Biological Cybernetics manuscript No.
(will be inserted by the editor)

Frequency Modulation of Large Oscillatory Neural Networks

Francis wyffels · Jiwen Li · Tim Waegeman · Benjamin Schrauwen · Herbert Jaeger

Received: date / Accepted: date

Abstract Dynamical systems which generate periodic signals are of interest as models of biological central pattern generators (CPGs) and in a number of robotic applications. A basic functionality that is required in both biological modelling and robotics is frequency modulation. This leads to the question of whether there are generic mechanisms to control the frequency of neural oscillators. Here we describe why this objective is of a different nature, and more difficult to achieve, than modulating other oscillation characteristics (like amplitude, offset, signal shape). We propose a generic way to solve this task which makes use of a simple linear controller. It rests on the insight that there is a bidirectional dependency between the frequency of an oscillation, and geometric properties of the neural oscillator's phase portrait. By controlling the geometry of the neural state orbits, it is possible to control the frequency on the condition that the state-space can be shaped such that it can be pushed easily to any frequency.

Keywords Reservoir Computing · Pattern Generators · Frequency Modulation

1 Introduction

Across the animal kingdom, many biological functions involve the neural generation of periodic motor patterns. Classical examples include processes of ingestion, digestion, breathing, heartbeat, eye motion, reproduction, grooming and locomotion. The generation of the requisite periodic motion signals is commonly attributed to *central pattern generators* (CPGs, Grillner (1985), reviews: Ijspeert (2008); Büschges et al (2011)). In the classical understanding of this concept, a CPG is a small, genetically archaic, neural circuit, typically located in the brainstem or spinal cord, which is often capable of autonomous oscillations even in the absence of neural input. Models of CPGs range in abstraction from detailed reconstructions of neural circuits to simplified ordinary differential equations (ODEs) which capture essentials of the observable dynamics. All of these models can be considered small in the sense that they employ low-dimensional state spaces and/or a small number of neurons (say, order of 10 or less).

However, there are indications that biological CPGs are embedded in, or closely interact with, larger circuits than what has standardly been realised in models. For instance, recent studies in a leech model (Briggman and Kristan Jr. (2006), see Briggman and Kristan Jr. (2008) for a review), show that crawling and swimming are effected by two CPGs of which most neurons overlap, forming a comprehensive, multi-functional circuit, which can operate in at least two different regimes at very different timescales. Büschges et al (2011) supply further evidence for a more complex and larger-scale picture, afforded by the advent of genetic manipulation techniques. Furthermore, humans (and possibly other animals) are capable of voluntary action with periodic components which obviously involve cortical contributions, e.g., in sports, music or dance. Regardless of whether cortical signals are themselves periodic, or whether they interact with "lower" CPGs, the

F. wyffels · T. Waegeman · B. Schrauwen
Ghent University, Electronics and Information Systems Department, Sint-Pietersnieuwstraat 41, 9000 Ghent, Belgium
Tel.: +32 9 264 95 26
Fax: +32 9 264 35 94
E-mail: Francis.wyffels@UGent.be

J. Li · H. Jaeger
Jacobs University Bremen gGmbH, Campus Ring, 28759 Bremen, Germany

complete dynamics involves neural populations of sizes much larger than in "classical" CPGs.

In robot engineering, tuneable periodic patterns have to be generated for a variety of motor behaviours. The field has been inspired by biological CPG research and has adopted concepts and terminology. Collaborations between roboticists and biologists aim at testing biological theory in robot models (e.g., stick insect walking (Cruse et al, 1995; Dean et al, 1999) or salamander locomotion (Ijspeert et al, 2007)) or, vice versa, at making biological solutions fertile for engineering (e.g., for humanoid (Nakanishi et al, 2004) or quadruped (Fukuoka et al, 2003) locomotion).

Like their counterparts in biological modelling, CPG models employed in robots have almost always been realised as ODEs or as small-sized neural oscillators. Typically, these models are autonomous dynamical systems and embed at least one limit cycle attractor. In order to add modulation capabilities, such systems have a small number of tuneable parameters. This enables the transparent modulation of dynamical characteristics such as amplitude, offset, phase lags and frequency, but also less trivial modulations such as independently adjusting the swing and stance phase (survey of design strategies in Buchli et al (2006)). Alternatively, by driving such a small dynamical system with a forcing term, the output can be shaped such that it follows a desired trajectory (see Ijspeert et al (2013) for a review).

Like in biological research, also in robotics there is a good reason to consider larger-scale dynamical systems for pattern generation – say, in the order of hundreds of dimensions or neurons. The reason is that one wishes to endow the pattern generators with a rich and learnable repertoire of a variability that extends far beyond the customary modulation of amplitude, offset and frequency. Such additional degrees of flexibility include waveform, relative phase angles (in multidimensional output systems), input and control gains, obstacle avoidance, phasing-in and phasing-out, starting and stopping, coordinated interaction with other behaviours, adaptation to different environments and target objects, high-dimensional sensor input, user command interfacing, and more. While for each of these qualities specific solutions have been proposed for small-sized CPGs, these have not been combined into integrated systems. It seems likely that pattern generation modules which can offer such flexibility would need to be larger than the customary CPGs. Furthermore, using neural networks seems to be a plausible route toward realising *learnability* of such functionalities.

In other work we and partners in the European AMARSi project (see acknowledgments) have taken first steps toward training large neural networks for robotic CPGs (Reinhart and Steil, 2008; wyffels and Schrauwen, 2009; Wrede et al, 2010; Rolf et al, 2010a,b; Reinhart and Steil, 2011; Waegeman and Schrauwen, 2011; Waegeman et al, 2012b). Specifically, we are using recurrent neural networks (RNNs) of the *echo state networks* (ESN) type, a particular flavour of what has become known as the *reservoir computing* (RC) paradigm. This approach led to encouraging progress in robust training and modulation of waveforms (wyffels and Schrauwen, 2009; Waegeman et al, 2012b), in merging the pattern generation with the end-effector control (Waegeman et al, 2012c), in bidirectional forward-inverse kinematic transformations (Reinhart and Steil, 2008), in fast learning of human-demonstrated motions (Wrede et al, 2010), in endowing a single RNN with the capacity to handle different tool objects (Rolf et al, 2010b), using a RNN for feedback control by online learning an inverse model (Waegeman et al, 2012a), or learning rhythmical patterns with tensegrity structures (Caluwaerts et al, 2013a). However, attempts have essentially failed so far to extend generic learning and control mechanisms, which work well for amplitude and offset modulation, to frequency modulation (Li and Jaeger, 2011). We will argue below in Section 4 that frequency modulation is a task which is intrinsically different from modulating other characteristics of a neuro-dynamical system.

The biological perspective adds another angle to this riddle. Humans can generate voluntary action at varying speeds, both periodic/rhythmic (e.g., walking, singing) and non-periodic (e.g., reaching). Some of these can possibly be explained by specific speed modulation mechanisms of basal CPGs, but the human ability to reproduce arbitrary teacher motions ad hoc at different speeds suggests the existence of generic speed regulation mechanisms for some cortical processes which comprises large neural populations. But, individual neurons cannot be simply sped up or slowed down by changing a time constant like it is possible with ODEs. Alternatively, the oscillation period of small-sized ODEs can be modulated by an external input (Curtu et al, 2008; Daun et al, 2009; Zhang and Lewis, 2013). However, it is not clear how this scales up to large populations of interacting neurons.

In this article we propose a generic basis for neural processing speed adjustments of *large* oscillatory RNNs which does not hinge on time-constant changing mechanisms. The key observation is that when a (large or small) RNN is driven by a periodic signal which passes through a frequency sweep, the geometry of the phase portrait co-varies with the driving frequency (Section 4). This connection can be exploited in reverse direction: if, in a suitable training setup, the

RNN has been trained as an oscillator, its frequency in signal generation mode can be modulated by controlling geometric properties of the phase portrait (Section 3). Indeed, adjusting a scalable bias suffices. We unfold this scenario within the reservoir computing framework, whose basics are briefly outlined in Section 2. A potential obstacle to frequency control is that disruptive bifurcations might occur during attempts to regulate the frequency. This danger can be kept at bay by a special kind of network regularisation which we call *equilibration* and explain in Section 5. We demonstrate the robustness of the method by showing that it always worked across all instances of a large sample of randomly varied RNNs, provided that the equilibration was successful (Section 6). In the concluding Section 7 we discuss in more depth the main contributions and results of this work.

2 Designing an Echo State Network pattern generator

In setting up and training our systems, we follow the principles of reservoir computing (RC) (Verstraeten et al, 2007). More specifically, we use a flavour of RC known as *echo state networks* (ESNs) (Jaeger, 2001). We first recall the basic usage of this method for training neural pattern generators.

2.1 ESN pattern generator design

An ESN setup for generating a desired one-dimensional periodic sequence $\mathbf{y}_{\text{desired}}$ is governed by the discrete-time state update equations

$$\mathbf{x}[k+1] = (1-\lambda)\mathbf{x}[k] + \lambda \tanh\left(\mathbf{W}_{\text{res}}\mathbf{x}[k] + \mathbf{W}_{\text{fb}}\mathbf{y}[k] + \mathbf{W}_{\text{bias}}\right), \quad (1)$$

$$\mathbf{y}[k+1] = \mathbf{W}_{\text{out}}^{\intercal}\mathbf{x}[k+1], \quad (2)$$

where $\mathbf{x}$ is the N-dimensional internal network state, $\mathbf{y}$ the (here: one-dimensional) output signal, $\mathbf{W}_{\text{res}}$ is the $N \times N$ internal weight matrix, $\mathbf{W}_{\text{fb}}$ is the $N \times 1$ output feedback weight vector, $\mathbf{W}_{\text{bias}}$ is a bias vector, $\mathbf{W}_{\text{out}}^{\intercal}$ are the output weights, and λ functions as a leaking rate. We adhere to the terminology of the field and call the recurrent internal layer governed by $\mathbf{W}_{\text{res}}$ the *reservoir* and $\mathbf{x}$ the *reservoir state*. This setup is illustrated in Fig. 1.

When creating such an ESN, the reservoir weights $\mathbf{W}_{\text{res}}$ are usually sampled from a standard normal distribution and then scaled to tune the dynamics of the ESN. For this, the spectral radius ρ, which is defined as the largest absolute eigenvalue of $\mathbf{W}_{\text{res}}$, is used and often $\rho = 1$ is taken as a reference point (Lukoševičius, 2012). Similarly, $\mathbf{W}_{\text{fb}}$ and $\mathbf{W}_{\text{bias}}$ are typically sampled from a standard normal distribution with variances scaled to o and β respectively. It is known that with small reservoir weights (and thus a small spectral radius), the system, when run with zero input, will possess dynamics characterised by a single global stable fixed point (which is zero if the bias $\mathbf{W}_{\text{bias}}$ is zero). When the weights are scaled up, at some point this globally stable fixed point dynamics undergoes a bifurcation. Generally, the weights are scaled up to a point just before this bifurcation (see Verstraeten et al (2007); Lukoševičius and Jaeger (2009); Sussillo and Abbott (2009); Yildiz et al (2012); Caluwaerts et al (2013b) for analysis and discussion of the impact of ρ on the system dynamics). Table 1 gives an overview of the system's parameters and their typical range.

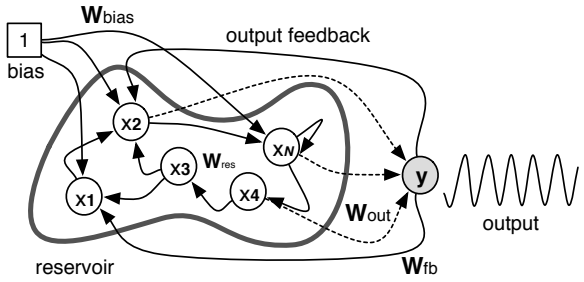

Fig. 1 Schematic overview of a reservoir system for robustly generating periodic patterns, i.e., an ESN pattern generator. Only the readout weights (dashed connections) are trained.

Apart from the weights, the timescale on which the system operates plays an important role. This can be effectively tuned by choosing the sample rate for the input and output signals of the system (Schrauwen et al, 2007). Alternatively, as applied in this work, the timescale can be set by tuning the leaking rate λ (Jaeger, 2001).

In reservoir computing, the only parameters that are traditionally modified/calculated in training are the output weights $\mathbf{W}_{\text{out}}^{\intercal}$. All other weights remain at their random, globally scaled, creation-time values.

Table 1 Typical parameters for ESN pattern generators.

Parameter	Description	Value
N	number of neurons	50 to 2000
ρ	spectral radius	0.5 to 2.0
β	bias weight variance	0 to 1
o	output feedback scale	0 to 10
λ	leak-rate	0 to 1

2.2 Training the readout weights using FORCE learning

The optimisation criterion for training $\mathbf{W}_{\text{out}}$ is the squared error between the actual output $\mathbf{y}[k]$ and a desired output $\mathbf{y}_{\text{desired}}[k]$, averaged over time. Computing the weights $\mathbf{W}_{\text{out}}$ can be done with a variety of computational schemes, online or offline, each of which implements a linear regression of the reservoir states $\mathbf{x}[k]$ on the targets $\mathbf{y}_{\text{desired}}[k]$. In order to ensure stability of the reservoir-output feedback loop dynamics, the standard approach is to regularise the output weights. For offline training, state noise injection (Jaeger, 2002) and ridge regression (wyffels et al, 2008) are commonly used. When training the readout weights $\mathbf{W}_{\text{out}}$ online, Sussillo and Abbott (2009) introduced a weight adaptation algorithm called FORCE learning, which we find works well for training ESNs with output feedback, which is why we use it here.

FORCE learning differs from standard (offline) approaches to reservoir training in three ways. First, it is an online learning method, where the output weights are adapted at each training time step. Second, the network weights are initialised before training such that the spectral radius of the overall weight matrix is significantly larger than 1. As a result, the reservoir exhibits spontaneous activity. Third, the actual self-generated output – and not the correct teacher signal – is fed back into the ESN pattern generator during training.

For training the output weights $\mathbf{W}_{\text{out}}$, FORCE learning prescribes to use learning algorithms that rapidly reduce (and keep small) the magnitude of the difference between the actual and desired output (Sussillo and Abbott, 2009). For this, the well-known recursive least squares (RLS) online learning algorithm is adopted. With RLS, the reservoir states $\mathbf{x}[k+1]$ are updated using Equation 1, while at every time step the readout weights $\mathbf{W}_{\text{out}}[k+1]$ and the output $\mathbf{y}[k+1]$ are adjusted according to the following equations:

$$\mathbf{e}[k+1] = \mathbf{W}_{\text{out}}^{\intercal}[k]\mathbf{x}[k+1] - \mathbf{y}_{\text{desired}}[k+1] \quad (3)$$

$$\mathbf{P}[k+1] = \mathbf{P}[k] - \frac{\mathbf{P}[k]\mathbf{x}[k+1]\mathbf{x}^{\intercal}[k+1]\mathbf{P}[k]}{1+\mathbf{x}^{\intercal}[k+1]\mathbf{P}[k]\mathbf{x}[k+1]} \quad (4)$$

$$\mathbf{W}_{\text{out}}[k+1] = \mathbf{W}_{\text{out}}[k] - \mathbf{e}[k+1]\mathbf{P}[k+1]\mathbf{x}[k+1] \quad (5)$$

$$\mathbf{y}[k+1] = \mathbf{W}_{\text{out}}^{\intercal}[k+1]\mathbf{x}[k+1]. \quad (6)$$

Here $\mathbf{e}[k+1]$ is the difference between the actual output $\mathbf{y}[k+1]$ and the desired output $\mathbf{y}_{\text{desired}}$ at time step $k+1$. $\mathbf{P}$ $(N \times N)$ is an estimation of the inverse of the correlation matrix of the network states $\mathbf{x}$ and is initialised at $\mathbf{P}[0] = \alpha^{-1}\mathbb{I}$, where $\mathbb{I}$ is the identity matrix, with α typically small (set at 0.1 in this work). The readout weights $\mathbf{W}_{\text{out}}[k]$ at time step k are initialised to $\mathbf{W}_{\text{out}}[0] = \mathbf{0}$. After K time steps, when training is finished, the readout weights are kept fixed ($\mathbf{W}_{\text{out}} = \mathbf{W}_{\text{out}}[K]$) and the reservoir system can be used for recursively generating patterns by using equations 1 and 2. Previous work on FORCE learning (Sussillo and Abbott, 2009) and its applications (Waegeman et al, 2012a) have shown that $\mathbf{e}[k+1]$ (see equation 3), and, consequently also the readout weights $\mathbf{W}_{\text{out}}[k+1]$, converges.

Using the above procedure, we can train a reservoir system with any rhythmic signal such that it generates this periodic pattern and thus becomes an ESN pattern generator.

3 Modulating an ESN pattern generator

The objective of this work is to realise frequency modulation of ESN pattern generators. One way to do this is by directly training the ESN pattern generator such that its output changes under the influence of an additional input signal (see for example Jaeger (2002); Sussillo and Abbott (2009)). However, the problem with these input driven systems is that their modulation range is defined by a well chosen training set of input-output combinations during the training phase. Any unseen input might lead to an undesired shape modulation during the modulation phase due to the open loop nature of controlling the output.

To overcome this problem, Li and Jaeger (2011) proposed a method to control a number of characteristics of an oscillating ESN pattern generator by means of an external, trainable control loop. While this worked well for modulating amplitude and shift, attempts to regulate frequency essentially failed. This indicated that frequency modulation of CPG output may have a different nature compared to those of other characteristics.

In this work, the objective of this controller (Fig. 2) is to make the network-generated oscillation track a desired frequency, of which the measured and desired period lengths at time step k are denoted by $T[k]$ and $\hat{T}[k]$, respectively. Therefore, the controller needs access to measurements of period length $T[k]$. For the purpose of our demonstrations a coarse, simple observer is sufficient. We counted the number of simulation time steps between two successive maxima of the the network output signal, where a maximum at time k was defined by the condition $(\mathbf{y}[k-1] < \mathbf{y}[k]) \wedge (\mathbf{y}[k] > \mathbf{y}[k+1])$. Notice that these measurements result in integer readings which are constant for at least the duration of a period.

The target value $\hat{T}[k]$ is likewise integer-valued. Its interpolation should be changing on a timescale that is

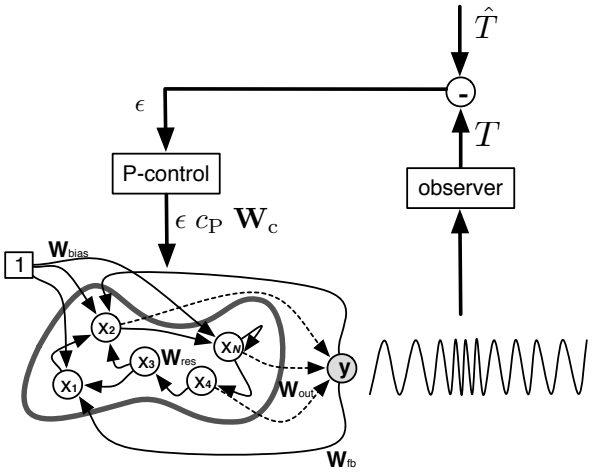

Fig. 2 Schematic overview of the control architecture. For explanation see text.

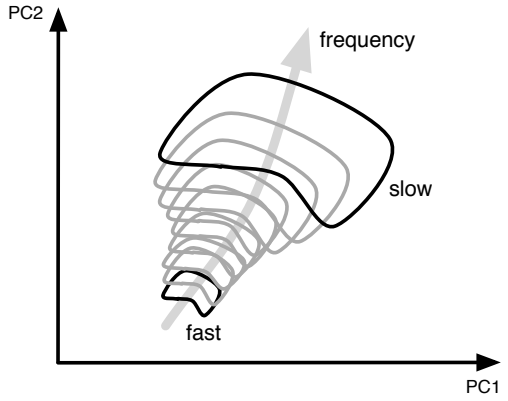

Fig. 3 Sketch of the 2-dimensional projection of the state-space of a linearly tuneable ESN pattern generator. The proposed control architecture hinges on a smooth monotonic variation of the oscillation signal in state space.

at least one order of magnitude slower than the timescale of the individual oscillations. Comparing the measured period signal with the target gives a (normalised) error

$$\epsilon = \frac{\hat{T}[k+1] - T[k+1]}{T[k+1]}. \tag{7}$$

The control input to the reservoir is simply a bias vector $\mathbf{W}_{\mathrm{c}}$ ($N \times 1$) which is scaled with a constant proportional gain c_{P} (a scalar that has to be tuned) and the error ϵ (scalar), leading to a controlled network update equation of the form

$$\begin{aligned}\mathbf{x}[k+1] = (1-\lambda)\ \mathbf{x}[k] \\ +\lambda \tanh\Big(\mathbf{W}_{\mathrm{res}}\ \mathbf{x}[k] + \mathbf{W}_{\mathrm{fb}}\ \mathbf{y}[k] \\ +\mathbf{W}_{\mathrm{bias}} + \epsilon\ c_{\mathrm{P}}\ \mathbf{W}_{\mathrm{c}}\Big).\end{aligned} \tag{8}$$

This method obviously hinges on finding a suitable control bias $\mathbf{W}_{\mathrm{c}}$. In (Li and Jaeger, 2011) we used a perturbation-based learning approach to train $\mathbf{W}_{\mathrm{c}}$, and in (Jaeger, 2010) a technique based on a correlational analysis. In this work, we employ a third, simple and intuitive method which assumes that, as sketched in Fig. 3, the location of the oscillations in state space vary monotonically with frequency. We constitute $\mathbf{W}_{\mathrm{c}}$, as follows:

- Drive the reservoir using equation 1 with an external oscillating signal of length K with decreasing frequency. The driving is effected by writing the target oscillation into the output neuron $\mathbf{y}$ ("teacher forcing"). Collect the driven neuron states $\mathbf{x}[k]$.
- Smoothen this raw network signal by taking a moving average, to obtain $\mathbf{x}_{\mathrm{avg}}[k]$. In this work we used a time window of $2T_0$ with T_0 the initial period length.
- Calculate the control weights: $\mathbf{W}_{\mathrm{c}} = \mathbf{x}_{\mathrm{avg}}[K] - \mathbf{x}_{\mathrm{avg}}[2T_0]$.

As mentioned before, achieving frequency modulation by using this simple control scheme can be very hard. The question that remains is: *How can we shape the state-space of the ESN pattern generator such that it looks like the one sketched in Fig. 3?*

4 Frequency expressed as geometrical characteristics of the reservoir

The difficulties of achieving frequency modulation with the described control framework indicate that frequency modulation of the CPG's output may be of different nature compared to those of other characteristics.

This difference can be illuminated in two ways. Most non-frequency related properties of an output signal generated from a CPG can be modulated by post-processing the output with suitable filters. All of these filters do not interfere with the core CPG dynamics. Such a decoupling of modulation from generation is not possible when frequency is at stake. Another view on the same conundrum is obtained when one considers phase portraits of ODE-based CPGs which are modulated by varying control parameters (Buchli et al, 2006). When the modulation target is not frequency, the phase portraits invariably alter their geometry; when conversely frequency is changed (by varying the ODE's time constant), the phase portrait remains the same.

Previous work on small-sized ODEs (Curtu et al, 2008; Daun et al, 2009; Zhang and Lewis, 2013) suggest

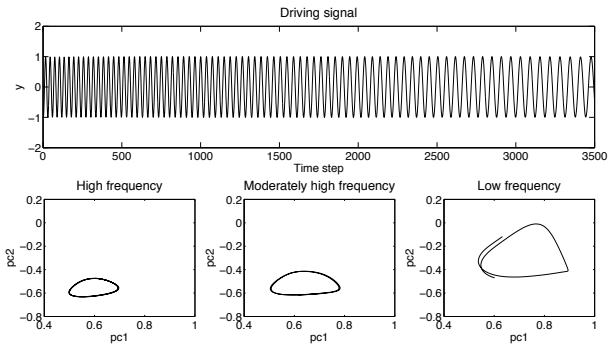

Fig. 4 Driving an ESN pattern generator by a oscillation with gradually decreasing frequency (top plot) does not cause the dynamics to simply slow down. Instead, as can be observed in the bottom plots, the geometrical/metric properties of the phase portraits change. From left to right one can observe that the geometry of the phase portrait is changing while the frequency of the teacher signal is decreased. For the three phase portraits we used Principal Component Analysis (PCA) (Jolliffe, 2005) to obtain a 2-dimensional projection of the reservoir states.

that the oscillation period of half-center oscillators can be controlled by external inputs, a mechanism which does not rely on changing the time constant. To understand the nature of frequency modulation of a *large* non-linear dynamical system such as the discussed ESN pattern generator, we first investigate the dynamics of ESN pattern generator under external driven input.

Specifically, we inject a gradually changing oscillation ($y[k] = \sin(0.075s[k]k)$, with $s[k]$ a time dependent scaling factor which linearly decreases from 2 to 1, see top panel in Fig. 4) through the feedback weights $\mathbf{W}_{\text{fb}}$ of an ESN pattern generator to drive the reservoir, and record the reservoir states. Then we visualise the phase portraits of this system by computing the 1st and 2nd largest principal component (PC) of the trajectory, and plotting the projections of reservoir states on the 1st PC versus that of the 2nd PC (bottom panels in Fig. 4). From the bottom panels of this figure one can see clearly that the phase portraits of the reservoir change in shape and offset across this driving input frequency-sweeping oscillation, and do so quite substantially. Details of the experimental setup are documented in Section 6.

So it appears that when a reservoir network *is passively driven* by an external signal with varying *frequency* characteristics, its excited dynamics responds with a variation not only of its speed but also of its *geometrical* characteristics. In the remainder of this article we investigate whether this causation can be reversed: is it possible to modulate (only) the geometrical characteristics of the internal dynamics of a reservoir, have this reservoir *actively generate* an output signal, and obtain a purely frequency variation in the latter? In other words: *Can we train an ESN pattern generator such that, in state space, the oscillating signal changes in a smooth monotonic way, i.e., its 2-dimensional projection looks similar to the sketch in Fig. 3.*

The answer, as we will see, is yes. Indeed, in the case study that we are going to present, it is enough to add a bias of varying scale to the network dynamics in order to obtain a geometry change that induces a frequency sweep in the output. However, implementing a robust geometry-to-frequency causation is not without difficulties. The main challenge that we encountered was to avoid bifurcations along the scaling route of the additional bias input. The key to success turned out to be a reservoir pre-training which we termed *equilibration* in earlier work (Jaeger, 2010; Li and Jaeger, 2011). In the next section we recapitulate the basic ideas of this technique.

5 Equilibration

The mechanism behind equilibration – namely, "internalizing" a driven dynamics into a reservoir – has been independently (re-)introduced under different names and for a variety of purposes (*self-prediction* Mayer and Browne (2004), *equilibration* Jaeger (2010), *reservoir regularisation* Reinhart and Steil (2012), *self-sensing networks* Sussillo and Abbott (2012), *innate training* Laje and Buonomano (2013)). It appears to be an RNN adaptation principle that is fundamental, versatile and simple. In the following subsections we explain the concept of equilibration by a simple example, after which we discuss the equilibration of ESN pattern generators.

5.1 Synthetic example of equilibration

To illustrate the concept of equilibration it is helpful to consider a simple synthetic example (repeated from Li and Jaeger (2011)). At the top of Fig. 5 the phase portrait of a stable circular oscillation defined in cylindrical coordinates by $\dot{r} = \tau(-r + \exp x)$, $\dot{\theta} = 1$, $\dot{x} = -cx$ is shown. Here r is a radius, θ an angle, and x a position. This system has a globally attracting limit cycle at $x = 0, r = 1$. If we would treat x as an input control parameter, we would be left with a 2-dimensional system in r and θ whose dynamics is controlled by x. Feeding x with different constant values would yield stable circular oscillations with radius equal to $\exp(x)$ (bottom panel). There is another way to obtain exactly the same bottom-panel phase portrait: use the 3-dimensional autonomous system equation

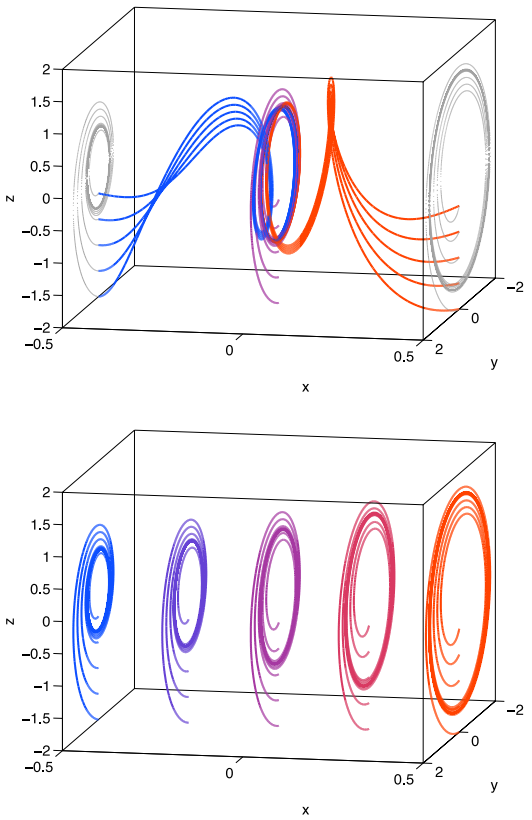

Fig. 5 Two phase portraits. Top panel: this system is governed by $\dot{r} = \tau(-r + \exp x)$, $\dot{\theta} = 1$, $\dot{x} = -cx$. The portrait at the bottom can be interpreted in two ways: (i) as a collection of phase portraits of a 2-dimensional system $\dot{r} = \tau(-r + \exp x)$, $\dot{\theta} = 1$ in variables r, θ, controlled by an external input x, or (ii) as a 3-dimensional system $\dot{r} = \tau(-r + \exp x)$, $\dot{\theta} = 1$, $\dot{x} = 0$. The polar coordinates θ, r are plotted to the y, z plane. For further comments see text.

$$\dot{r} = \tau(-r + \exp x), \tag{9}$$

$$\dot{\theta} = 1, \tag{10}$$

$$\dot{x} = 0. \tag{11}$$

Notice that the dynamics of this 3-dimensional autonomous system is neutrally stable in the x direction, while in the y, z directions it produces a stable oscillation whose amplitude depends on x.

The bottom-panel system exhibits what we call *equilibration*. Intuitively, it has *internalised* the x-input controlled dynamics in which x essentially stands still at different values and maintains a fixed circular oscillation. In this equilibrated system, each circle oscillation (together with its x-value) is now an indifferently stable behaviour mode of the system. Furthermore, small noise added to the system evolution will send the oscillation amplitude (and x) on a slow random walk.

Now assume that the two systems shown in Fig. 5 were used as sine-wave generators, by extracting the y-coordinate ($y = r\cos\theta$) as the output signal. The original (un-equilibrated) system at the top will generate a stable oscillation with a stable unit amplitude, i.e., from any initialisation this un-equilibrated system will converge to the state where $x = 0$ and where the system exhibits a stable oscillation with unit amplitude. The equilibrated companion will likewise stably generate oscillations, but their amplitude will only be neutrally stable and would go through a random walk in the presence of state noise. Now, if the objective were to obtain an oscillator whose amplitude can be *controlled* by an external controller, it seems intuitive that the equilibrated system should be easier to control than the un-equilibrated one. The equilibrated system already "knows how" to oscillate at different amplitudes. In order to make it exhibit one of its "stored" oscillation pattern, the external controller only has to gently steer the x-value to the appropriate value. Since the x-dynamics is neutrally stable, this can be achieved with minimal control energy (with zero magnitude in the adiabatic limit). The un-equilibrated system seems harder to control: the native x-dynamics, which always tries to push x toward 0, has to be overcome by a suitable counter-action - for instance, by regulating x with a proportional controller of high enough gain. This requires that the system has to undergo a control input of significant magnitude. While for the simple system that we used here for illustration this would pose no real obstacle, it is not trivial to steer the dynamics of a very complex system (e.g., a high-dimensional RNN) by large-magnitude control input.

The discussed example is about an oscillatory system where the target characteristic, which we aim to modulate, is amplitude. We used this example because amplitude can be more readily visualised than frequency. The topic of this article is however frequency control. It should be clear that the same story could be told for that case. The "raw" system would then be given, for instance, by $\dot{r} = \tau(-r + 1)$, $\dot{\theta} = \exp(x)$, $\dot{x} = -cx$, while the equilibrated system would again have $\dot{x} = 0$.

5.2 Equilibrating ESN pattern generators

Similar to the synthetic equilibrated example, we want to build an equilibrated ESN pattern generator which internally hosts a collection of oscillators of different frequencies. By externally driving it to a particular different condition, the pattern generator can spontaneously produce the oscillating output with a particular fixed frequency. To construct such an equilibrated ESN system, we use the FORCE learning procedure (Section 2)

to build an ESN pattern generator of $1,000$ neurons and train the system's readout weights $\mathbf{W}_{\text{out}}$ with a specially designed target signal, namely, an oscillation with a gradually changing frequency: $y_{desired}[k] = \sin(0.075s[k]k)$ with $s[k]$ a time dependent scaling factor which linearly increases from 1 to 3 and k from 1 to $10,000$. Fig. 6 depicts this target signal.

An intuitive explanation for this training scheme is that the network is trained to oscillate in different frequencies. Consequently, when the training is successful the system will be able to oscillate in different frequencies by itself. In other words, the network should contain a collection of oscillators of different frequencies. If such a network can be made to oscillate at any frequency in the training range, with frequency neutrally stable (similar to amplitude in the synthetic example of Section 5.1), this would demonstrate that we have an instantiation of equilibration.

However, to our knowledge, with the current training methods for large RNNs, only an approximately equilibrated system can be realised. Its hallmark would be that when it is initialised to a particular frequency by external driving, after releasing it from the driving signal its frequency will *slowly* migrate toward a preferred frequency. In terms of our synthetic example (see Section 5.1): the system from Eqns. 9 – 11 would be *approximately* equilibrated if Eqn. 11 would for instance read $\dot{x} = -\varepsilon x$ for some small ε (or any other slow relaxation dynamics for x).

In our experiments we set the approximately equilibrated ESN pattern generator to six different initial oscillation conditions by driving the reservoir through feedback weights $\mathbf{W}_{\text{fb}}$ with external oscillations of different but fixed frequency (period lengths between 28 and 87, e.g., the highest and lowest frequency used in the training sequence). Then we let the ESN pattern generator freely run by itself which results into the so-called *cueing plots* (see for example Fig. 7). Indeed, we then observe that the equilibration-trained reservoir slowly converges to a fixed frequency independent of the initial condition we chose. The bottom plot in Fig. 7 shows the output of an approximately equilibrated ESN pattern generator starting from high frequency initial oscillation condition. In contrast, the top plot in Fig. 7 shows the behaviour of the same network that was trained on a single frequency. Note again that the details (parameters) of the experimental setup are documented in Section 6.

It is interesting to compare the reservoir trajectories of an ESN pattern generator under different conditions, namely, (i) driven by external input (Section 4), (ii) trained with an equilibration training method (trained with a gradually changing frequency), and (iii) trained

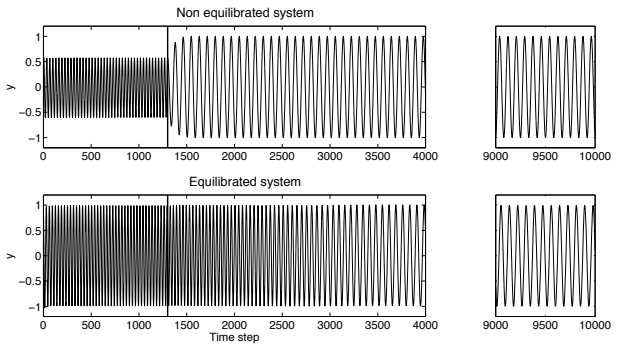

Fig. 7 The behaviour of a non-equilibrated (top) and an approximately equilibrated (bottom) ESN pattern generator. Both systems were primed by driving them through their output feedback with a high-frequency oscillation. After step $1,250$, each system was unclamped and let run freely. The non-equilibrated system changes abruptly into its preferred frequency, while the approximately equilibrated system exhibits a slow drift of frequency toward its preferred frequency.

with the basic learning method (Section 2, trained with on one frequency). Fig. 8 provides a PC-projected phase portrait view on the differences between these conditions. When the approximately equilibrated setup is cued with a high-frequency driving signal and then released, its circling trajectory moves "monotonically" and slowly toward the preferred (slower) frequency. This behaviour is very similar to the setup while the reservoir was driven by an external (gradually) frequency changing oscillation. The non-equilibrated setup, when started from the same fast oscillation, displays a trajectory which quickly moves from the initial cycle to the final one, but on the way changes direction (moves first upwards, then downwards in the vertical plotting dimension). If the aim is to control frequency, it seems plausible that a *smooth monotonic* change of geometrical location leads to easier control mechanisms than non-monotonic changes of geometrical localisation of trajectories. All of this would deserve a more in-depth study, but for the present purpose we are contented with these intuitive impressions.

Here we have effected (approximate) equilibration by only training the readout weights of the ESN. In some of our earlier work (Jaeger, 2010; Li and Jaeger, 2011), approximately equilibration of an ESN pattern generator was achieved through recomputing all the system weights ($\mathbf{W}_{\text{res}}$, $\mathbf{W}_{\text{out}}$, $\mathbf{W}_{\text{fb}}$ and $\mathbf{W}_{\text{bias}}$). A systematic investigation of training methods for equilibration remains for future work.

6 Simulation results

We first exhibit a typical example of a working frequency control (Section 6.1) and then report findings

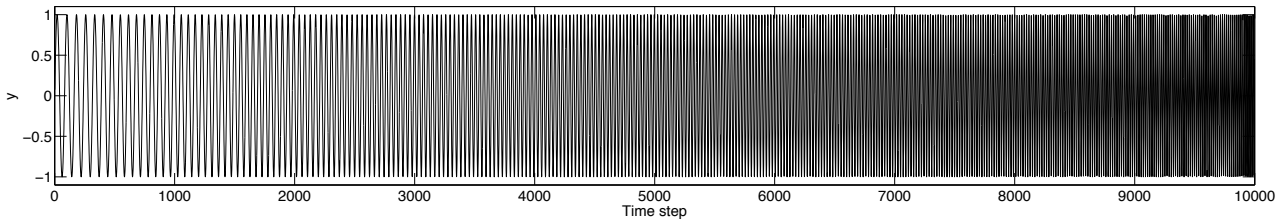

Fig. 6 The signal used for equilibration training: $y_{desired}[k] = \sin(0.075s[k]k)$, with $s[k]$ a time dependent scaling factor which linearly increases from 1 to 3.

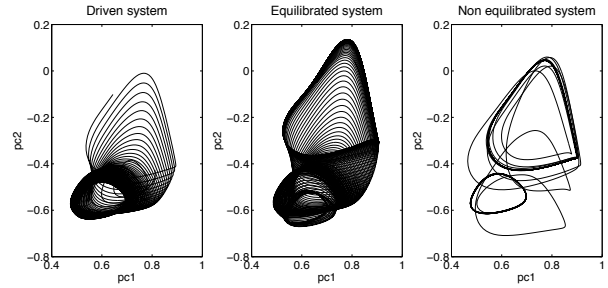

Fig. 8 Comparison of the state evolution (projections of the first and second principal components of the reservoir's states) of a driven (left), approximately equilibrated (middle) and a non-equilibrated (right) network. For explanation see text.

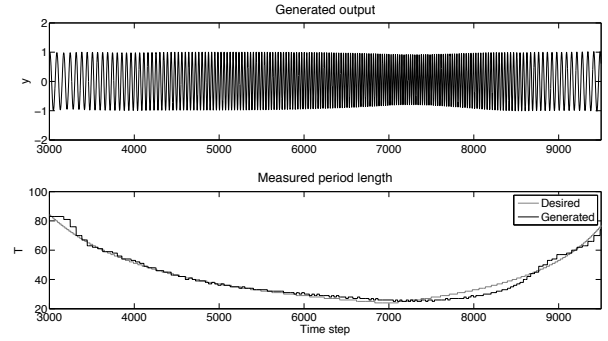

Fig. 9 Output of a frequency adjustable system (top). The desired frequency (light gray, bottom) goes through a slow dip and is tracked reasonably well (black curve, bottom).

from a survey of trials with a large number of randomly created reservoir systems (Section 6.2).

6.1 A typical example

Here, we use the reservoir system that was used throughout this paper as an example to illustrate its frequency modulating capabilities. To recapitulate, this reservoir system has size $N = 1,000$, a leak-rate set at 0.1 and weights $\mathbf{W}_{\text{res}}$, $\mathbf{W}_{\text{fb}}$ and $\mathbf{W}_{\text{bias}}$ respectively sampled from normal distributions $\mathcal{N}(0,1)$, $\mathcal{N}(0,1.5)$ and $\mathcal{N}(0,0.5)$. The scaling factors were obtained by manual tuning.

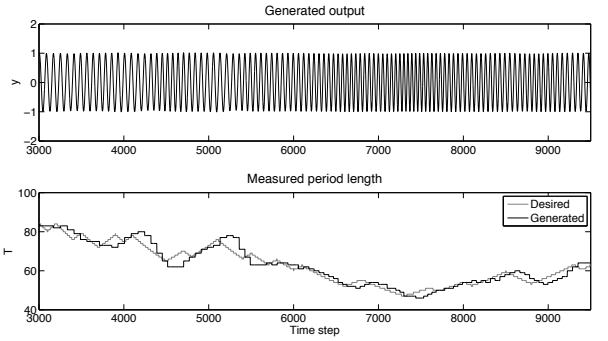

Fig. 10 Output of a frequency adjustable system (top). The control target follows a random walk (bottom, target: light grey, measured frequency: black).

After creation of the weight matrices, $\mathbf{W}_{\text{res}}$ was rescaled such that the spectral radius ρ was set at 1.8. The readout weights $\mathbf{W}_{\text{out}}$ were trained for equilibration with a 10,000 step oscillation $y_{desired}[k] = \sin(0.075s[k]k)$ whose frequency was linearly sped up by a factor of 3 by ramping $s[k]$ from 1 to 3 (see Fig. 6 for an illustration). For this, we followed the procedure outlined in Section 2. After this preparation, we verified that the equilibration was successful by visually inspecting its cueing plots. In these plots the frequency must change gradually (i.e., like the one in the bottom plot in Fig. 7).

After having obtained an approximately equilibrated reservoir, we computed the control bias $\mathbf{W}_{\text{c}}$ and activated the control loop, as described in section 3. The proportional control gain c_{P} was determined by coarse hand-tuning. For this $c_{\text{P}} = 1$ was considered as a starting point after which c_{P} was decreased or increased in order to achieve a more accurate tracking or larger control range, respectively. Figs. 9 and 10 demonstrate that frequency could be controlled in a range of a factor 3.

6.2 From equilibration to modulatability

The equilibration procedure does not always result in a system which behaves as "smoothly" as shown in Figs. 7 (bottom) and 8 (middle). We found two conditions which impede subsequent controllability. First,

it occurs that the equilibration-trained system escapes into aperiodic (presumably chaotic) or fixed-point dynamics. These systems are not able to maintain a fixed frequency oscillation and consequently are not suitable. Second, even if the equilibration-trained system exhibits periodic behaviour and has only a single preferred frequency, convergence toward this frequency from other cueing frequencies may be too strong (e.g., Figs 7 (top) and 8 (right)). For these systems, the equilibration worked out to an insufficient degree. Consequently, these systems can not be frequency-controlled by using a simple linear controller.

We carried out a two-stage screening, starting from a population of 20,000 randomly created reservoirs. From these, we first discarded the ones that showed aperiodic or fixed-point behaviour, and in a second step the ones that showed an insufficient degree of equilibration. The ones that were left over were tested for controllability.

Specifically, the 20,000 raw reservoirs, similar to subsection 6.1, all had 1,000 neurons and a leak-rate of 0.15. The spectral radius ρ, feedback scaling o and bias β scaling was set at 1.8, 1.5 and 0.5 respectively. Each of these reservoirs was then equilibration-trained as described in subsection 5.2. Each system thus prepared was then cued with six oscillations whose periods spanned the training range of 28 to 87 steps. After cueing for 1,250 steps, the system was let run freely until 10,000 time steps were passed. We assumed that convergence to any preferred dynamic mode would occur within this runtime.

For each pattern generator we first checked whether all of its six free runs converged to a fixed frequency oscillation. Therefore, we removed all systems which free run let to a fixed point or aperiodic behaviour.

Next, we checked whether the equilibration had worked out in the desired fashion, ideally looking like the bottom plot in Fig. 7.

In order to automatically glean systems which have the desired "smooth and slow" transient from a cued period length toward the preferred one, we employed the following heuristic. From among the six cueing runs we used the one that started from the highest (period 28) and the lowest (period 87) frequency. From the generated time series, we obtained the evolution of period lengths $T_1, ..., T_i, ..., T_K$ (e.g., Fig. 11) with T_K the last measured period length from the free run. The sequence $T_1, ..., T_i, ..., T_K$ was then smoothed by a moving average filter. From the smoothed sequence, we calculated heuristic measures Ξ, Ψ, Υ for speed of convergence,

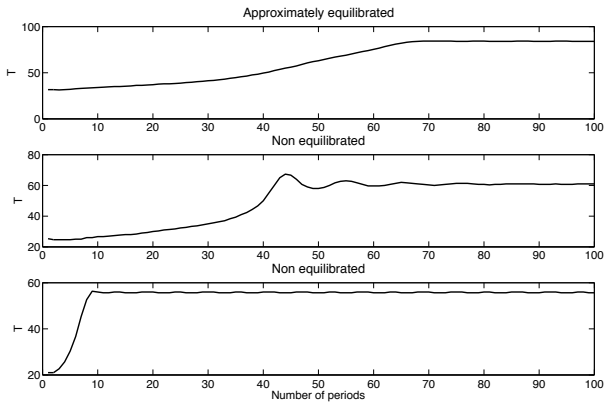

Fig. 11 Three examples of period length transient dynamics from a (cued) short period to a (preferred) longer period. Plots show progression of period length against number of periods. Only the example in the top plot exhibits the desired quasi-linear ramping-up.

curvature and monotonicity, respectively:

$$\Xi = \max|T_{i+1} - T_i| \text{ for } \forall i = 1...K-1 \quad (12)$$

$$\Psi = \max|(T_{i+1} - T_i) - (T_i - T_{i-1})| \quad (13)$$
$$\text{for } \forall i = 2...K-1$$

$$\Upsilon = \sum_{i=2}^{K-1} |\text{sgn}(T_{i+1} - T_i) - \text{sgn}(T_i - T_{i-1})| \quad (14)$$

We considered a period sequence $T_1, ..., T_i, ..., T_K$ as proof of a successful equilibration if it was monotonic (in the sense that $\Upsilon = 0$), not too steep ($\Xi < 2.0$) and not too curved ($\Psi < 0.2$).

Out of the 20,000 systems, 1,573 remained after our selection procedure. All 1,573 systems were found frequency adjustable which was determined by the mean absolute error (MAE) between the desired period lengths and the observed period lengths. The same target as in Fig. 9 was used and the threshold for acceptance was set at MAE = 4.0. In roughly half of the cases the default initial proportional gain c_P, which was obtained after rough manual tuning in the example from Section 6.1, was sufficient to meet the objective. In the other cases, manual tuning of c_P was necessary.

These empirical results show that frequency modulation can be realised in high dimensional non-linear pattern generators provided that they are successfully (approximately) equilibrated.

7 Discussion

In this contribution we argued that

– biological and robotic pattern generators need to be adjustable in many ways, and

– modulation of speed differs fundamentally from modulation of other, "geometric" characteristics, and
– richly trainable and adjustable pattern generators are likely to require neural networks of substantial size, –
– which leads to the question of generic mechanisms for frequency control of neural pattern generators.

Here we considered the special case of oscillatory networks, and demonstrated that their frequency can be made controllable by

– training the network in a way that it is forced to adapt its weights to accommodate to a range of frequencies ("equilibration"),
– which, if is successful, results in a monotonic interdependence between temporal and spatial properties of the network dynamics,
– which in turn can be exploited for controlling time by spatial state shifts through a bias term in a proportional control loop.

Our study is a proof of principle, with many design decisions made ad hoc. Variations and extensions offer themselves in many ways, e.g., improved control schemes (for example, PID controllers instead of simple P controllers, or making the bias weights $\mathbf{W}_c$ frequency-dependent), more sophisticated observers for frequency, other equilibration methods (for instance, training all reservoir weights instead of training only the output weights, as done in Jaeger (2010)), investigating other signal shapes, etc. Furthermore, in our groups we also investigate altogether different learning architectures for making frequency adjustable ESN pattern generators. For instance, in (Jaeger, 2007) a multifrequency generator is directly trained as an open-loop control system. Thus, the present study does not claim to offer *the* solution for constructing large frequency adjustable oscillatory neural networks. We nonetheless consider the following as relevant contributions:

– pointing out the importance and difficulty of the *neural speed control* problem in the first place,
– clarifying the existence and functional role of *equilibration* of dynamics for making selected characteristics robustly controllable, and
– demonstrating that a temporal – spatial interdependency of trajectories can be shaped and exploited for control.

Biological evolution is likely to adopt whatever works well. Biological research has identified frequency control mechanisms in small CPG model systems which rely on specific, idiosyncratic mechanisms. Roboticists simply adapt the speed of ODE based pattern generators by changing the time constants. The work presented in this article does not aim at replacing or refuting any of these insights or techniques. However, in higher cortical processing domains in biological systems, or in flexibly trainable RNN-based robotic control modules, we perceive an arena where generic neural speed control schemes such as the one illustrated in this work might become important.

Acknowledgements The authors would like to thank the anonymous reviewers for their constructive comments that helped improving this manuscript. The research leading to the results presented here has received funding from the European Community's Seventh Framework Programme (EU FP7) under grant agreement n.248311 *Adaptive Modular Architecture for Rich Motor Skills* (AMARSi).

References

Briggman K, Kristan Jr W (2006) Imaging dedicated and multifunctional neural circuits generating distinct behaviors. The Journal of Neuroscience 26:10,925–10,933

Briggman K, Kristan Jr W (2008) Multifuctional pattern-generating circuits. Annual Review of Neuroscience 31:271–294

Buchli J, Righetti L, Ijspeert A (2006) Engineering entrainment and adaptation in limit cycle systems. Biological Cybernetics 95:645–664

Büschges A, Scholz H, El Manira A (2011) New moves in motor control. Current Biology 21:R513–R524

Caluwaerts K, D'Haene M, Verstraeten D, Schrauwen B (2013a) Locomotion without a brain: Physical reservoir computing in tensegrity structures. Artificial Life 19:35–66

Caluwaerts K, wyffels F, Dieleman S, Schrauwen B (2013b) The spectral radius remains a valid indicator of the echo state property for large reservoirs. In: Proceedings of the International Joint Conference on Neural Networks

Cruse H, Brunn D, Bartling C, Dean J, Dreifert M, Kindermann T, Schmitz J (1995) Walking: A complex behavior controlled by simple networks. Adaptive Behavior 3(4):385–418

Curtu R, Shpiro A, Rubin N, Rinzel J (2008) Mechanisms for frequency control in neuronal competition models. SIAM Journal on Applied Dynamical Systems 7:609–649

Daun S, Rubin J, Rybak I (2009) Control of oscillation periods and phase durations in half-center central pattern generators: a comparative mechanistic analysis. Journal of Computational Neuroscience 27:3–36

Dean J, Kindermann T, Schmitz J, Schumm M, Cruse H (1999) Control of walking in the stick insect: From

behavior and physiology to modeling. Autonomous Robots 7:271–288

Fukuoka Y, Kimura H, Cohen A (2003) Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts. The International Journal of Robotics Research 22:187–202

Grillner S (1985) Neurobiological bases of rhythmic motor acts in vertebrates. Science 228:143–149

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

Ijspeert A, Crespi A, Ryczko D, Cabelguen JM (2007) From swimming to walking with a salamander robot driven by a spinal cord model. Science 315(5817):1416–1420

Ijspeert A, Nanaishi J, Hoffmann H, Pastor P, Schaal S (2013) Dynamical movement primitives: Learning attractor models for motor behaviors. Neural Computation 25:328–373

Jaeger H (2001) The "echo state" approach to analysing and training recurrent neural networks. Gmd report 148, German National Research Center for Information Technology

Jaeger H (2002) A tutorial on training recurrent neural networks, covering bppt, rtrl, ekf and the "echo state network" approach. Gmd report 159, International University Bremen

Jaeger H (2007) Echo state network. In: Scholarpedia, vol 2, p 2330, URL `http://www.scholarpedia.org/article/Echo_State_Network`

Jaeger H (2010) Reservoir self-control for achieving invariance against slow input distortions. Technical report 23, Jacobs University Bremen

Jolliffe I (2005) Principal Component Analysis. Encyclopedia of Statistics in Behavioral Science.

Laje R, Buonomano DV (2013) Robust timing and motor patterns by taming chaos in recurrent neural networks. Nature Neuroscience 16(7):925–933

Li J, Jaeger H (2011) Minimal energy control of an ESN pattern generator. Technical report 26, Jacobs University Bremen, School of Engineering and Science

Lukoševičius M (2012) A practical guide to applying echo state networks. Neural Networks: Tricks of the Trade, Reloaded 7700:659–686

Lukoševičius M, Jaeger H (2009) Reservoir computing approaches to recurrent neural network training. Computer Science Review 3:127–149

Mayer NM, Browne M (2004) Echo state networks and self-prediction. In: Biologically Inspired Approaches to Advanced Information Technology, LNCS, vol 3141, Springer Verlag Berlin / Heidelberg, pp 40–48

Nakanishi J, Morimoto J, Endo G, Chenga G, Schaal S, Kawato M (2004) Learning from demonstration and adaptation of biped locomotion. Robotics and Autonomous Systems 47:79–91

Reinhart R, Steil J (2011) A constrained regularization approach for input-driven recurrent neural networks. Differential Equations and Dynamical Systems 19(1–2):27–46

Reinhart R, Steil J (2012) Regularization and stability in reservoir networks with output feedback. Neurocomputing 90:96–105

Reinhart R, Steil JJ (2008) Recurrent neural associative learning of forward and inverse kinematics for movement generation of the redundant pa-10 robot. In: Proceedings of the ECSIS Symposium on Learning and Adaptive Behaviors for Robotic Systems, pp 35–40

Rolf M, Steil JJ, Gienger M (2010a) Goal babbling permits direct learning of inverse kinematics. IEEE Transactions on Autonomous Mental Development 2(3):216–229

Rolf M, Steil JJ, Gienger M (2010b) Learning flexible full body kinematics for humanoid tool use. In: Proceedings of the International Symposium on Learning and Adaptive Behavior in Robotic Systems

Schrauwen B, Defour J, Verstraeten D, Van Campenhout J (2007) The introduction of time-scales in reservoir computing, applied to isolated digits recognition. In: Proceedings of the International Conference on Artificial Neural Networks

Sussillo D, Abbott L (2012) Transferring learning from external to internal weights in echo-state networks with sparse connectivity. PLoS ONE 7(5):e37,372

Sussillo D, Abbott LF (2009) Generating coherent patterns of activity from chaotic neural networks. Neuron 63:544–557

Verstraeten D, Schrauwen B, D'Haene M, Stroobandt D (2007) An experimental unification of reservoir computing methods. Neural Networks 20:391–403

Waegeman T, Schrauwen B (2011) Towards learning inverse kinematics with a neural network based tracking controller. In: Lecture Notes in Computer Science, vol 7064, pp 441–448

Waegeman T, wyffels F, Schrauwen B (2012a) Feedback control by online learning an inverse models. IEEE Transactions on Neural Networks and Learning Systems 23:1637–1648

Waegeman T, wyffels F, Schrauwen B (2012b) A recurrent neural network based discrete and rhythmic pattern generator. In: Proceedings of the European Symposium on Artificial Neural Networks

Waegeman T, wyffels F, Schrauwen B (2012c) Towards a neural hierarchy of time scales for motor control. In: Lecture Notes in Computer Science, vol 7426, pp 146–155

Wrede S, Johannfunke M, Nordmann A, Rüther S, Weirich A, Steil J (2010) Interactive learning of inverse kinematics with nullspace constraints using recurrent neural networks. In: Proceedings of the 20th Workshop on Computational Intelligence

wyffels F, Schrauwen B (2009) Design of a central pattern generator using reservoir computing for learning human motion. In: Proceedings of the ECSIS Symposium on Advanced Technologies for Enhanced Quality of Life, pp 118–122

wyffels F, Schrauwen B, Stroobandt D (2008) Stable output feedback in reservoir computing using ridge regression. In: Proceedings of the International Conference on Analog Neural Networks

Yildiz I, Jaeger H, Kiebel S (2012) Re-visiting the echo state property. Neural Networks 35:1–9

Zhang C, Lewis T (2013) Phase response properties of half-center oscillators. Journal of Computational Neuroscience 35:55–74