EU FP7

AMARSi

Learning Minimal Energie Controller

ICT-248311

D 5.1

March 2011 (12 months)

Minimal Energy Control of an ESN Pattern Generator

Authors: Jiwen Li, Herbert Jaeger

Due date of deliverable	28th February 2011
Actual submission date	28th February 2011
Lead Partner	TUG
Revision	Final
Dissemination level	Partner Report

JACOBS
UNIVERSITY

Jiwen Li, Herbert Jaeger

Minimal Energy Control of an ESN Pattern Generator

Technical Report No. 26
February 2011

School of Engineering and Science

Minimal Energy Control of an ESN Pattern Generator

Jiwen Li, Herbert Jaeger

Jacobs University Bremen
School of Engineering and Science
Campus Ring
28759 Bremen
Germany

E-Mail: {*j.li, h.jaeger*}*@jacobs-university.de*
http://minds.jacobs-university.de

Abstract

In this report we present a method of adding a feedback control mechanism to an echo state network (ESN) pattern generator in order to modulate its output patterns with the purpose of tracking slowly varying control targets, e.g. shift, amplitude, or frequency of an oscillatory pattern. A proof-of-principle case study is presented where a basic ESN is trained to produce a stable sinewave oscillation with fixed shift, amplitude and frequency. With the controller in place, the system demonstrates that the shift, amplitude and frequency of the produced sine waveform can be modulated simultaneously by suitably generated slow varying control signals inserted into the network. Furthermore, an equilibration procedure is introduced to relearn ESN weights such that the equilibrated ESN pattern generator can approximately reproduce the reservoir dynamics across the controllable range, with the feedback control loop switched off. As a result, when reconnecting the feedback control loop to the equilibrated ESN, the energy of the control signals are many orders of magnitude smaller compared to the native system.

1 Introduction

Neural periodic pattern generators [2] are widely assumed to exist in animals, serving a multitude of functions, from generating heartbeat to complex locomotion patterns. The term *central pattern generator (CPG)* is commonly used for such oscillator circuits, although the usage of this term is not clear-cut and, somewhat confusingly, also refers to neural pattern generators which are not located

"centrally" in the vertebrate spine or in central insect ganglions, but instead are locally attached to their target organs, e.g. the heart or the stomach. Here we abstractly consider only neural periodic pattern generators which indeed are located centrally, participate in motor pattern generation, can be richly modulated from higher up in the CNS hierarchy, and and can be shaped by learning [3]. The use of the term, "central pattern generator", should be unproblematic here.

Formal or computational models of modulatable CPGs often take the form of ordinary differential equations (ODEs) describing the dynamics of a small number of state variables, instantiating one of the standard nonlinear oscillator equations. The modulation is then effected by regulating suitable control parameters in the system equations. A typical example is the canonical nonlinear oscillation model [4], which has the form of

$$\begin{aligned} \dot{z} &= -\frac{\mu}{E_0}(E - E_0)z - k^2 u, \\ \dot{u} &= z \end{aligned}$$

and which shows a closed cyclic trajectory $\frac{z^2}{2} + \frac{k^2 u^2}{2} = E_0$ (const) on the phase plane. Here the parameter k corresponds to the frequency of the oscillator, and E_0 corresponds to the desired total energy and determines the amplitude of the oscillation. μ determines the convergence rate to the limit cycle.

Mathematical/computational models of this kind may be appropriate for "low-level", simple and stereotyped periodic dynamics, and they may be realized in animals by "hard-coded" neural circuits of small size, like in heartbeat [9] or the well-known stomatogastric pattern generator [1]. However, it is less clear whether such ODE-based, low-dimensional models can capture the nature of CPGs which are assumed to reside in the vertebrate spine (or even higher levels of vertebrate neural systems). First, these CPGs are constituted by a large number of neurons, rendering these biological systems high-dimensional, at least in a formal sense. Second, the modulations can often be adapted by learning along numerous control target dimensions, even dimensions not foreseen by evolution (e.g. a human learning dance patterns). This "open-endedness" seems difficult to capture by modulation mechanisms based on a limited number of specific control parameters built into one of the classical, low-dimensional oscillator equations.

In this report we introduce a generic method which allows us to train a large recurrent neural CPG network to become modulatable along control dimensions which are not a priori limited in number or predetermined in character. Our architecture is comprised of the following components.

- The core component is a discrete-time recurrent neural network (RNN) of the "Echo State Network"(ESN) type [5, 8] with output feedback, with one or several output nodes which are trained to generate a stable, periodic,

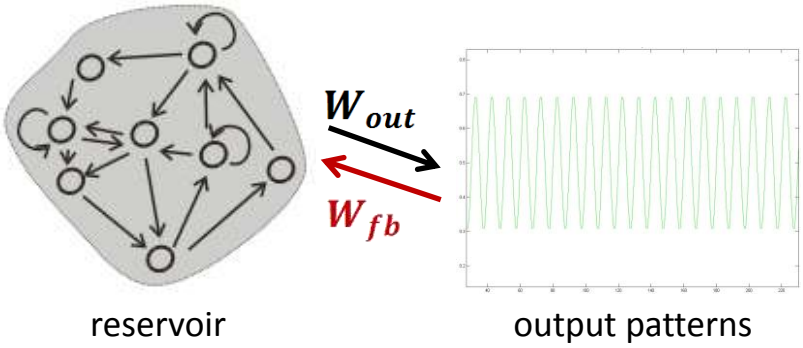


Figure 1: The baseline periodic pattern generator, implemented by an ESN.

"baseline" pattern (see Figure 1). Following the tradition of reservoir computing, we will call the "body" of this RNN (i.e., excluding the output units) a *reservoir*. In formal terms, the dynamics embodied by this core system is composed of one global periodic attractor.

- From this periodic output, arbitrary *slow observables* can be extracted, e.g. shift (offset of the mean from zero), amplitude, or frequency. We call an observable *slow* if its value changes on a timescale that is separated from the baseline oscillation timescale by at least one base-10 order of magnitude. How the observers for the slow observables are implemented is not a crucial part of our model – any measurement subsystem which transforms an oscillation into a slow observable can be used. This may range from simple smoothing to frequency measurements to complex observables obtained by involved mechanisms like slow feature analysis.

- The objective is to make any such slow observable controllable in the sense of tracking control. That is, we assume that for each such slow observable, a slow reference signal is given. Contrasting this reference with the slow observable yields a likewise slow error signal.

- The error signal (a vector of slow errors, one per controlled observable) is fed to a feedback controller $\mathbf{C}$ which issues a control input $\mathbf{c}(n)$ into the

reservoir. Concretely, it will turn out that $\mathbf{c}(n)$ may take the simple form of a (slowly varying) bias vector added to the reservoir dynamics.

Furthermore, we introduce an equilibration method to "encode" the effect of the feedback control mechanism mentioned above into the reservoir. This is done by harvesting the control-input modulated reservoir dynamics and relearning the reservoir weight $\mathbf{W}$ and feedback weight $\mathbf{W}_{fb}$. By using these new weights, the new network, which we call *equilibrated*, can approximately reproduce the reservoir dynamics of the old network without using the control loop. Finally, by using these new weights and the feedback control loop together, the same tracking control tasks can be achieved using very small feedback control energy.

In the remainder of the report we use a step-by-step example to demonstrate how a sinewave signal generator network can be obtained by ESN training (Section 2), and then subsequently be augmented by a feedback controller to become modulatable with respect to shift, amplitude, and frequency (Section 3). In Section 4 we explain the principle of equilibration, and demonstrate how it is implemented in our example. Finally in section 5, we summarize our methods and architecture, points out the current problems of the methods, and sketch the further research we want to pursue.

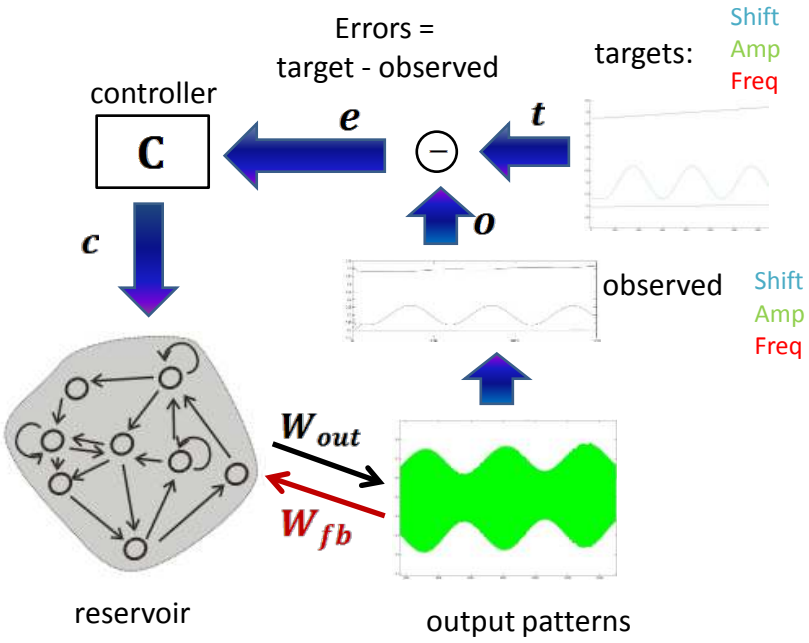


Figure 2: Overview of architecture. For explanation see text.

2 Training an ESN as a periodic oscillator

The proposed strategy to construct a modulatable ESN pattern generator proceeds in two stages. In the first stage, an ESN with output feedback is trained, in the standard reservoir computing way, to obtain a stable oscillator with fixed shift, amplitude and frequency. In the second stage, a feedback tracking controller C is trained and linked into the system. Throughout this technical report, we will consider a single demo example with scalar output.

We consider a standard ESN with N reservoir units generating scalar output $y(n)$ and feeding it back to the reservoir:

$$\begin{aligned} \mathbf{x}(n+1) &= \tanh\left(\mathbf{W}\mathbf{x}(n) + \mathbf{W}^{fb}y(n)\right), & (1) \\ y(n) &= \sigma\left(\mathbf{W}^{out}\mathbf{x}(n)\right), & (2) \end{aligned}$$

where $\mathbf{W}$ is the reservoir weight matrix (size $N \times N$), $\mathbf{W}^{fb}$ is the feedback weight matrix (size $1 \times N$), $\mathbf{W}^{out}$ is the output weight matrix (size $N \times 1$), and σ is the logistic sigmoid $\sigma(a) = 1/(1 + \exp(-a))$. Concretely, for our demo we choose $N = 400$, construct $\mathbf{W}$ to have a connectivity of about 20%, scale it to a spectral radius of 1.0, and sample the feedback weights $\mathbf{W}^{fb}$ from a uniform distribution in $[-0.5, 0.5]$. These are ad-hoc settings. Throughout this report we assume that the reader is familiar with the basic concepts of reservoir computing, and we will not detail out standard procedures from that field.

In our demo example, we use a 15000 time-step sine oscillation as our training signal. This sine signal has a period of 10 time steps, an amplitude of 0.2 and a shift of 0.5. We will use notation $u(n)$ for the training signal. Figure 3 illustrates the training signal.

The network outputs are computed to minimize the mean square error on a one-timestep prediction task, i.e. the teacher signal is $teach(n) = u(n-1)$. For training, the reservoir is driven with the input signal through $\mathbf{W}^{fb}$ for 15000 time steps, and then output weights are computed from the harvested reservoir states (dismissing the first 1000 ones to account for initial state washout) by ridge regression with a regularization constant $\alpha = 0.05$. This results in a normalized mean square training error (NRMSE) of 0.000376, with a mean absolute output weight size of 0.104. For testing, the reservoir is left running freely for another 15000 time steps. The resulted output $y(n)$ shows a stable oscillation of sine waveform, which imitates the training signal $u(n)$. Figure 4 illustrates the output produced from the trained network.

It becomes clear from the training NRMSE and an inspection of Figure 4(b) that our ESN can be successfully trained to generate the desired sine oscillation.

Next we install an observer to track shift, amplitude, and frequency of the output signal $y(n)$. Exploiting the fact that the output is known to be close to a (discretely sampled) sinewave, for simplicity we base this observer on a localization of peaks and troughs. At each timestep, the observer generates a local estimate of

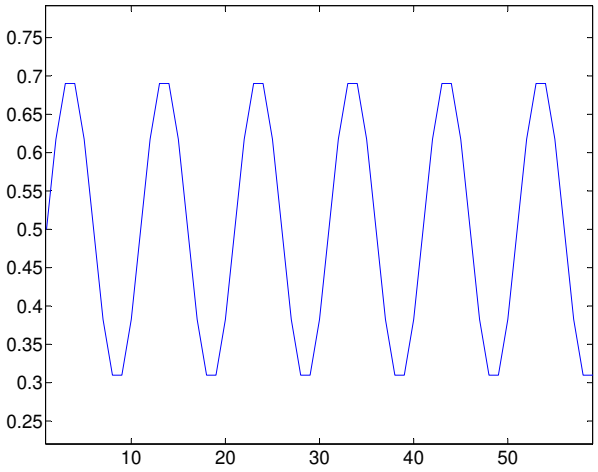


Figure 3: Closeup on the input signal used in demo example.

the current shift, amplitude, and frequency $m_s(n)$, $m_a(n)$, $m_f(n)$. The observer maintains an internal 6-dimensional state $\mathbf{s}(n-1) = [y(n-1)\ y(n-2)\ p(n-1)\ v(n-1)\ l_t(n-1)\ m_f(n-1)]$. Here, $y(n-1)$ represents the output value in the previous time step, $y(n-2)$ represents the output value in the pre-previous time step, $p(n-1)$ represents the most recently detected output peak value until the (n-1)th time step, $v(n-1)$ represents the most recently detected trough value until the (n-1)th time step, $l_t(n-1)$ represents the time period since the last detected peak until the (n-1)th time step, and $m_f(n-1)$ records the most recently computed frequency of the output until (n-1)th time step.

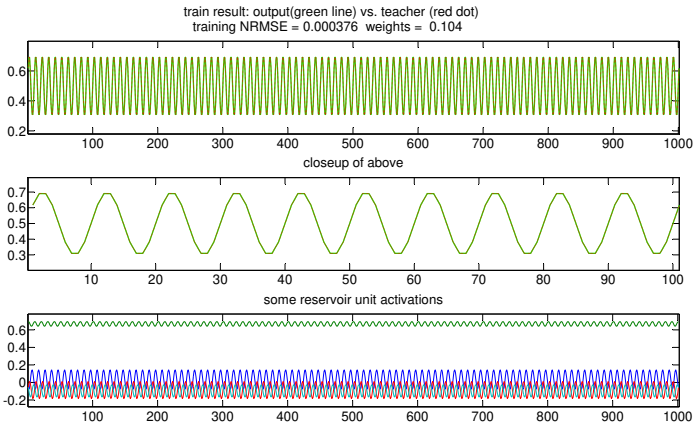
With the current output $y(n)$ and the previous observer internal state $\mathbf{s}(n-1)$, the current measurement of shift $m_s(n)$ is computed as following:

1. Detect whether the previous output value $y(n-1)$ is a peak, by the following condition: if $(y(n) < y(n-1)) \wedge (y(n-2) < y(n-1))$, then $y(n-1)$ is a peak. If $y(n-1)$ is a peak, $p(n) = y(n-1)$, else $p(n) = p(n-1)$.

2. Detect whether the previous output value $y(n-1)$ is a trough, by the following condition: if $(y(n) > y(n-1)) \wedge (y(n-2) > y(n-1))$, then $y(n-1)$ is a valley. If $y(n-1)$ is a valley, $v(n) = y(n-1)$, else $v(n) = v(n-1)$.

3. Compute the current measurement on shift as $m_s(n) = \frac{p(n)+v(n)}{2}$.

Similarly, to compute the current measurement of amplitude $m_a(n)$, we utilize the above steps (1) and (2) to update the current peak value $p(n)$ and the current trough value $v(n)$. Then we compute the current measurement on amplitude as $m_a(n) = \frac{|p(n)-v(n)|}{2}$.



(a) (b)

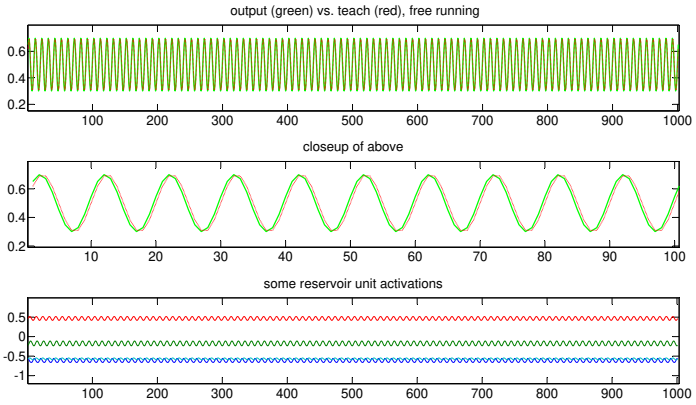


Figure 4: Training an ESN as a periodic pattern generator. (a) Training phase, (b) free running phase. Top: teacher vs. network output. Middle: closeup of the top panel. Bottom: four traces of reservoir unit activations.

To compute the current measurement of frequency $m_f(n)$, we use the following procedure:

1. Use the peak detection method mentioned in step (1) of computing $m_s(n)$ to judge whether the previous output value $y(n-1)$ is a peak.

2. If $y(n-1)$ is a peak, the time slice from last peak $l_t(n-1)$ is used to update the current frequency as $m_f(n) = \frac{1}{l_t(n-1)}$. Else, $m_f(n) = m_f(n-1)$

3. Update the time slice since the last peak $l_t(n)$ as: if $y(n-1)$ is a peak, $l_t(n) = 1$. Else, $l_t(n) = l_t(n-1) + 1$.

Finally, after all three measurements $m_s(n)$, $m_a(n)$ and $m_f(n)$ are computed for the current time step n, we update the observer internal state vector $\mathbf{s}(n) = [y(n)\ y(n-1)\ p(n)\ v(n)\ l_t(n)\ m_f(n)]$.

The local estimates $\mathbf{m}(n) = [m_s(n), m_a(n), m_f(n)]'$ (we use $'$ to denote transpose) will be jittery due to sampling inaccuracies imposed on this simple peak and trough location. We therefore subsequently feed $\mathbf{m}(n)$ through a simple smoother, obtaining the final measurements $\mathbf{o}(n)$:

$$\mathbf{o}(n+1) = 0.99 \cdot \mathbf{o}(n) + 0.01 \cdot \mathbf{m}(n+1). \tag{3}$$

Next, with the observer module added in, we rerun the ESN another 15000 time steps to generate an output sequence, and simultaneously monitor the shift, amplitude and frequency of the output sequence, obtain traces of the three components of $\mathbf{o}(n)$, and compute the means of the three traces. They are $meanMShift = 0.5$, $meanMAmp = 0.2$, $meanMFreq = 0.1$, as anticipated.

3 Training and testing the controller

Our goal is to create a simple linear proportional feedback controller whose purpose is to modulate the shift, amplitude and frequency of the output sine waveform with the purpose of tracking the corresponding control targets.

First we describe the control targets we use in the demo. We denote the target as $\mathbf{t}$. At each time step n, the target is a column vector in the form of $\mathbf{t}(n) = [t_s(n)\ t_a(n)\ t_f(n)]'$, where $t_s(n)$ is the shift target, $t_a(n)$ is the amplitude target, and $t_f(n)$ is the frequency target. In our demo, a 15000 time step ramp signal with starting value of 0.475 and ending value of 0.525 is used as the shift target sequence, which represents a maximal 5% relative deviation from the shift of the training signal. A 15000 time step slow sine oscillation with a period of 5000 time steps, an amplitude of 0.06, and a shift of 0.2 is used as the amplitude target sequence, which represents a maximal 30% relative deviation from the amplitude of the training signal. Another 15000 time step ramp signal with starting value of 0.095 and ending value of 0.105 is used as the frequency target sequence, which

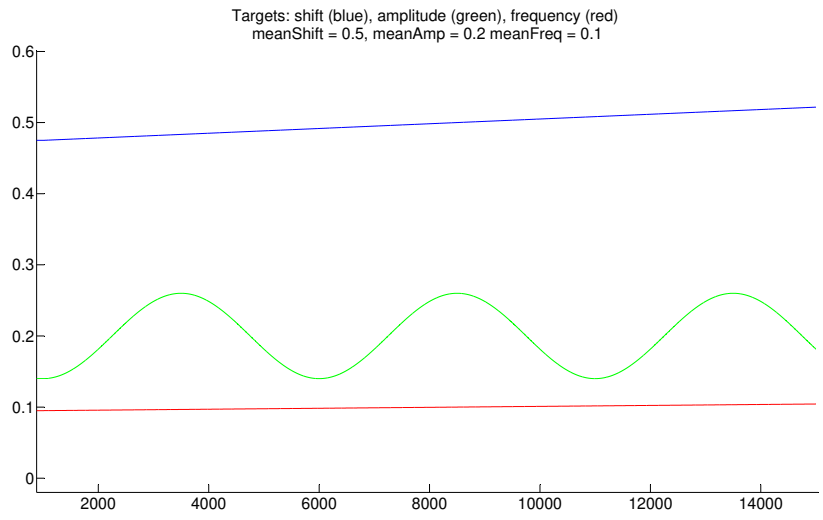


Figure 5: The control targets used in the demo, shift (blue), amplitude (green), frequency (red)

also represents a maximal 5% relative deviation from the frequency of the training signal. Figure 5 illustrates the trace of the control targets used in the demo.

With the control target defined, then the error signal $\mathbf{e}(n)$ at any time step n is computed as the differences between the targets and the observers. Formally, it is $\mathbf{e}(n) = \mathbf{t}(n) - \mathbf{o}(n) = [t_s(n) - o_s(n), t_a(n) - o_a(n), t_f(n) - o_f(n)]'$.

Now we design a controller whose purpose is to bring the error signal $\mathbf{e}(n) = [e_s(n)\ e_a(n)\ e_f(n)]'$ to zero. Specifically, we want to find three control vectors $\mathbf{c}_s$, $\mathbf{c}_a$, $\mathbf{c}_f$ of size N such that the controlled reservoir dynamics

$$\mathbf{x}(n+1) = \tanh\left(\mathbf{W}\mathbf{x}(n) + \mathbf{W}^{fb}y(n) + \gamma_s e_s(n)\mathbf{c}_s + \gamma_a e_a(n)\mathbf{c}_a + \gamma_f e_f(n)\mathbf{c}_f\right), \quad (4)$$

exhibits zero (or more realistically, low-amplitude) $\mathbf{e}(n)$. Here γ_s, γ_a and γ_f are control gains for shift, amplitude, and frequency.

We compute the control vectors $\mathbf{c}_s$, $\mathbf{c}_a$ and $\mathbf{c}_f$ from the results of a series of perturbations of the reservoir dynamcis. Specifically, we carry out the following steps.

1. We freely run the reservoir 1000 time steps to home in on a stationary output sine oscillation. This yields the initial conditions for the perturbation experiments. Specifically, the reservoir state, the observer values, and the observer internal state values at the last time step in this free run will be used as the starting states of all following perturbation experiments. We

denote the reservoir initial state as $\mathbf{x}(start)$, the observer initial values as $\mathbf{o}(start)$, and the observer internal state initial values as $\mathbf{s}(start)$.

2. Starting from $\mathbf{x}(start), \mathbf{o}(start), \mathbf{s}(start)$ we let the system run freely for further 250 timesteps. From the last 50 timesteps we average the observations $\mathbf{o}(n)$, obtaining unperturbed baseline values $[\bar{o}_s, \bar{o}_a, \bar{o}_f]$ for them.

3. We now repeat this 250-timestep run, started again from $\mathbf{x}(start), \mathbf{o}(start), \mathbf{s}(start)$, as many times as there are neurons in the reservoir, i.e. N times. At each instance, we perturb one unit x_i in turn by adding a small constant bias $\delta = 0.01$ to it throughout the 250 timesteps. Concretely, for $i = 1, \ldots, N$ we run the reservoir from the starting condition with the perturbed update rule

$$\mathbf{x}(n+1) = \tanh\left(\mathbf{W}\mathbf{x}(n) + \mathbf{W}^{fb}y(n) + \delta\mathbf{e}_i\right), \tag{5}$$

where $\mathbf{e}_i$ is the ith unit vector of length N. Again, from each of these runs we glean the average reading of the three slow observers $\mathbf{o}(n)$ over the last 50 steps; let us call these $[\bar{o}^i_s, \bar{o}^i_a, \bar{o}^i_f]$. Figure 6 shows a plot of the observer traces of a series of perturbation experiments.

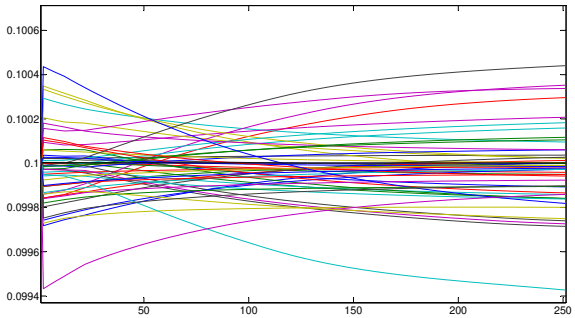


Figure 6: Observer traces under perturbations. The thick black line in the center of the bundle is the reading from the unperturbed baseline run.

4. We take the gradients of the slow observables w.r.t. the individual perturbations,

$$[c^i_s\ c^i_a\ c^i_f] = \frac{[\bar{o}^i_s, \bar{o}^i_a, \bar{o}^i_f] - [\bar{o}_s, \bar{o}_a, \bar{o}_f]}{\delta_i}, \tag{6}$$

and stack them in three N-dimensional column *control vectors*

$$[\mathbf{c}_s\ \mathbf{c}_a\ \mathbf{c}_f] = [c^i_s\ c^i_a\ c^i_f]_{i=1,\ldots,N}. \tag{7}$$

5. In order to minimize interactions between the controls for shift, amplitude and frequency, we finally orthogonalize $[\mathbf{c}_s \ \mathbf{c}_a \ \mathbf{c}_f]$ by the Gram-Schmidt procedure, applied in the order $\mathbf{c}_s \rightarrow \mathbf{c}_a \rightarrow \mathbf{c}_f$.

Figure 7 shows the three control vectors thus obtained. Note that with the construction methods mentioned above, these vectors are pairwise orthogonal.

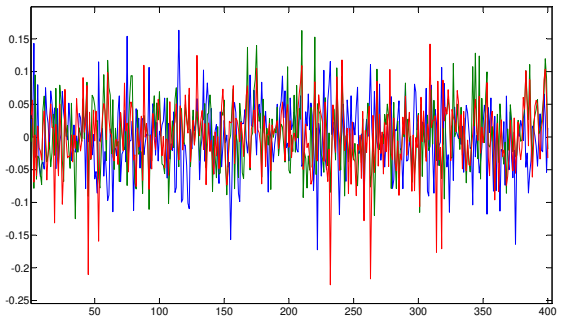


Figure 7: The three control vectors $\mathbf{c}_s$ (blue), $\mathbf{c}_a$ (green) $\mathbf{c}_f$ (red) obtained in our demo example.

The rationale behind this method of computing control vectors is simple:

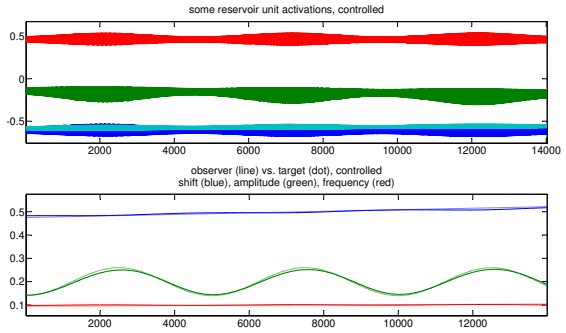
- The control objective is to suppress the amplitude of the (slow) error signal $\mathbf{e}(n) = \mathbf{t}(n) - \mathbf{o}(n)$.
- Each of $\mathbf{c}_s$, $\mathbf{c}_a$ and $\mathbf{c}_f$ reflects in its ith component the differential contribution of a slow perturbation of $x_i(n)$ to a change of $\mathbf{o}_s$, $\mathbf{o}_a$ and $\mathbf{o}_f$.
- With the control law (4), these contributions are directly cancelled.

Figure 8 shows the behavior of the reservoir, and the observer of $\mathbf{o}_s$, $\mathbf{o}_a$ and $\mathbf{o}_f$ when the control is enabled. By manual experimentation, control gains $\gamma_s = 5$ $\gamma_a = 10$ $\gamma_f = 20$ were found to work well enough for demonstration purposes.

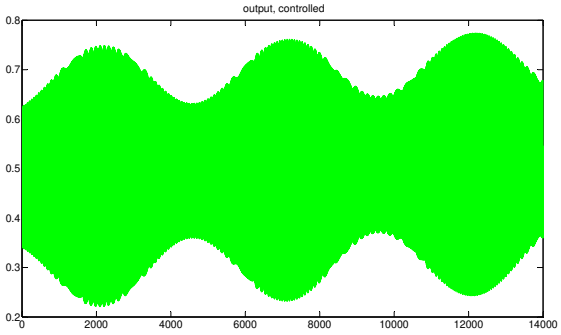
An inspection of Figure 8 clearly shows that the control functions to a large extent, although not to perfection.

4 Equilibration as a means to reduce control energies

The control loop has to act against the attractor "forces" of the reservoir, which "try" to stabilize shift, amplitude and frequency at their baseline values. In this situation we investigated how a method can help that we called *equilibration* in [7],



(a)



(b)

Figure 8: Behavior of controlled reservoir. (a) top: some reservoir state traces condition (same units are traced as in Figure 4b). (a) bottom: online measured observers (solid lines) vs. control targets (dotted). (b): oscillation output of controlled system.

and which has been independently discovered and explored for various purposes elsewhere [10, 11].

The idea behind equilibration is as simple as it is general. Consider a generic setting where a reservoir obeying the generic state update equation 1 is driven by external influences of some sort – for instance, by sensor input, or (as in our demo) by control input. Then,

1. collect states $\mathbf{x}(n)$ from an extended run of the driven system,

2. use these to recompute from scratch all the weights relevant for the reservoir dynamics (including, if applicable, input weights and output and output feedback weights), such that the collected states are reconstructed through these new weights in the least mean square error sense,

3. replace the old weights with the new ones.

In an ideal case, with the new weights in place, the original dynamics captured in the training states $\mathbf{x}(n)$ will be recovered by the new weights. The new weights will however typically be of smaller magnitude than the old ones (if the least mean square error minimization was properly regularized), which is generally beneficial for robustness of dynamics in the presence of noise and for generalization in the sense of machine learning. However, the most intriguing effect of the new weights is that the new system has "internalized" the driving "forces" which gave rise to the original collection $\mathbf{x}(n)$ of states.

For an intuitive understanding, and for preparing the application to our demonstration case, consider a simplified hypothetical case where the original reservoir acts as a stable oscillator with one stable amplitude. Assume further that the external driver slowly modulated this amplitude, e.g. by externally changing some control parameter within the reservoir. Then, in the equilibrated system, each of the amplitudes that were maintained in the externally driven phase for some time would become indifferently stable, whereas in the original system only the single baseline amplitude would be stable. If the equilibrated system would be subjected to some small state noise during a run, we would observe that the oscillation's amplitude would slowly change on a random walk.

The mathematical core (not the learning procedure) of the difference between the unequilibrated original system and the equilibrated counterpart can be highlighted by a simple formal example. Consider, as a baseline system, the 3-dimensional system

$$\begin{aligned} \dot{r} &= \tau\,(-r + \exp x), \\ \dot{\theta} &= 1, \\ \dot{x} &= -c\,x. \end{aligned}$$

This system describes a stable circular oscillation of radius 1 in polar coordinates θ, r at an x-location of 0. Figure 9 shows a phase portrait. If in the evolution of this system the value of x would be clamped to other than its naturally stable value of 0, the resulting amplitude would be $\exp x$. Now assume that by external manipulation x is indeed slowly varied, forcing the system through extended periods of time wherein the amplitude is thus changed together with x, and assume further that a new set of equations is sought which can describe this behavior. One would arrive at a new set of equations,

$$\begin{aligned} \dot{r} &= \tau\,(-r + \exp x), \\ \dot{\theta} &= 1, \\ \dot{x} &= 0, \end{aligned}$$

which has "internalized" the forced dynamics in which x essentially stands still at different values and the amplitude accordingly assumes different values. In the

new – equilibrated – system, each amplitude (togehter with its x-value) is now an indifferently stable behavior mode of the system. Furthermore, small noise added to the system evolution will send the amplitude (and x) on a slow random walk.

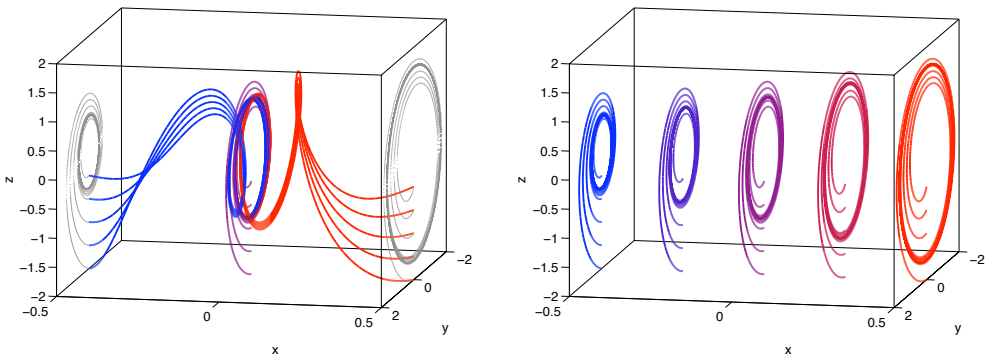


Figure 9: The mathematical core of equilibration. Left: original system, right: equilibrated system. For explanation see text. The polar coordinates θ, r of the equations in the text are plotted to the y, z plane.

Returning to our demo system, we equilibrated it as follows.

1. We repeated the controlled run shown in Figure 8, and collected the reservoir states $\mathbf{x}(n)$ and the output unit's states $y(n)$ from this run. Notice that these states come from periods during which, due to the slow targets and the control, shift and amplitude and frequency differed from the native stable values in various ways.

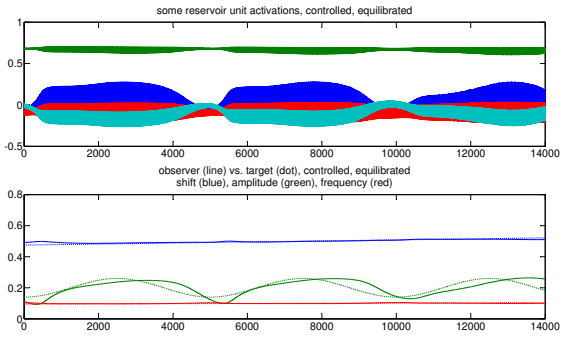
2. We then re-computed the reservoir weights $\mathbf{W}$ and feedback weights $\mathbf{W}^{fb}$, obtaining equilibrated versions $\mathbf{W}_{eq}$ and $\mathbf{W}^{fb}_{eq}$, by minimizing the square error

$$MSE(\tilde{\mathbf{W}}, \tilde{\mathbf{W}}^{fb}) = \sum_{n=1}^{15000-1} \left(\tanh^{-1}(\mathbf{x}(n+1)) - (\tilde{\mathbf{W}}\mathbf{x}(n) + \tilde{\mathbf{W}}^{fb} y(n)) \right)^2, \quad (8)$$

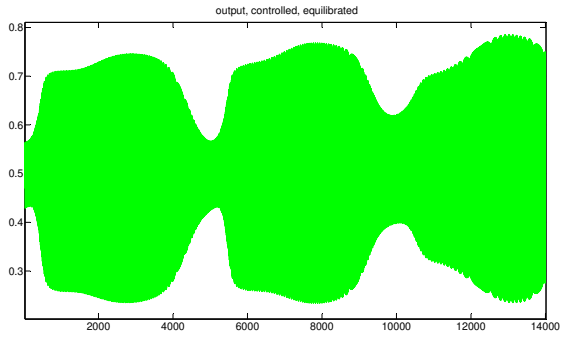
   using ridge regression with a regularization constant $\alpha = 0.05$. We did not recompute the output weights $\mathbf{W}^{out}$.

The bottomline purpose of equilibration in our task of controlling slow observables is to reduce the signal power of control inputs inserted into the reservoir during controlled runs. The intuition is that in a perfectly equilibrated system, control could be achieved with asymptotically zero energy of control signals if the targets change "adiabatically" slowly; and with very small energy of control signals if the equilibration is not perfect or the change rates of the targets are not exceedingly slow.

To demonstrate this point, the equilibrated system was again re-run in controlled mode, using the same targets as before, and the same control vectors $\mathbf{c}_s, \mathbf{c}_a, \mathbf{c}_f$. Because equilibration has bought us a highly increased sensitivity of the reservoir to control inputs, the control gains had to be reduced accordingly. By manual experimentation, a reduction by three orders of magnitude, from the original values of $\gamma_s = 5, \gamma_a = 10, \gamma_f = 20$ to $\gamma_s^{eq} = 0.005, \gamma_a^{eq} = 0.01, \gamma_f^{eq} = 0.02$ were found to work reasonably well. Figure 10(a) reveals the behavior of the equilibrated, controlled system.



(a)



(b)

Figure 10: Control of equilibrated reservoir. (a) Top: some reservoir state traces. Bottom: online measured observers (lines) against control targets (dotted). (b) Network oscillation output.

To quantify the reduction in control energy, we monitored the energies $(\gamma\ e(n))^2$ along the controlled runs in both the native and the equilibrated system. As expected, for the latter these energies are much smaller. Concretely, for the three control dimensions shift, amplitude and target we observed reduction factors for the mean control energies of approximately 4.6e+05, 1.1e+05, and 6.3e+05, respectively, i.e., reductions by five orders of magnitude.

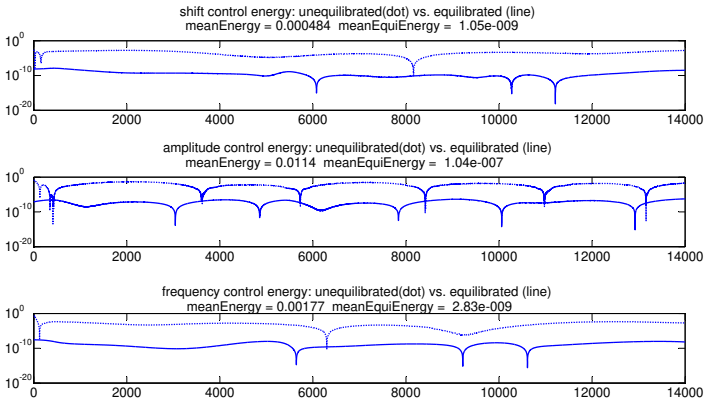


Figure 11: Control energies in native (dotted) vs. equilibrated system (lines). For explanation see text.

5 Discussion

The two core ideas of the approach outlined in this report can be summarized as follows:

1. In order to render a given oscillator network controllable w.r.t. some slow observable, we estimate a bias vector which incorporates the gradient of this observable w.r.t. slow forced shifts of the network unit's operating points. When scaled with a suitable gain and multiplied with the current tracking error, adding this bias to the ongoing network dynamics results in a (P-type) feedback control loop.

2. In order to render the network more sensitive to control, it can be equilibrated by recomputing its internal weights such that, in the ideal case, all behaviors occuring within the controllable range become indifferently stable. This leads to a very large reduction in the required control signal energy.

These two ideas are to a large extent independent of each other: the method of using gradient-defined bias vectors for control does not need equilibration to function; and reducing control energies by equilibration would function with other control mechanisms as well.

We demonstrated the conceptual viability of these two ideas in a simple demonstration of simultaneously modulating shift, amplitude, and frequency of a neural sinewave generator.

A number of refinements, extensions and alternatives suggest themselves and will be investigated as next steps.

- Instead of a simple P-controller, one may use PI or PID type controllers. We expect that this will improve the tracking accuracy, especially in the equilibrated system where our currently implemented pure P-controller leads to rather imprecise tracking (evident in Figure 10).

- The perturbation method for obtaining control vectors as gradient vectors is computationally expensive and biologically hardly plausible. If a "behavioral teacher" is available which provides paired modulated output and slow observables, a much faster and biologically feasible learning method can be used. This alternative method, described in [7] in a different context, would identify the control vector $\mathbf{c}$ with the regression gradient β in the estimated regression equation $\mathbf{o}(n) = \alpha + \beta < \mathbf{x}(n) >$, where $< \mathbf{x}(n) >$ is a smoothed version of the reservoir states.

- Among the three observables considered in this report, we found frequency especially hard to control. Elsewhere, however, we have demonstrated that slow bias input to an oscillator ESN can change its frequency over several (base 2) orders of magnitude [6]. In that study, however, it was the weights from the reservoir to the oscillator output neurons which were trained, not the bias vector, which was randomly given and remained fixed. This suggests that if we admit to adapt further parameters beyond the control vectors, we may very much increase the attainable controllability ranges.

- Investigate how the simultaneous controls of targets $\mathbf{t}_s(n)$, $\mathbf{t}_a(n)$ and $\mathbf{t}_f(n)$ interact with each other. We found that if only one of these targets is controlled for, the control is more efficient than when other targets are controlled simultaneously (not reported).

- In order further demonstrate the universality of the ESN based pattern generation method, we plan to train the core pattern-generating ESN to produce patterns more complex than sines (e.g. those used in robotic locomotion control tasks), and use our feedback control method to modulate these patterns.

- Animals can quickly modulate their locomotion patterns, e.g. when hitting an obstacle, with a speed of adaptation which is only possible with purely reflex-based, distal mechanisms. To come closer to such functionalities of biological pattern generators, we want to investigate how to directly couple sensor inputs into our controlled ESN pattern generator.

Finally, we want to point out a rather more visionary line of research based on the idea of equilibration, which we will pursue in the context of the EU FP7 project "AMARSi". The idea is to apply the principle of equilibration not only to a single neural circuit, but to an entire robotic system, including its body hardware. Abstractly, such a comprehensive system is just another high-dimensional dynamical system. While an isolated ESN has essentially only weights and biases

to offer as adjustable parameters, an integrated robot system has many more, including parameters defining its physical layout (e.g. spring constants, limb masses and sizes, etc.). In a computer simulation model, such parameters are as easily adaptable as are neural weights. From an equilibration perspective, recomputing such an extended set of parameters to make numerous behavioral modes of the (simulated) robot indifferently stable might lead to a combined physical/neural re-design where control energies for given control objectives are much reduced. Metaphorically speaking, equilibrated modes could in principle be sustained with zero control energy, like in passive walking demonstrators. We believe that this design idea holds some promise for passively compliant robots.

Acknowledgements. The research described in this report was supported by a grant from the European Commission (Grant Nr. 248311 AMARSi: Adaptive Modular Architecture for Rich Motor Skills, `http://www.amarsi-project.eu/`).

References

[1] A. Ayali. The insect frontal ganglion and stomatogastric pattern generator networks. *Neurosignals*, 13:20–36, 2004. http://stg.rutgers.edu/stgrefs/stg_library/Ayali_2004.pdf.

[2] S. L. Hooper. Central pattern generators. *Embryonic ELS*, 1999. http://crablab.zool.ohiou.edu/hooper/cpg.pdf.

[3] A. J. Ijspeert. Central pattern generators for locomotion control in animals and robots: a review. *Neural Networks*, 21/4:642–653, 2008. http://birg2.epfl.ch/publications/fulltext/ajIjspeert08a.pdf.

[4] A. J. Ijspeert, J. Nakanishi, and S. Schaal. Learning rhythmic movements by demonstration using nonlinear oscillators. *IROS*, pages 958–963, 2002. http://birg2.epfl.ch/publications/fulltext/ijspeert02b.pdf.

[5] H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks. GMD Report 148, GMD - German National Research Institute for Computer Science, 2001. http://www.faculty.jacobs-university.de/hjaeger/pubs/EchoStatesTechRep.pdf.

[6] H. Jaeger. Echo state network. In *Scholarpedia*, volume 2, page 2330. 2007.

[7] H. Jaeger. Reservoir self-control for achieving invariance against slow input distortions. technical report 23, Jacobs University Bremen, 2010.

[8] H. Jaeger and H. Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304:78–80, 2004. http://www.faculty.jacobs-university.de/hjaeger/pubs/ESNScience04.pdf.

[9] M. N. Levy and P. J. Martin. Neural regulation of the heart beat. *Ann. Rev. Physiol.*, 43:443–453, 1981. http://www.annualreviews.org/doi/pdf/10.1146/annurev.ph.43.030181.002303.

[10] N. M. Mayer and M. Browne. Echo state networks and self-prediction. In *Biologically Inspired Approaches to Advanced Information Technology*, volume 3141 of *LNCS*, pages 40–48. Springer Verlag Berlin / Heidelberg, 2004.

[11] R. F. Reinhart and J. J. Steil. A constrained regularization approach for input-driven recurrent neural networks. *Differential Equations and Dynamical Systems*, 2010 (online pre-publication). DOI 10.1007/s12591-010-0067-x.