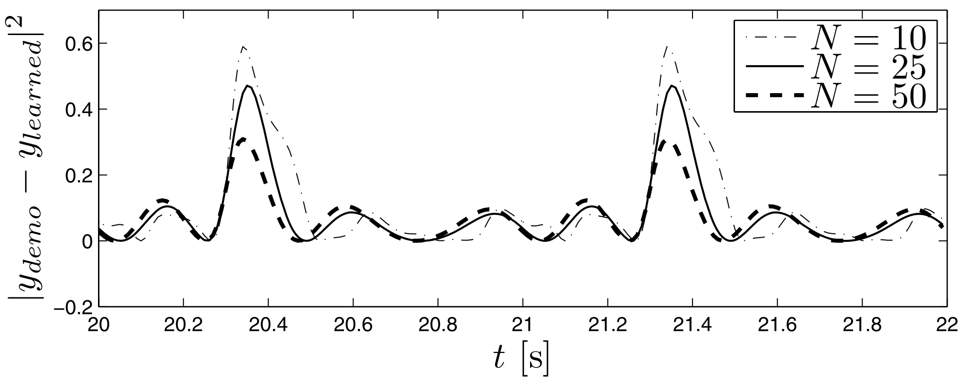

Fig. 7 The error of learning decreases with the increase of the number of Gaussian-like kernel functions. The error, which is quite small, is mainly due to a very slight (one or two sample times) delay of the learned signal

Table 1 Equations and parameters of the system

Canonical Dynamical System	Output Dynamical System
$\dot{\phi}_i = \omega_i - Ke(t)\sin(\phi_i)$, (1)	$\dot{z} = \Omega\left(\alpha_z(\beta_z(g-y)-z) + \frac{\sum_{i=1}^N \Psi_i w_i r}{\sum_{i=1}^N \Psi_i}\right)$, (8)
$\dot{\omega}_i = -Ke(t)\sin(\phi_i)$, (2)	$\dot{y} = \Omega z$, (9)
$e(t) = y_{demo}(t) - \hat{y}(t)$	$\Psi_i = \exp(h(\cos(\phi - c_i) - 1))$, (10)
$\hat{y}(t) = \sum_{i=0}^M \alpha_i \cos(\phi_i)$, (4)	
$\dot{\alpha}_i = \eta \cos(\phi_i) e(t)$, (5)	
$K = 20$	$\alpha_z = 8$
$M = 5$	$\beta_z = 2$
$\eta = 1$	$N = 25$
	$h = 2.5N$, $c \in [0, 2\pi]$
	$\lambda = 0.95$, (14)

The number N of Gaussian-like kernel functions could be set automatically if we used the locally weighted learning (Schaal and Atkeson 1998), but for simplicity it was here set by hand. Increasing the number increases the accuracy of the reconstructed signal, but at the same time also increases the computational cost. Note that LWR does not suffer from problems of overfitting when the number of kernel functions is increased.[3] Figure 7 shows the error of learning e_r when using $N = 10$, $N = 25$, and $N = 50$ on a signal $y_{demo}(t) = 0.65\sin(2\pi t) + 1.5\cos(4\pi t) + 0.3\sin(6\pi t)$. Throughout the paper, unless specified otherwise, $N = 25$.

The forgetting factor $\lambda \in [0, 1]$ plays a key role in the behavior of the system. If it is set high, the system never forgets any input values and learns an average of the waveform over multiple periods. If it is set too low, it forgets all, basically training all the weights to the last value.

Unless stated otherwise, the parameters presented in Table 1 are used in this article. Besides the fixed parameters, the system also includes three control parameters that can be used to modulate trajectories *during replay*. These are g for the anchor point of oscillations (see (8)), r for the amplitude, and Ω for the frequency (see Sect. 4.1).

3.4 Combined system

In this section we present the behavior of the combined system with both frequency adaptation and waveform learning. Figure 8 shows the results for the case when the Canonical Dynamical System is still adapting to the frequency of the input signal.

Frequency adaptation affects the output of the combined system through both the frequency Ω and the phase Φ. The frequency Ω, while it is not completely adapted, affects the target trajectory, see (11). The phase Φ, while the frequency is changing, leads to an incorrect mapping of the input signal to the weight vector of the kernel functions. In order to learn the waveform of only one period of the demonstration signal, the frequency has to be adapted. Depending on the initial condition of the oscillator, coupling strength K, and complexity of the signal, the time it takes for the Canonical Dynamical System to adapt to the frequency of the input signal varies. In our case, and this applies to the general

[3]This property is due to solving the bias-variance dilemma of function approximation locally with a closed form solution to leave-one-out cross-validation (Schaal and Atkeson 1998).

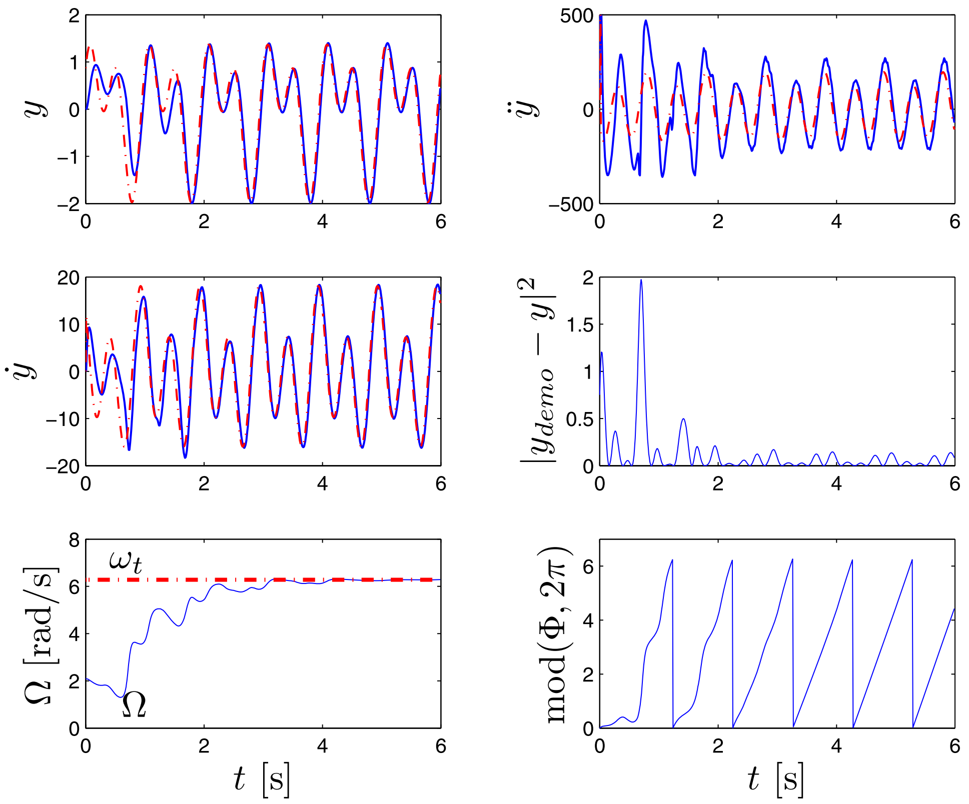

Fig. 8 The result of learning combined with frequency adaptation. Again the input signal is the *dash-dot line*. The *middle-right plot* shows the square of the difference of the input and the learned signal. The *bottom left plot* shows the frequency of the adaptive phase oscillator, for clarity reasons only the fundamental frequency is shown. 5 oscillators were used in the canonical dynamical system even though the input signal has only 2 frequency components. The other determined frequencies are 4π rad/s (the other component of the signal), and 22.994 rad/s (random value of the frequency that an oscillator was at when $e(t) = 0$, see (1)). Two oscillators converged to the same frequency for 2π and 4π rad/s

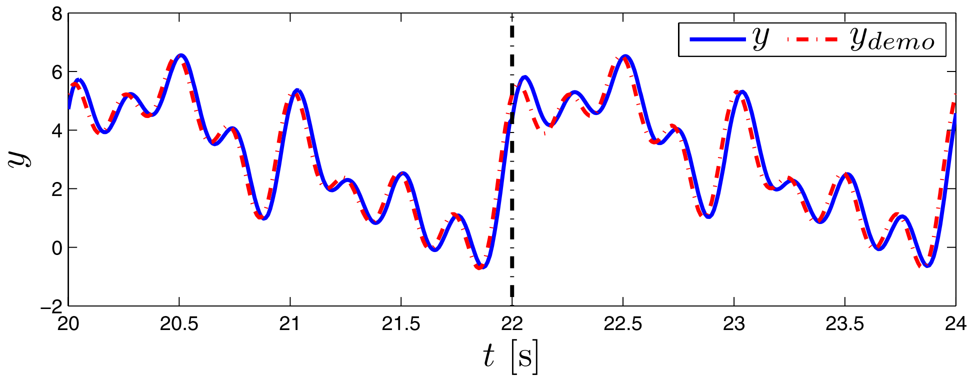

Fig. 9 Results of learning a complex signal $y_{demo} = 3 + 2\sin(\pi t) + \sin(2\pi t) + \sin(4\pi t + \pi/3) + 0.5\sin(6\pi t) + \cos(8\pi t)$ with 6 frequency components and the fundamental frequency $\Omega = \pi$ rad/s extracted using $M = 5$ oscillators. The *vertical dash-dot line* marks the transition between learning and pure reproduction. Since we used fewer oscillators than there are frequency components in the input signal, the frequency was oscillating slightly. At the time of the switching between learning–reproducing, it was not exactly at π, thus the slight delay of the signal. When repeating the signal, the fundamental frequency is one of the control parameters and can be freely changed

case as well, we make sure this time is very short with a high value of K. The learning of the weights takes place even as the frequency is still adapting. Once the adaptation of the frequency is complete, the weights (due to the forgetting factor λ) very rapidly take their final value. The outcome can be seen in the start of the plots in Fig. 8. We can see that the frequency roughly stabilizes within 3 seconds (bottom-left plot), leading to a linear increase of the phase Φ (bottom-right). The matching of the input and output signal (top-left) is even faster as a result of the forgetting factor.

Figure 9 shows the results for a more complex signal, this time with 6 frequency components. The input signal is the dash-dot line and the output signal is the solid line. The vertical dash-dot line marks the transition between learning and pure reproduction. As we can see, the input and output signal are almost identical for both cases; a slight shift is present because of difference in frequencies. This is because

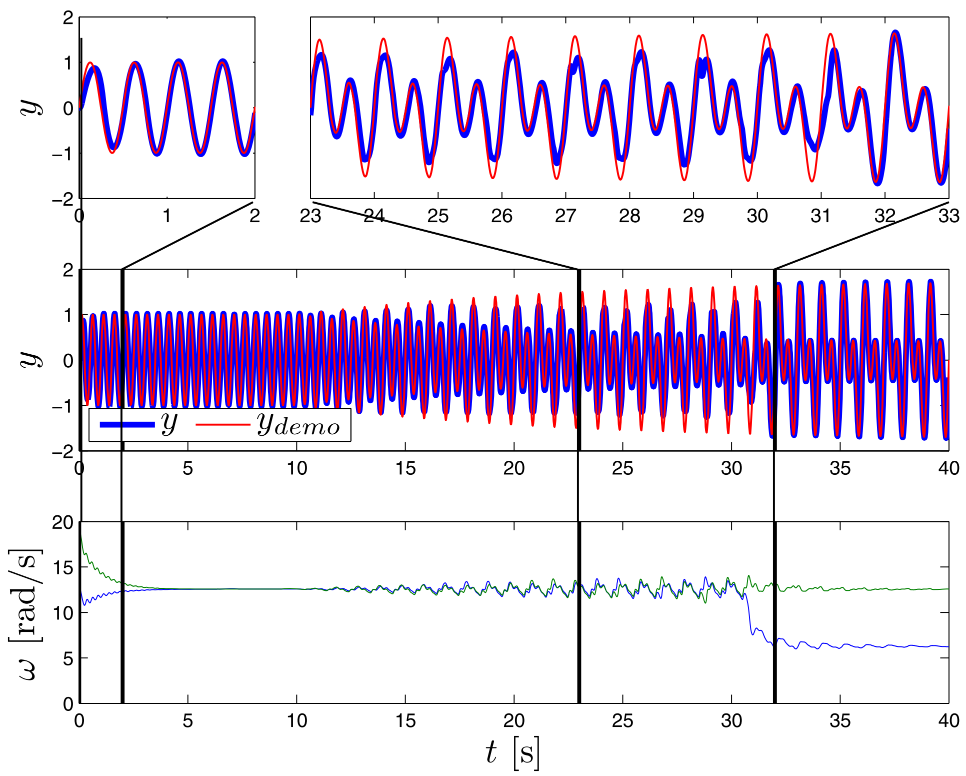

Fig. 10 The *middle plot* shows the input signal y_{demo}. The *bottom plot* shows the frequency adaptation. The *top plots* show close-ups of the interesting parts. Refer also to the text

the fundamental frequency was determined using $M = 5$ oscillators, therefore the adapted frequencies oscillate slightly as there are fewer oscillators than there are frequency components in the input signal.

An interesting aspect of our system is also the ability to deal with non-stationary signals and to adapt to changes of the input signal both in waveform and frequency. Figure 10 presents the results of a system adapting to a signal where the waveform of the input signal introduces a frequency doubling as it changes from $y_{demo}(t) = \sin(4\pi t)$ to $y_{demo}(t) = \sin(4\pi t) + (1 - e^{0.085t})\sin(2\pi t)$. As we can see, the learned signal has a double peak even before the Canonical Dynamical System has determined the frequency halving. This is a result of the forgetting factor $\lambda = 0.95$ and rapid weight learning. Once the oscillations of the frequency signal are big enough (bottom plot), one of the oscillators "jumps" into another basin of attraction, adapting to a new frequency (2π), and the system then takes this frequency to be the fundamental Ω. We can see in the top right plot that the learned and the demonstration signal are almost identical after $t = 32$ s. A higher K makes the frequency "jump" happen faster, as it increases the oscillations of the estimated frequency. Determining that the frequency halved clearly demonstrates the ability of the system to adapt to the changing input signal.

The combination of frequency adaptation and waveform learning exploits the advantages of both systems. The biggest advantage of all is the fact that the frequency of the input signal does not have to be pre-determined. For the Output Dynamical system to work correctly, the frequency of the input signal has to be known. The combined system eliminates the need to somehow pre-determine the frequency and thus allows the system to be used in an almost black-box fashion, i.e., without manual intervention of a user.

Even though the canonical dynamical system by itself could reproduce the demonstration signal ($\hat{y}$, (4)), using the Output Dynamical system allows for easier modulation in both frequency and amplitude, learning of complex patterns without extracting all frequency components, is more robust to perturbations, and acts as a sort of a filter. Moreover, when multiple output signals need to be synchronized by the canonical system, only one canonical system is used as a "pace maker", and the individual output systems need to make sure that the waveforms of the different degrees-of-freedom is realized appropriately.

4 System properties

In this section we present the behavior of the system when modulating the learned trajectories, dealing with perturbations, and dealing with multiple DOFs in parallel. The described possibilities are the properties of both the Canonical Dynamical System and the Output Dynamical System. All the modulations are applied when repeating the signal and

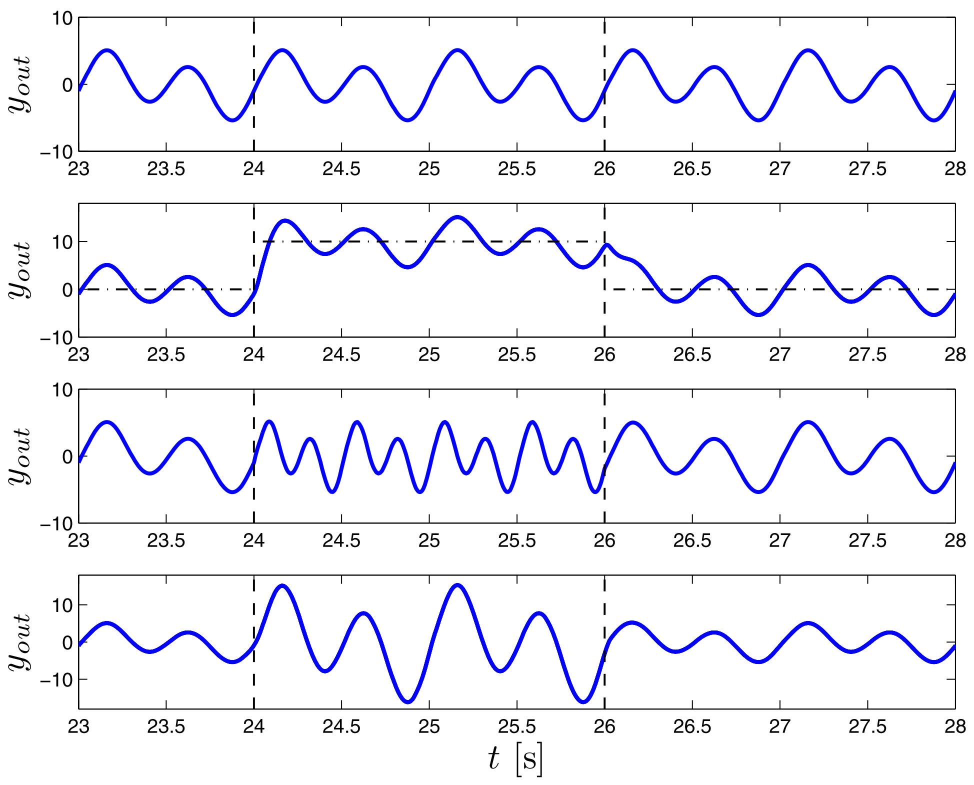

Fig. 11 Modulations of the learned signal. The learned signal (*top*), modulating the baseline for oscillations g (*second from top*), doubling the frequency Ω (*third from top*), doubling the amplitude r (*bottom*)

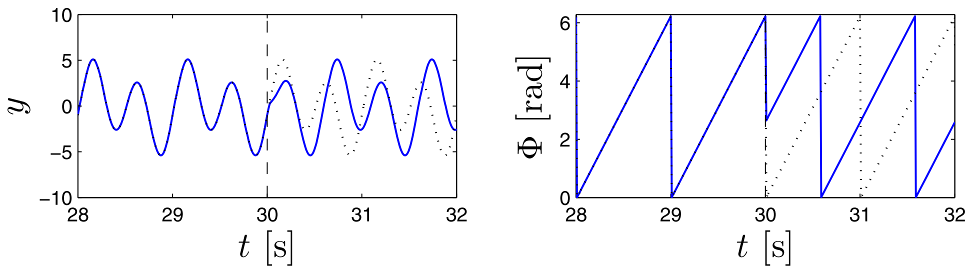

Fig. 12 Dealing with perturbations: Reacting to a random perturbation of the state variables y, $\dot{y}$ and Φ at $t = 30$ s

the Canonical Dynamical System is a phase oscillator (see (6)).

4.1 Modulations of learned trajectory

The system is designed to permit on-line modulations of the trajectories originally learned. This is one of the important motivations behind the use of dynamical systems to encode trajectories.

Changing the parameter g corresponds to a modulation of the baseline of the rhythmic movement. This will smoothly shift the oscillation without modifying the signal shape. The results are presented in the second plot in Fig. 11. Modifying Ω and r corresponds to the changing of the frequency and the amplitude of the oscillations, respectively. Since our differential equations are of second order, these abrupt changes of parameters result in smooth variations of the trajectory y. This is particularly useful when controlling articulated robots, which require trajectories with limited jerks. Changing of the parameter Ω only comes into consideration when one wants to repeat the learned signal at a desired frequency that is different from the one we adapted to with our Canonical Dynamical System. Results of changing the frequency Ω are presented in the third plot of Fig. 11. Results of modulating the amplitude parameter r are presented in the bottom plot of Fig. 11.

4.2 Dealing with perturbations

The Output Dynamical System is inherently robust against perturbations. Figure 12 illustrates the time evolution of the system repeating a learned trajectory at the frequency of 1 Hz, when the state variables y, z and Φ are randomly changed at time $t = 30$ s. From the results we can see that the output of the system reverts smoothly to the learned trajectory. This is an important feature of the approach: the sys-

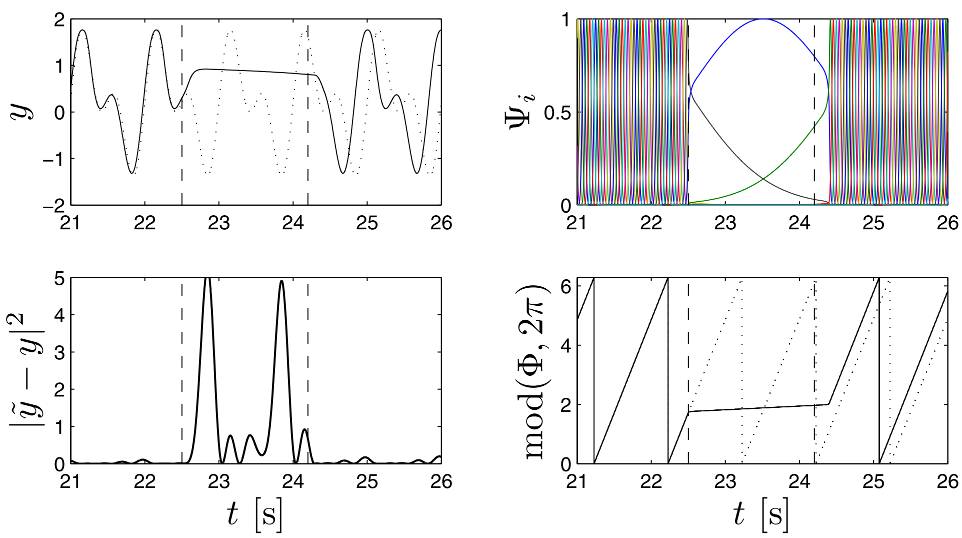

Fig. 13 Reacting to a perturbation with a slow-down feedback. The desired position y and the actual position $\tilde{y}$ are the same except for the short interval between $t = 22.2$ s and $t = 23.9$ s. The *dotted line* corresponds to the original unperturbed trajectory

tem essentially represents a whole landscape in the space of state variables which not only encode the learned trajectory but also determine how the states return to it after a perturbation.

When controlling the robot, we have to take into account perturbations due to the interaction with the environment. Our system provides *desired* states to the robot, i.e. desired joint angles or torques, and its state variables are therefore not affected by the *actual* states of the robot, unless feedback terms are added to the control scheme. For instance, it might happen that, due to external forces, significant differences arise between the actual position $\tilde{y}$ and the desired position y. Depending on the task, this error can be fed back to the system in order to modify on-line the generated trajectories.

One type of such feedback is the "slow-down-feedback" that can be applied to the Output Dynamical System. This type of feedback affects both the Canonical and the Output Dynamical System. The following explanation is for the replay of a learned trajectory as perturbing the robot while learning the trajectory is not practical.

For the process of repeating the signal, for which we use a phase oscillator, we modify (9 and 6) to:

$$\dot{y} = \Omega\left(z + \alpha_{py}\left(\tilde{y} - y\right)\right), \tag{16}$$

$$\dot{\Phi} = \frac{\Omega}{1 + \alpha_{p\Phi}|\tilde{y} - y|}, \tag{17}$$

where α_{py} and $\alpha_{p\Phi}$ are positive constants.

With this type of feedback, the time evolution of the states is gradually halted during the perturbation. The desired position y is modified to remain close to the actual position $\tilde{y}$, and as soon as the perturbation stops, rapidly resumes performing the time-delayed planned trajectory. Results are presented in Fig. 13. As we can see, the desired position y and the actual position $\tilde{y}$ are the same except for the short interval between $t = 22.2$ s and $t = 23.9$ s. The dotted line corresponds to the original unperturbed trajectory. The desired trajectory continues from the point of perturbation and does not jump to the unperturbed desired trajectory.

Another example of a perturbation can be the presence of boundaries or obstacles, such as joint angle limits. In that case we can modify (9) to include a repulsive force $l(y)$ at the limit by:

$$\dot{y} = \Omega\left(z + l(y)\right). \tag{18}$$

For instance, a simple repulsive force to avoid hitting joint angles or going beyond a position in task space can be

$$l(y) = -\gamma \frac{1}{(y_L - y)^3} \tag{19}$$

where y_L is the value of the limit. Figure 14 illustrates the effect of such a repulsive force.

Such on-line modifications are one of the most interesting properties of using autonomous differential equations for control policies. These are just examples of possible feedback loops, and they should be adjusted depending on the task at hand.

4.3 Multi-dimensionality

As indicated in Fig. 1, the system can be expanded to several dimensions, for instance to control various joints on a humanoid robot. Basically, the presented system can work in parallel for each of the given DOF of the signal with all the processes working in parallel for each DOF. Figure 15 presents the results of frequency and waveform adaptation for a 2-DOF signal.

When applying the system to more than one DOF, we have to consider the following scenarios: the trajectories of

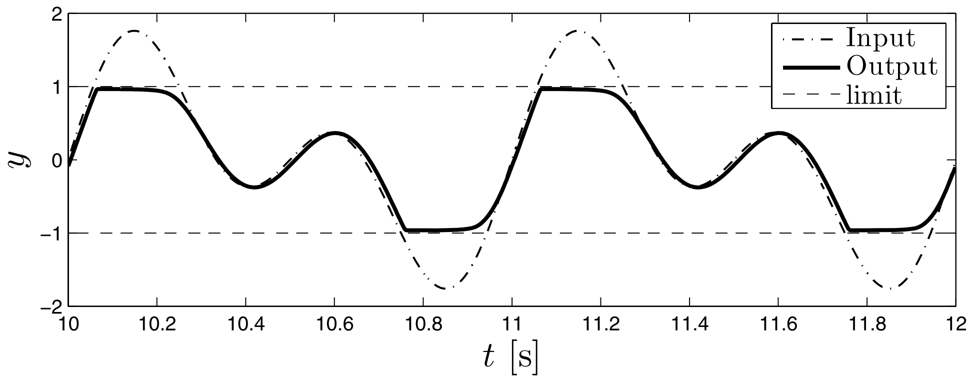

Fig. 14 Output of the system with the limits set to $y_l = [-1, 1]$ for the input signal $y_{demo}(t) = \cos(2\pi t) + \sin(4\pi t)$

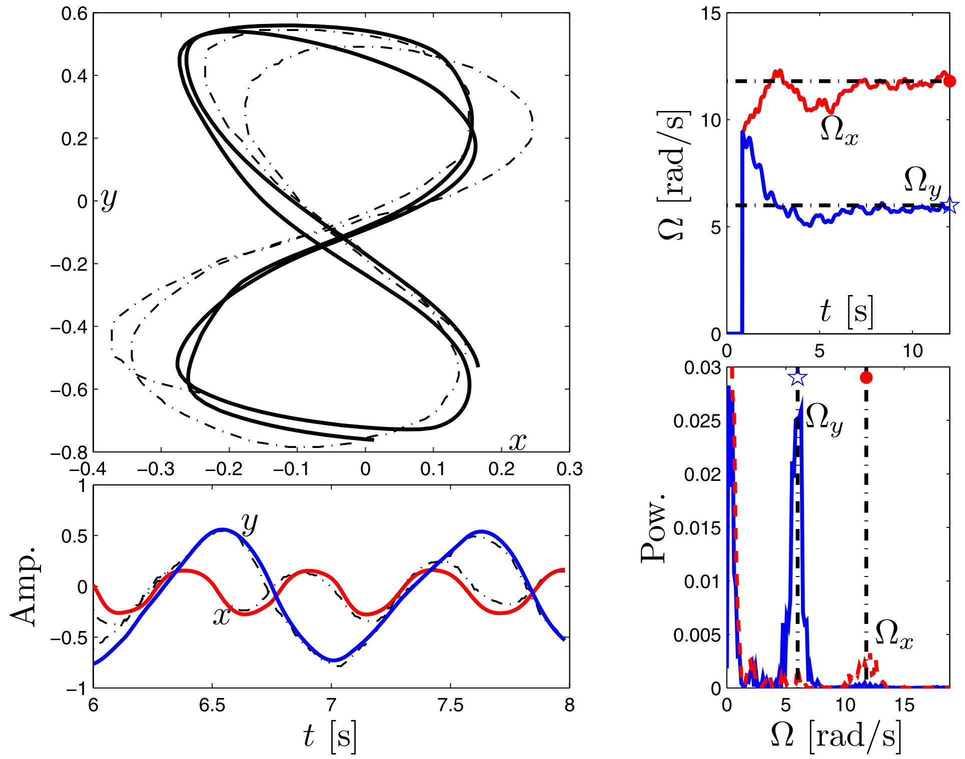

Fig. 15 Results for hand-generated signal with constant learning and frequency adaptation. The *top left-plot* shows two seconds of the input (*dash-dot line*) and the learned output signal (*solid*) in 2D. As the signal is constantly adapting, the reproduced trajectory is not stationary. The *bottom-left plot* shows 2 s of the input signals (*dash-dot*) and the output signals on a common axis. The *top-right plot* shows the learned frequencies. The *bottom-right plot* shows the power spectrum for both dimensions of the input signal. Notice that the system has found that $\Omega_x \cong 2\Omega_y$

separate DOF are frequency locked (i.e. there is a rational ratio between all DOFs), or they are not. Frequency-locked trajectories would typically be produced by a system in which the various DOFs are coupled (e.g. the human musculo-skeletal system). When recording the motion of the human arm performing a repetitive task in 2-D space, e.g. moving a computer mouse in the shape of a figure-8, it is obvious that the motion is frequency locked, and also has a common "clock", as presented in the left plot of Fig. 17. The opposite case is using separate "clocks" for separate DOF, presented in the middle plot of Fig. 17.

When applying our system in parallel for both frequency and waveform learning, the results of the frequency learning for separate DOF will not return exactly the same value for the frequency, leading to drift of the signals when repeating.

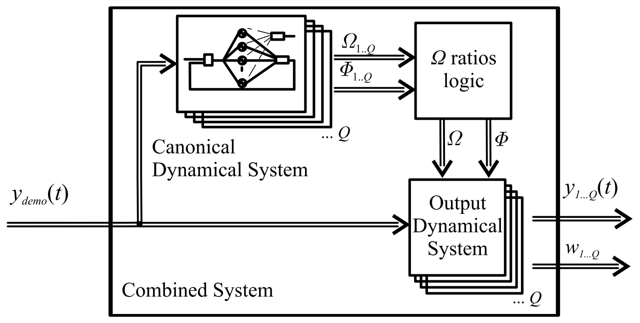

Fig. 16 Modified structure of the system with an introduced logical block to determine frequency ratios

Also, the frequencies can be multiples of each other rather than the same values (Fig. 15). While we do on-line learning,

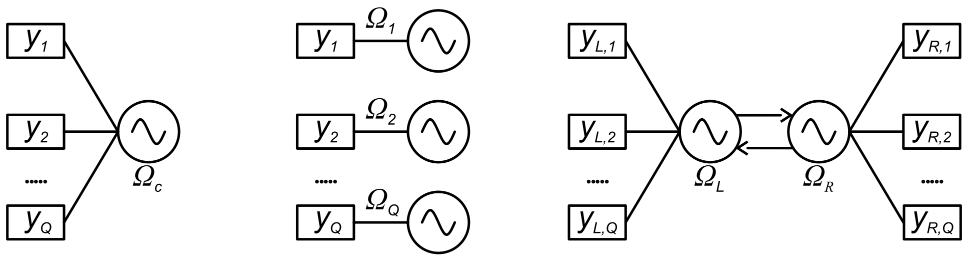

Fig. 17 Using a common clock for all the DOF (*left*). Using different clocks for different DOF or actuated systems (*center*). Using two coupled clocks for two multi-DOF systems (*right*)

this is constantly corrected for by the learning and the frequency adaptation, and does not pose a problem. When repeating the signal, on the other hand, this has to be corrected if we want to avoid drift. There are different approaches to solving this problem.

Using a logical block. One of the options is to use simple logical operations to determine if separate degrees of the signals should have a common frequency, i.e. we can use the same frequency for both signals if they are within a certain interval or if their ratio is within a certain interval (e.g. rounding the ratio of two measured frequencies 1.01 Hz and 2.0 Hz to 2, and using a common fundamental frequency of 1.01 Hz). Additionally, a logical block can determine the common frequency when signals are in a ratio of, e.g. 4 vs. 3 (as within the Canonical Dynamical System for a single DOF), even though such an example might not be too common.

Using only one fundamental frequency Ω has to be considered not only in repeating the signal but also when learning, i.e. constantly changing the weight vector. Using only one frequency, e.g. 1 Hz for $\Omega_x = 1$ Hz and $\Omega_y = 2$ Hz, means that we have to stretch $y_{demo,y}(t)$ over two periods (but with the same number of kernel functions) resulting in slightly less accurate signal reconstruction. Figure 16 shows the control scheme for such an approach. This is the approach we use in all of the presented results unless stated otherwise.

A slight modification of using such a block is simple rounding of the ratio and using a separate Canonical Dynamical System for each of the DOF. The draw-back of this approach is that there is no common clock and that we need separate Canonical Dynamical Systems also when just repeating the signal. Additionally, we have to know the ratio when repeating the signal. On the other hand this approach allows changing the frequency ratio between separate DOF.

Adding the signals. One of the possible approaches is to add the signals of separate DOF, and to input the added signal into a single Canonical Dynamical System that can handle the number of frequency components of such a signal. This can have unwanted side-effects, such as doubling or canceling-out frequencies. Furthermore, since the signals are from the same system, noise in one DOF of the signal is almost sure to be present in the other-one as well. This causes higher oscillations of the estimated frequency. Nonetheless, for simple signals, such an approach works quite well.

Coupling the signals. This approach comes into consideration only when considering signals that are not just the DOFs of a common system. An example of such is drumming with both hands. Even though using a common clock for both hands would do, we can as well couple the two oscillators of separate arms and obtain a single resulting frequency due to synchronization. To do this, we edit the equation of separate phase oscillators given by (6) as follows:

$$\dot{\Phi}_{11} = \Omega_1 + b\sin(\Phi_2 - \Phi_1)\,, \tag{20}$$

$$\dot{\Phi}_2 = \Omega_2 + b\sin(\Phi_1 - \Phi_2)\,, \tag{21}$$

which is also a model of oscillatory behavior of biological systems (Crespi et al. 2005). This approach is only applicable when reproducing a learned signal and not while learning it.

For all the applications using the arms of the HOAP-2 robot we used a common clock for all of the DOF.

5 Experimental evaluation

In this section we present experimental evaluations of our system in different experiments.

5.1 Experimental setup

We tested the proposed system in learning of the frequency and waveform by demonstration with a humanoid robot HOAP-2. HOAP-2 is a humanoid robot constructed by Fujitsu, and has altogether 25 DOF, of which we used 8 DOF to control the movement of the arms.

To control the robot, we separated the interface and the dynamical systems on one side, and the actual control of the robot's motion on the other side on two computers, as is presented in Fig. 18.

The implementation of both dynamical systems on the interface side assumes a common time step for numerical integration. On the interface side we record the position of the mouse cursor (or some other input device) in Matlab/Simulink and send it to the computer controlling the

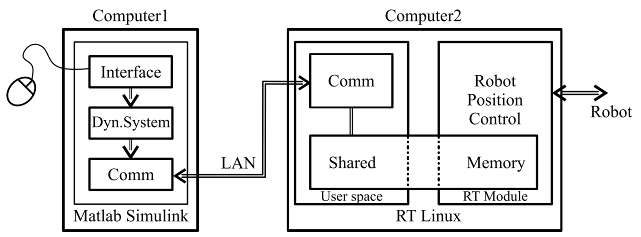

Fig. 18 Schematic of the robot control. We input the signal with a computer mouse in a Matlab/Simulink environment, run it through the Dynamical System and send it over the LAN using the UDP protocol to the computer controlling the robot in real time

robot, with a frequency of 100 Hz. The robot is position-controlled using Fujitsu's PID controller with a frequency of 1000 Hz. Since the robot control is 10 times faster than the loop on the side of the dynamical systems, the referential values between two received values are generated by interpolation.

5.2 Joint/task space

For a serial mechanism such as a robotic arm to repeat a trajectory, it has to learn the motion of all the joints. Our system allows us to either learn all of the joint trajectories directly or to learn the trajectories of the end-effector in task space and apply inverse kinematics algorithms to map the motion to the joints of the robot. By applying inverse kinematics algorithms we (usually) have to learn fewer trajectories, but we give up some of the control to the inverse kinematics algorithm.

Using such control structure we implemented a drumming task with a slightly simplified inverse kinematics algorithm that pre-determines one of the orientations in space, by-passing the problem of the arm redundancy.

5.3 Different inputs (mouse, Wiimote, robot)

The system is designed to work with any input device we can interface with the Matlab/Simulink environment. In this section, we present three different input options.

Computer mouse. With a standard computer mouse we can input a 2 DOF signal, which is enough for a simple one-handed drumming task or for repeating different arm trajectories on a plane. This is an easy way to record human motion, but requires some scaling of the robot's motion. Performing a periodic trajectory with a mouse usually also requires some practice.

Nintendo Wii remote. The Nintendo Wii remote or a Wiimote is a joystick of the Nintendo Wii game console. It has an infrared camera to track position of IR LEDs and gyroscopes for the orientation (see www.nintendo.com). It can be interfaced with a personal computer over the Bluetooth.

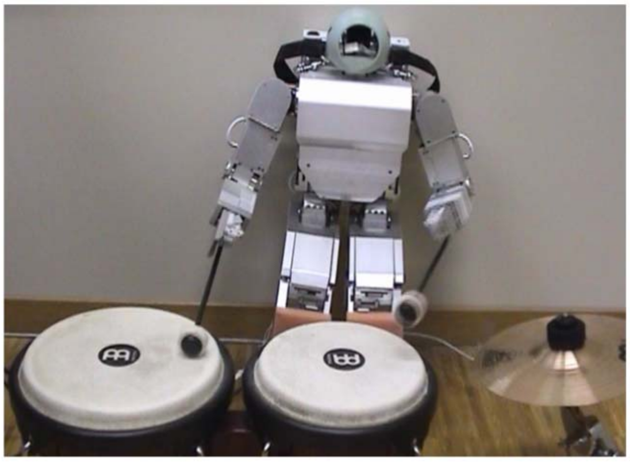

Fig. 19 Two handed drumming with the HOAP-2 robot

We used 2 Wiimotes to record the position of the demonstrator's hands and repeated two-handed drumming with the robot, as shown in Fig. 19. Using a Wiimote has the same scaling draw-back as using a computer mouse. Furthermore, the Wiimotes require quite precise movements of the arms, maintaining the "aim" at the IR LEDs (the sensor bar). Nevertheless, we can record 6 DOF per Wiimote. The draw-back of using the Wiimotes is the "coupling" of the position and orientation, i.e., when changing the yaw of the device, we also change the position. This can be solved by fixing the Wiimote and moving the IR LEDs, but that deprives us of the buttons. As our robot only has 4 DOF per arm, we were only interested in the position, using 2 Wiimotes for 6 DOF.

Kinesthetic demonstration. We can also record the robot motion by moving the robot by hand and adapting to the recorded joint positions (see also Hersch et al. 2008). This way we record all the joints we want to use, by-passing the inverse kinematics algorithm and also the calibration/scaling problem. The draw-back is that complex simultaneous motion of several DOF is difficult to achieve.

5.4 Different tasks

In this section we present some of the trajectories we learned with our system. Figure 20 shows some of the learned trajectories demonstrated on the HOAP-2 robot.

We tested the system for more complex trajectories, which are quite difficult to repeat periodically with the hand with a computer mouse/Wiimote or by kinesthetic demonstration, so we generated the signals. Figure 21 shows the results for two different 2D patterns. For both patterns we used $N = 100$ kernel functions. As we can see, the system successfully learned both patterns. The frequency adaptation took longer due to the increased complexity of the signal.

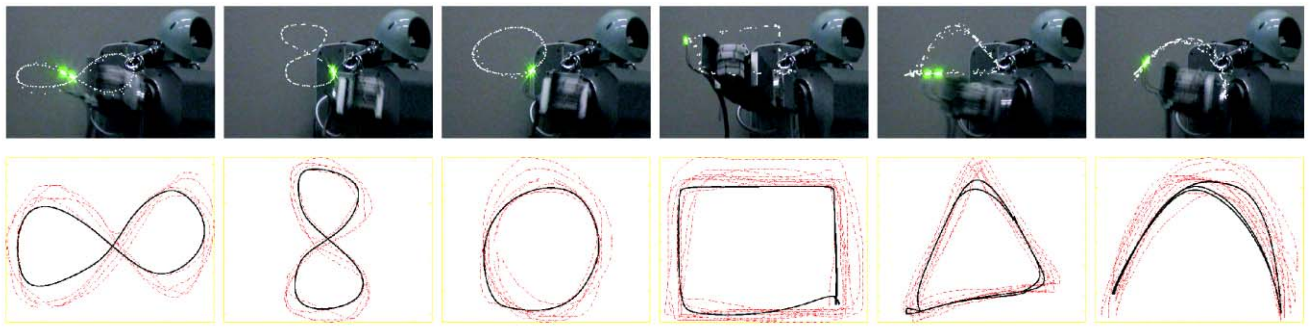

Fig. 20 Some of the learned 2D trajectories using a computer mouse as the input. The figure includes tracked trajectories of the tip of the robot's arm, and the plots of the input (*dotted*) and the learned trajectory (*solid line*)

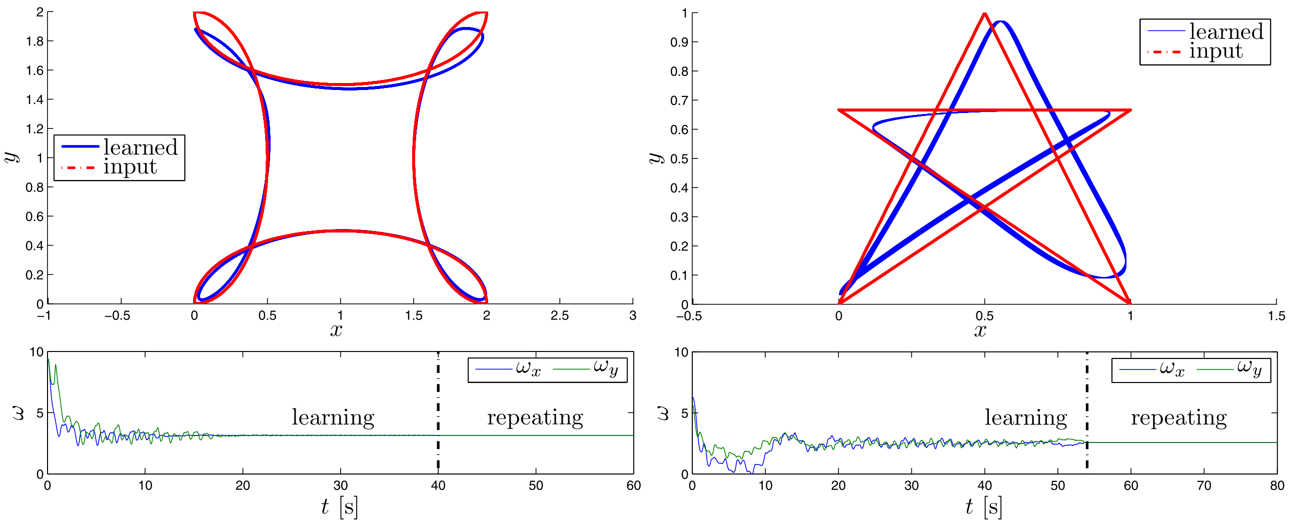

Fig. 21 Our system can cope with signals with multiple frequency components. *Bottom plots* show only the chosen fundamental frequency. The plotted signals were created during the repeating phase. To demonstrate the abilities of the system, the plots were done using the middle scheme from Fig. 17

5.5 Movement recognition

An interesting aspect of the system is that, given the temporal and spatial invariance of the representation, trajectories that are topologically similar are fit by similar parameters w_i (Ijspeert et al. 2002a). To illustrate the possibility of movement recognition, we compared some of the trajectories presented in Fig. 20. Each of the trajectories was manually recorded 5 times. Similarities between two trajectories can be measured by computing the correlation between their parameter vectors. The correlation gives the cosine of the angle between these two vectors. It can be calculated by $\frac{w_1^T w_2}{|w_1||w_2|}$, where w_1 and w_2 are the parameter vectors of two trajectories. These vectors are the union $w_1 = [w_1^x, w_1^y]$ of the parameter vectors for the $x(t)$ and $y(t)$ trajectories. From Fig. 22 we can see that the system clearly identifies similar trajectories (i.e. finds a high correlation between the 5 repeats of the same movement) despite the variations that naturally occurred when manually producing them. This can also be verified with a statistical t-test, which rejects the hypothesis that weights w_i at position i are distributed around the mean value of weights at position i for a different trajectory.

6 Discussion

In this section we first discuss the possibilities our systems presents us from the point of view of systems for imitating trajectories, followed by a discussion on expanding the system to combine different movements.

6.1 Possibilities of the system

This article presents a system for learning the frequency and the waveform of a periodic signal. The system was designed

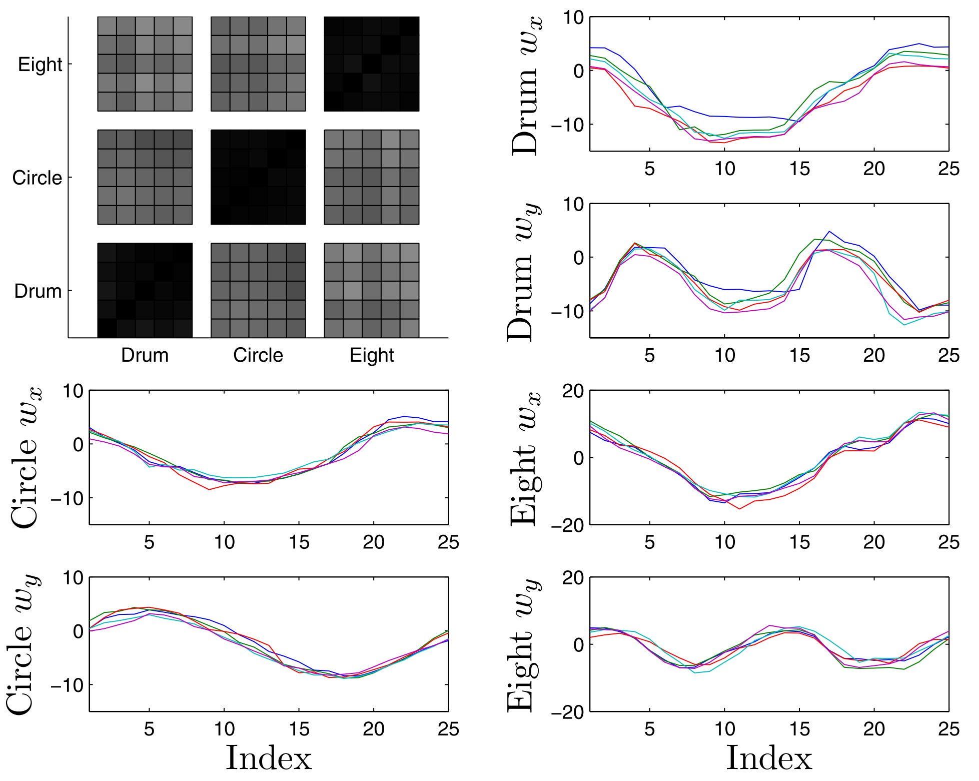

Fig. 22 In the *top left* plot we can see the calculated correlation, which is marked with *darker squares* for higher values. We can tell that some signals are more alike than the others, by having the diagonal squares darker than the other. In the other plots we can see the weight vectors for three different trajectories. Each trajectory was manually repeated 5 times. *Top-right* the hitting of two drums (as in last plot of Fig. 20), *bottom left* a circle and *bottom right* a horizontal figure-8

to match the desirable characteristics of an imitation system that we listed in Sect. 2.

Ease of learning and representing a trajectory. In the supervised learning framework presented in this paper, learning is based on locally weighted regression, which has several interesting properties. The computation is very fast compared, for instance, to gradient descent methods such as back-propagation in neural networks (Simard and LeCun 1991).

The parameters w_i are learned completely independently for each kernel function Ψ_i. This is not the case for radial basis functions, whose parameters are learned cooperatively. Here, the parameters w_i are independent of the number of basis functions and only depend on the location of the center of each kernel. They can be used robustly for categorization of different learned trajectories.

The parameters w_i are learned incrementally and adjusted each time a new data point is provided. With the forgetting factor λ set under 1.0, the newer values are given more weight, allowing changes to the trajectory on the fly.

For the learning of the weight parameters w_i to be effective, the frequency Ω of the input signal has to be known. The used adaptive frequency oscillators in a feedback loop allows quick and accurate frequency adaptation of both hand- and numerically-generated signals. All of the signal processing is embedded in the dynamics of the Canonical Dynamical System.

The number of adaptive frequency oscillators needed for the frequency adaptation was intentionally kept low at $M = 5$. Accurate values for complex input signals with multiple frequency components ensure better results, therefore for very complex signals increasing this number might be reasonable. Good results can also be achieved by a low number of oscillators, but this requires averaging of the chosen fundamental frequency over a certain period of time. All the results in this paper were achieved using $M = 5$ and averaging was not used. The frequency can be arbitrarily modulated during replay of the signal, as it becomes one of the control parameters.

Compactness of the presentation. Even though we input complex trajectories, only a limited number of parameters w_i are needed for encoding them. Similar to via-points, our presentation is a more compact presentation than storing all data points. This is further emphasized by the learning of only one period of the motion. The number and width of the Gaussian-like kernel Ψ_i functions can be adjusted depending on the desired accuracy of the reproduction, with a few and wide functions for a rough approximation, and many thin functions for a detailed approximation.

Ability to take perturbations into account. The dynamical systems do not only encode trajectories, but a whole attractor landscape. In other words, they also encode how the system returns to the learned trajectories after a perturbation of the state variables. This means that by design, the state variables can be perturbed and transient perturbations will rapidly be forgotten.

Since we generate the trajectories on-line by integrating the dynamical systems, it is also possible to modify the dynamical systems to include feedback terms. This is useful when the dynamical systems are used to control a robot. In the case of dealing with obstacles, feedback terms can slow down the dynamics and/or avoid some areas of the work space. Similarly repulsive forces can be included to avoid hitting joint angles limits.

The possibility to do on-line trajectory modulation is a key property of our system, and we see it as essential for trajectory planning applications in problems of human-robot interaction, dealing with contacts, avoiding obstacles and adjusting to a dynamic environment.

Ease of re-use for related tasks and of modification for new tasks. Besides the modulations related to perturbations, possible modulations include changes of frequency, amplitude and baseline. This means that from a single demonstration the system is capable of producing a range of trajectories with similar features. This is an important aspect since it gives the opportunity to reuse learned trajectories in new conditions. This makes our system similar to pattern generators observed in animals, for instance central pattern generators for the control of locomotion (Ijspeert 2008).

Ease of categorization for trajectory recognition. The system can to some extent be used for classification of spatiotemporal trajectories. The weight vectors can be used as a signature of the trajectory, and trajectories with similar velocity profiles can be classified accordingly, for instance, to their correlation. This makes the system useful for observing whether the learned movement is new or close to a previously learned one. This is particularly important to avoid learning too many movements, when the system is used in a learning by imitation framework, for instance.

The system, however, cannot differentiate between different trajectories and same trajectories with different control parameters, such as g. To account for parameter g one would have to exclude the 0 Hz frequency component from the demonstration signal, which can be easily done by averaging the learned periodic trajectory. This allows comparison against translated trajectories. The parameter r has no effect on the correlation as it does not affect the angle between two vectors.

6.2 Combining movements

This section gives a brief explanation of one of the possibilities offered by the system.

The system is designed to be linear in the weight space, which means that adding weight vectors of two different trajectories will result in a trajectory which is the sum of the two trajectories, allowing us to create very complex trajectories, by learning several simple ones. For this to work we would however need to know the desired phase relation between the learned signals. As it is, we cannot know whether the system has learned, for example, $\sin(t)$ or $\sin(t+\phi)$, and adding the two signals might not result in $2\sin(t)$. Combining the moves is therefore feasible but requires a higher level of task-oriented control system, which, as such, is not in the scope of this paper.

7 Conclusion

We presented a dynamical system that allows us to learn both the frequency and the waveform of the demonstration signal. To the best of our knowledge, this is the first realization of a generic learning system that can learn both quantities at the same time, being based on nonlinear dynamical systems that can guarantee basic stability and convergence properties of the learned system. We demonstrated applicability of the suggested techniques by learning various periodic movements for a humanoid robot, and illustrated the usefulness of the learned parametrization.

Acknowledgements The work described in this paper was supported by funding from the Swiss National Science Foundation, ARRS (Slovenia), and partially conducted within the European Commission's Cognition Unit, project no. IST-2004-004370: RobotCub, and the EU Cognitive Systems project PACO-PLUS (FP6-2004-IST-4-027657) funded by the European Commission.

References

Buchli, J., & Ijspeert, A. (2004). A simple, adaptive locomotion toy-system. In S. Schaal, A. Ijspeert, A. Billard, S. Vijayakumar, J. Hallam, & J. Meyer (Eds.), *From animals to animats 8. Proceedings of the eighth international conference on the simulation of adaptive behavior (SAB'04)* (pp. 153–162). Cambridge: MIT.

Buchli, J., Righetti, L., & Ijspeert, A. (2006). Engineering entrainment and adaptation in limit cycle systems—from biological inspiration to applications in robotics. *Biological Cybernetics*, *95*(6), 645–664.

Bullock, D., & Grossberg, S. (1989). VITE and FLETE: neural modules for trajectory formation and postural control. In W. Hersberger (Ed.), *Volitional control* (pp. 253–297). Amsterdam: Elsevier.

Calinon, S., Guenter, F., & Billard, A. (2007). On learning, representing and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, Special issue on robot learning by observation, demonstration and imitation, *37*(2), 286–298.

Crespi, A., Badertscher, A., & Guignard, A. (2005). AmphiBot I: An amphibious snake-like robot. *Robotics and Autonomous Systems*, *50*(4), 163–175.

Drumwright, E., Jenkins, O. C., & Mataric, M. (2004). Exemplar-based primitives for humanoid movement classification and control. In *Proceedings of the 2004 IEEE international conference on robotics and automation (ICRA 2004)*, pp. 140–145.

Fod, A., Mataric, M., & Jenkins, O. C. (2002). Automated derivation of primitives for movement classification. *Autonomous Robots*, *12*(1), 39–54.

Gams, A., Zlajpah, L., & Lenarcic, J. (2007). Imitating human acceleration of a gyroscopic device. *Robotica*, *25*(4), 501–509.

Grimes, D., Rashid, D., & Rao, R. (2006). Learning nonparametric models for probabilistic imitation. In *NIPS*, pp. 521–528.

Guenter, F., Hersch, M., Calinon, S., & Billard, A. (2007). Reinforcement learning for imitating constrained reaching movements. *RSJ Advanced Robotics*, Special Issue on Imitative Robots, *21*(13), 1521–1544.

Harris, C. M., & Wolpert, D. M. (1998). Signal-dependent noise determines motor planning. *Nature*, *394*(6695), 780–784. doi:10.1038/29528.

Hersch, M., Guenter, F., Calinon, S., & Billard, A. (2008). Dynamical system modulation for robot learning via kinesthetic demonstrations. *IEEE Transactions on Robotics*.

Ijspeert, A., Hallam, J., & Willshaw, D. (1999). Evolving swimming controllers for a simulated lamprey with inspiration from neurobiology. *Adaptive Behavior*, *7*(2), 151–172.

Ijspeert, A., Nakanishi, J. & Schaal, S. (2002a). Learning attractor landscapes for learning motor primitives. In *Advances in neural information processing systems 15 (NIPS2002)*.

Ijspeert, A., Nakanishi, J., & Schaal, S. (2002b). Learning rhythmic movements by demonstration using nonlinear oscillators. In *Proceedings of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS2002)* (pp. 958–963).

Ijspeert, A., Nakanishi, J., & Schaal, S. (2002c). Movement imitation with nonlinear dynamical systems in humanoid robots. In *IEEE International Conference on Robotics and Automation (ICRA2002)* (pp. 1398–1403).

Ijspeert, A. J. (2008). Central pattern generators for locomotion control in animals and robots: a review. *Neural Networks*, *21*(4), 642–653.

Inamura, T., Toshima, I., Tanie, H., & Nakamura, Y. (2004). Embodied symbol emergence based on mimesis theory. *International Journal of Robotic Research*, *23*(4–5), 363–377.

Kawamura, S., & Fukao, N. (1994). Interpolation for input torque patterns obtained through learning control. In *Proceedings of the third international conference on automation, robotics and computer vision (ICARCV'94)*.

Kawato, M. (1996). Trajectory formation in arm movements: minimization principles and procedures. In H. Zelaznik (Ed.), *Advance in motor learning and control* (pp. 225–259). Champaign: Human Kinetics Publisher.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotic Research*, *5*(1), 90–98.

Li, P., & Horowitz, R. (1999). Passive velocity field control of mechanical manipulators. *IEEE Transactions on Robotics and Automation*, *15*(4), 751–763.

Ljung, L., & Söderström, T. (1986). *Theory and Practice of Recursive Identification*. Cambridge: MIT.

Maass, W., Natschläger, T., & Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, *14*(11), 2531–2560.

Mataric, M. (1998). Behavior-based robotics as a tool for synthesis of artificial behavior and analysis of natural behavior. *Trends in Cognitive Science*, *2*(3), 82–87.

Mezger, J. W. I., & Giese, M. (2005). Trajectory synthesis by hierarchical spatio-temporal correspondence: comparison of different methods. In *APGV '05: Proceedings of the 2nd symposium on Applied perception in graphics and visualization* (pp. 25–32). New York: ACM.

Miyamoto, H., Schaal, S., Gandolfo, F., Koike, Y., Osu, R., Nakano, E., Wada, Y., & Kawato, M. (1996). A Kendama learning robot based on bi-directional theory. *Neural Networks*, *9*, 1281–1302.

Mussa-Ivaldi, F. (1997). Nonlinear force fields: a distributed system of control primitives for representing and learning movements. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation* (pp. 84–90). Los Alamitos: IEEE, Computer Society.

Nishii, J. (1998). A learning model for oscillatory networks. *Neural Networks*, *11*(2), 249–257.

Okada, M., Tatani, K., & Nakamura, Y. (2002). Polynomial design of the nonlinear dynamics for the brain-like information processing of whole body motion. In *Proceedings of ICRA 2002* (Vol. 2, pp. 1410–1415). IEEE.

Paine, R. W., & Tani, J. (2004). Motor primitive and sequence self-organization in a hierarchical recurrent neural network. *Neural Networks*, *17*, 1291–1309.

Righetti, L., & Ijspeert, A. (2006). Programmable central pattern generators: an application to biped locomotion control. In *Proceedings of the 2006 IEEE international conference on robotics and automation* (pp. 1585–1590).

Righetti, L., Buchli, J., & Ijspeert, A. (2006). Dynamic Hebbian learning in adaptive frequency oscillators. *Physica D*, *216*(2), 269–281.

Rohrer, B., & Hogan, N. (2003). Avoiding spurious submovement decompositions: a globally optimal algorithm. *Biological Cybernetics*, *89*(3), 190–199.

Schaal, S. (1999). Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, *3*, 233–242.

Schaal, S., & Atkeson, C. (1998). Constructive incremental learning from only local information. *Neural Computation*, *10*, 2047–2084.

Schaal, S., Mohajerian, P., & Ijspeert, A. (2007). Dynamics systems vs. optimal control a unifying view. *Progress in Brain Research*, *165*, 425–445.

Simard, P., & LeCun, Y. (1991). Reverse TDNN: an architecture for trajectory generation. In *NIPS* (pp. 579–588).

Tani, J., & Ito, M. (2003). Self-organization of behavioral primitives as multiple attractor dynamics: A robot experiment. *IEEE Transactions on Systems, Man, and Cybernetics*, *33*(4), 481–488.

Tsuji, T., Tanaka, Y., Morasso, P., Sanguineti, V., & Kaneko, M. (2002). Bio-mimetic trajectory generation of robots via artificial potential field with time base generator. *Systems, Man and Cybernetics, Part C, IEEE Transactions on*, *32*(4), 426–439.

Ude, A., Riley, M., Nemec, B., Kos, A., Asfour, T., & Cheng, G. (2007). Synthesizing goal-directed actions from a library of example movements. In *IEEE-RAS/RSJ international conference on humanoid robots (Humanoids 2007)*.

Zegers, P., & Sundareshan, M. (2003). Trajectory generation and modulation using dynamic neural networks. *IEEE Transactions on Neural Networks*, *14*(3), 520–533.

Andrej Gams received his Ph.D. in robotics from the University of Ljubljana, SLO, in 2009. He is employed as an assistant at the Department of Automation, Biocybernetics and Robotics at "Jozef Stefan" Institute in Ljubljana, since 2004. His fields of interest include human arm motion and imitation of rhythmic tasks. He did part of his doctoral research at the Biologically Inspired Robotics Group (BIRG) at EPFL.

Auke J. Ijspeert is a SNF (Swiss National Science Foundation) assistant professor at the EPFL (the Swiss Federal Institute of Technology at Lausanne), and head of the Biologically Inspired Robotics Group (BIRG). He has a BSc/MSc in Physics from the EPFL, and a Ph.D. in artificial intelligence from the University of Edinburgh. He carried out postdocs at IDSIA and EPFL (LAMI) with Jean-Daniel Nicoud and Luca Gambardella, and at the University of Southern California (USC). Before returning to the EPFL, he was a research assistant professor at USC, and an external collaborator at ATR (Advanced Telecommunications Research institute) in Japan. He is still affiliated as adjunct faculty to both institutes. His research interests are at the intersection between robotics, computational neuroscience, nonlinear dynamical systems, and adaptive algorithms (optimization and learning algorithms). He is interested in using numerical simulations and robots to get a better understanding of the functioning of animals (in particular their fascinating sensorimotor coordination abilities), and in using inspiration from biology to design novel types of robots and adaptive controllers. He is regularly invited to give talks on these topics. With his colleagues, he has received the Best Paper Award at ICRA2002, the Industrial Robot Highly Commended Award at CLAWAR2005, and the Best Paper Award at the IEEE-RAS Humanoids 2007 conference. He is/was the Technical Program Chair of 5 international conferences (BioADIT2004, SAB2004, AMAM2005, BioADIT2006, LATSIS2006), and has been a program committee member of over 35 conferences.

Stefan Schaal is an Associate Professor at the Department of Computer Science and the Neuroscience Program at the University of Southern California, and an Invited Researcher at the ATR Human Information Sciences Laboratory in Japan, where he held an appointment as Head of the Computational Learning Group during an international ERATO project, the Kawato Dynamic Brain Project (ERATO/JST). Before joining USC, Dr. Schaal was a postdoctoral fellow at the Department of Brain and Cognitive Sciences and the Artificial Intelligence Laboratory at MIT, an Invited Researcher at the ATR Human Information Processing Research Laboratories in Japan, and an Adjunct Assistant Professor at the Georgia Institute of Technology and at the Department of Kinesiology of the Pennsylvania State University. Dr. Schaal's research interests include topics of statistical and machine learning, neural networks, computational neuroscience, functional brain imaging, nonlinear dynamics, nonlinear control theory, and biomimetic robotics. He applies his research to problems of artificial and biological motor control and motor learning, focusing on both theoretical investigations and experiments with human subjects and anthropomorphic robot equipment.

Jadran Lenarčič is currently Director of the "Jozef Stefan" Institute in Ljubljana, Slovenia. His research interests are in robotics, in particular in robot kinematics, humanoid robotics and biomechanics. He conducted a number of research projects in these areas and published several scientific journal and conference papers. He was visiting professor at the University of Hull (England), University of Bologna (Italy), and visiting scholar at the Notre Dame University (USA). He is co-chairing a series of international symposia Advances in Robot Kinematics.