JACOBS
UNIVERSITY

Herbert Jaeger

Reservoir Self-Control for Achieving Invariance Against Slow Input Distortions

Technical Report No. 23
October 2010

School of Engineering and Science

Reservoir Self-Control for Achieving Invariance Against Slow Input Distortions

Herbert Jaeger

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

E-Mail: h.jaeger@jacobs-university.de
http://www.faculty.jacobs-university.de/hjaeger/

Abstract

A method is presented for immunizing an Echo State Network against slow, task-irrelevant variation in its input. The main idea is to extract some principal components of the smoothed reservoir dynamics, and augment the reservoir with a feedback controller which attempts to pull these components to zero. This leads to a "homeostatic" self-stabilization of the reservoir dynamics. A proof-of-principle case study is presented where the network has to predict a two-mode input signal which is subjected to a large-amplitude, slow variation in offset. With the controller in place, this task is solved with exactly the same quality as in a condition where the input is undistorted. Furthermore, the action of the controller can be "compiled into" the reservoir, by recomputing reservoir weights such that the controlled network dynamics is recovered without the controller switched on. One thus finally obtains a reservoir whose internal dynamics are largely invariant against slow distortion in the input.

1 Introduction

Many signal processing, control, and temporal pattern recognition tasks are confronted with input data showing distorting variation on long timescales, where the variation is a disturbance for the processing task at hand. Here are some illustrative examples:

- A speech-to-text recognition system has to cope with speaker variation which becomes manifest in different pitch, speed or accent.
- A handwriting recognizer has to be robust against different slants, character widths, or sizes.
- A humanoid robot walking controller has to adapt to different ground slopes.

The unifying characteristic of these examples is that the distortion changes on a slower timescale than the natural timescale(s) of the processing task. For instance, in speech recognition the natural processing timescales are those of phonemes, words and phrases, while different speakers will take turns using the system at a lower rate; likewise, ground slope typically changes on longer scales than steplength.

In this report I present an unsupervised learning method whereby a generically trainable recurrent neural network (RNN) can learn to immunize itself, by an active self-regulation mechanism, against slow variations in the input characteristics. More concretely, I propose an architecture with the following components and mechanisms:

- The core component is a discrete-time RNN of the "Echo State Network" (ESN) type [1, 2], destined to be trained in a supervised fashion on some "payload" task (prediction, classification, control or other).
- Following the rationale of reservoir computing, of which ESNs are an instantiation, the input signal drives the RNN (called "reservoir" in this context). The reservoir is not modified by training. Only passive readout mechanisms are trained which combine the payload output signal from the input-excited signals within the reservoir.
- Training data is provided which exhibits distracting slow variation.
- This slow variation is reflected in equally slow, systematic variation of the excited response signals within the reservoir. A bank of randomly created, slow-timescale observers monitors the reservoir signals, yielding a collection of slow observables o_i of the reservoir dynamics. The first few principal components of these o_i are computed, yielding a small selection p_j of likewise slow, derived observables of the reservoir dynamics. Being the leading principal components, they account for most of the slow variation in the reservoir.
- A second module added to this picture is a feedback controller C which can modify the ongoing, input-driven dynamics of the reservoir by inserting a control input $\mathbf{c}(n)$ into the reservoir. The controller is designed (and trained) such that it can act as a tracking controller for the slow observables p_j.

- After this controller has been installed and trained, the payload task is trained in a supervised way by computing readouts from the reservoir, in the usual fashion of reservoir computing. Importantly, while this payload training takes place, the controller is active. It receives *zero* targets for the slow observables p_j and thus attempts to cancel the reflections of slow variation in the input from the reservoir dynamics.
- If this would work to perfection, the reservoir dynamics with the controller switched on should be identical to reservoir dynamics obtained from driving input which has *no distracting, slow variation*, greatly facilitating the payload task learning.
- In exploitation, the controller is likewise switched on. Again, if everything works to perfection, this means that the reservoir response to the driving input is invariant to the slow variations in the input, yielding a payload output of a quality equivalent to situations when the input has no slow variations.

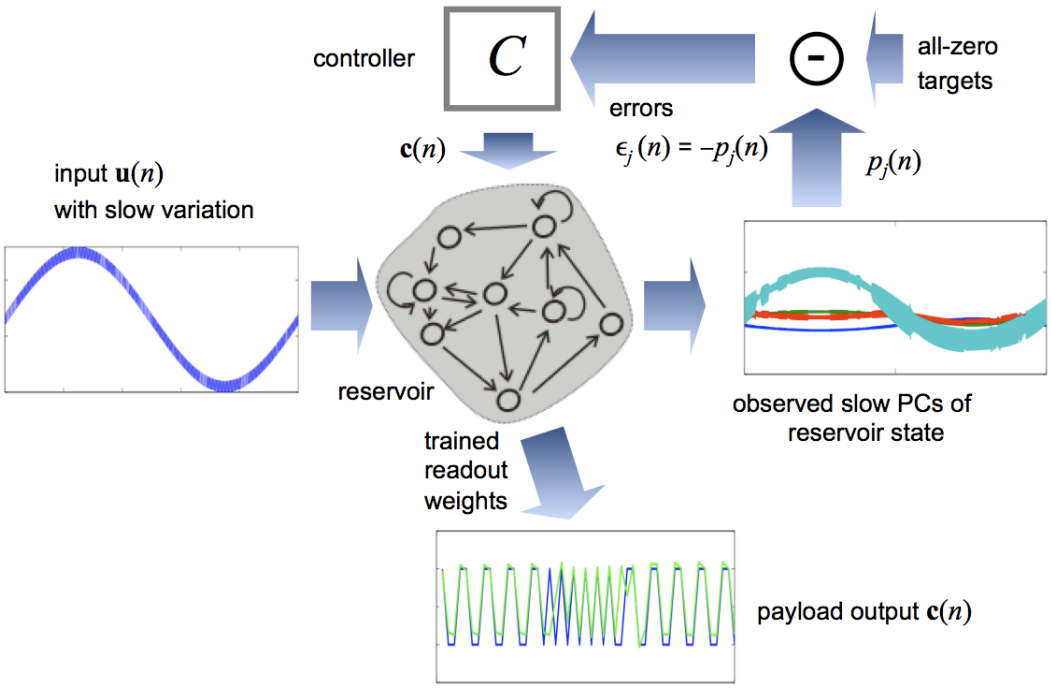


Figure 1: Overview of architecture. For explanation see text.

2 Model and demo example

The proposed strategy for reservoir self-control against slow variances in the input proceeds in two stages. In the first stage, a controller C is trained, in an unsu-

pervised way, to suppress some leading PCs of slow components in the reservoir dynamics. In the second stage, with the controller in place and activated, the system is trained on some payload task. Throughout this techreport, we will consider a single demo example with scalar input and scalar output. Before we embark on the subject of self-control, we inspect the behavior of the native system without self-control.

2.1 The native system and its payload task performance

I consider a standard ESN with N reservoir units receiving scalar input $u(n)$ and generating scalar output $y(n)$:

$$\mathbf{x}(n+1) = \tanh\left(\mathbf{W}\mathbf{x}(n) + \mathbf{W}^{in}u(n)\right), \quad (1)$$

$$y(n) = \mathbf{W}^{out}\mathbf{x}(n), \quad (2)$$

where $\mathbf{W}$ is the reservoir weight matrix (size $N \times N$), $\mathbf{W}^{in}$ is the input weight matrix (size $1 \times N$), and $\mathbf{W}^{out}$ is the output weight matrix (size $N \times 1$). Concretely, for our demo I chose $N = 50$, constructed $\mathbf{W}$ to have a connectivity of about 30%, scaled it to a spectral radius of 0.5, and sampled the input weights from a uniform distribution in $[-0.5, 0.5]$. These are ad-hoc settings. Throughout this report I assume that the reader is familiar with the basic concepts of reservoir computing, and I will not detail out standard procedures from that field.

In our demo example, the input signal is made from two randomly alternating generators, the first being a binary one-step oscillation and the second a binary two-step oscillation. A switch from one mode to the other was done with a probability of 0.02 per time step. Optionally, the input can be modulated by a shift which slowly oscillates with a period of 1000 time steps and an amplitude of 12. I will use notation $u(n)$ for the unmodulated and $\tilde{u}(n)$ for the modulated input. Figure 2 illustrates the input signal.

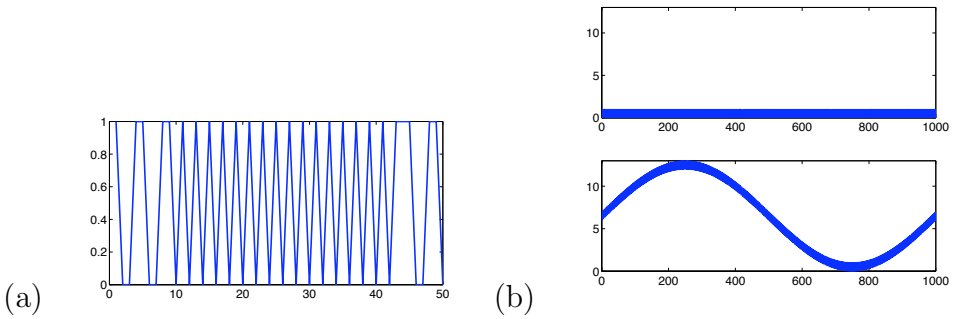
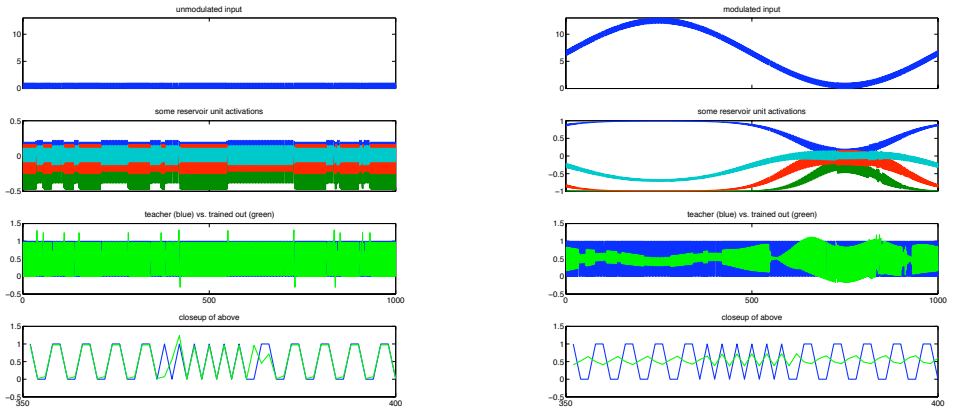


Figure 2: Input signal used in demo example. (a): closeup on unmodulated input. (b) 1000 step sequence of unmodulated (top) and shift-modulated (bottom) input.

The payload task which I consider is simple: predict the input by one time step. That is, the teacher signal is $d(n) = u(n-1)$. For obtaining a baseline performance,

the reservoir was driven with an unmodulated input for 6000 time steps, and output weights were computed from the harvested reservoir states (dismissing the first 1000 ones to account for initial state washout) by ridge regression with a regularization constant $\alpha = 0.2$. This resulted in a normalized mean square test error (NRMSE) of 0.33, with a mean absolute output weight size of 1.8. When the training was repeated using the shift-modulated input, an NRMSE of 0.75 with mean absolute output weights of 2.8 were obtained. Note that the task still was to predict the *unmodulated* input. Figure 3 illustrates the outputs generated from the trained network.



(a) (b)

Figure 3: Baseline performance of un-controlled, native reservoir system. (a) using unmodulated input, (b) using modulated input. Top: input. Second panels from above: four traces of reservoir unit activations. Third panel: teacher vs. trained network output. Last panel: closeup of third panel.

It becomes clear from the NRMSEs and an inspection of Figure 3 that the payload task training is severely impaired by the shift modulation of the input. Two apparent mechanisms for this impairment are that, first, the input modulation introduces a task-irrelevant source of variation into the reservoir dynamics which the linear regression of the readout training has to compensate; and that, second, the high input amplitudes incurred by the shift modulation drives many reservoir units close to saturation for some periods of time and reducues the task-relevant variation of reservoir unit activations in these intervals.

2.2 Training and testing the controller

We first add smoothed state observers $\mathbf{o}(n)$ to the picture:

$$\mathbf{o}(n+1) = 0.98 \cdot \mathbf{o}(n) + 0.02 \cdot \mathbf{x}(n+1). \tag{3}$$

The smoothing constant 0.02 is chosen such that the resulting slow signal smoothes out most of the fast, input-related oscillations of the state signal, but

is fast enough to track the slow sinewave of the input shift modulation. After running the reservoir with modulated input, normalized versions P of the leading four PCs of the obtained smooted state signal $\mathbf{o}(n)$ were computed as follows.

1. Center the $\mathbf{o}(n)$ to zero mean in each component, by subtracting their temporal means μ, giving $\bar{\mathbf{o}}(n) = \mathbf{o}(n) - \mu$. Let $\bar{\mathbf{O}} = [\bar{\mathbf{o}}(1) \cdots \bar{\mathbf{o}}(M)]$ (after discarding initial washout data).

2. Compute the singular value decomposition (SVD) of the correlation matrix $C = \bar{\mathbf{O}}\,\bar{\mathbf{O}}'$, where $'$ denotes transpose, getting $[\mathbf{U}\ \mathbf{S}\ \mathbf{V}] = svd(C)$.

3. Let $\mathbf{U}(:, 1:4)$ denote the first four columns of $\mathbf{U}$ (we will be using this Matlab notation from now on without further explanation). Obtain the first four PCs of the centered slow observables $\bar{\mathbf{O}}$ by $P_0 = \mathbf{U}(:, 1:4)'\ \bar{\mathbf{O}}$. P_0 contains the first four PCs in its four rows.

4. Normalize the rows of P_0 to unit variance, obtaining $P = diag(\sigma^{-2})\, P_0$, where σ is the vector of standard deviations of the rows of P_0. Since the source signals $\bar{\mathbf{o}}(n)$ had zero mean, the PCs in P have zero mean too.

We opted for using the first four PCs on the grounds of manual experimentation. The optimal number of leading PCs to be selected will depend on the case at hand. Figure 4 gives an impression of the signals involved in this process.

Notice that the PC signals in P can be obtained by an affine transformation $P(n) = diag(\sigma^{-2})\, \mathbf{U}(:, 1:4)\ (\mathbf{o}(n) - \mu)$, enabling a simple online generation of them.

Our next goal is to create a simple linear proportional feedback controller whose purpose is to bring the $P(n) = (p_1 \cdots p_4)'$ to zero. I want to find four control vectors $\mathbf{c}_1, \ldots, \mathbf{c}_4$ of size N such that the controlled reservoir dynamics

$$\mathbf{x}(n+1) = \tanh\left(\mathbf{W}\mathbf{x}(n) + \mathbf{W}^{in}u(n) - \sum_{i=1}^{4} \gamma_i p_i(n)\mathbf{c}_i\right), \tag{4}$$

exhibits zero (or more realistically, low-amplitude) $P(n)$. Here γ_i are control gains, and $-p_i(n)$ is the error signal (with respect to a zero target).

I set the control vector $\mathbf{c}_i$ to the vector of correlations of $p_i(n)$ with the centered raw state signals $\mathbf{x}(n)$. In detail, using notation $E[\cdot]$ for expectation, let $\bar{\mathbf{x}}(n) = \mathbf{x}(n) - E[\mathbf{x}(n)]$. Then, put

$$\mathbf{c}_i = E[p_i(n)\, \bar{\mathbf{x}}(n)]. \tag{5}$$

A numerical estimate of $\mathbf{c}_i$ can be obtained from the harvested reservoir state and P data of a trial run of the input-driven system by any standard offline or online estimation method for correlations. In our demo, I ran the system for 5000

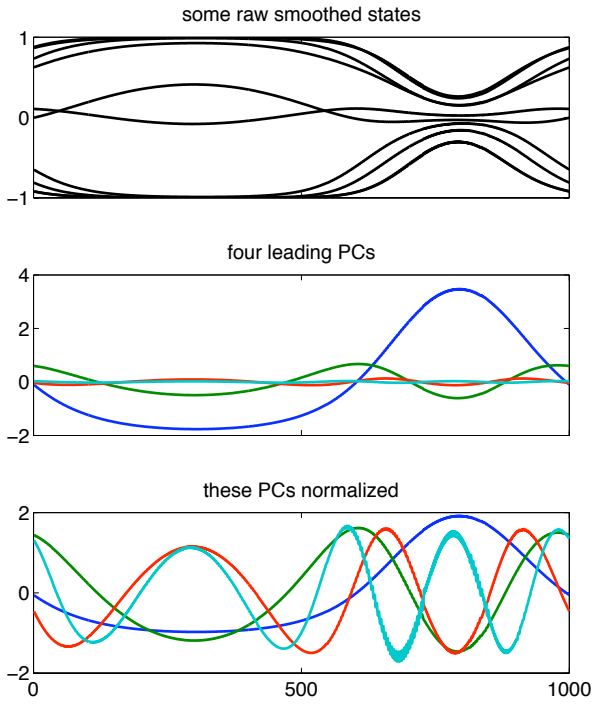


Figure 4: Illustrating the principal components of the smoothed reservoir unit activation signals. Top: a choice of some components of $\mathbf{o}(n)$. Center: the four leading PCs of all 50 such signals. Their energy quickly falls off: the first four singular values in $\mathbf{S}$ are 1.3, 0.068, 0.0026, 0.0002 ($\times$ 1.0e+04). Bottom: same, normalized to unit variance.

time steps, discarded the first 1000 steps for washout, and estimated $\mathbf{c}_i$ from the remaining 4000 sample points by

$$\langle p_i(n)\,(\mathbf{x}(n) - \langle \mathbf{x}(n) \rangle) \rangle,$$

where $\langle \cdot \rangle$ denotes the mean operator.

Figure 5 shows the four control vectors thus obtained. Note that by construction, these vectors are pairwise orthogonal. From Figure 5 it is also apparent that as the index i grows, the norm of the control vector $\mathbf{c}_i$ shrinks. This reflects the fact that the un-normed PCs P_0 capture the principal directions of variation in the (smoothed) reservoir state signals; thus the sqared norms of the control vectors should be roughly proportional to the corresponding singular values in $\mathbf{S}$ (only roughly because we compute the control vectors by correlation with the *un*smoothed states).

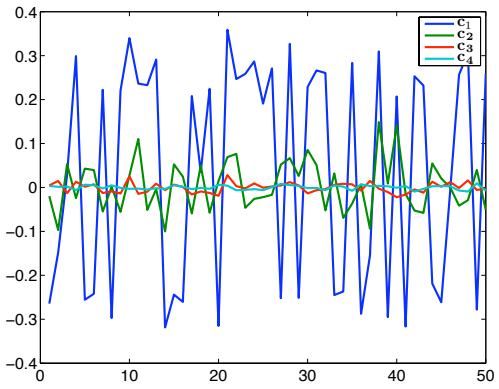


Figure 5: The four control vectors $\mathbf{c}_i$ of our demo example.

The rationale behind this construction of control vectors is simple:

- The control objective is to suppress the amplitude of the $p_i(n)$.
- The $p_i(n)$ are smoothed variations of reservoir states in certain principal directions of state space.
- Each $\mathbf{c}_i$ reflects in its jth component the contribution of $x_j(n)$ to this smoothed variation.
- With the control law (4), these contributions are directly cancelled.

Figure 6 shows the behavior of the reservoir, and the measured amplitudes of the four p_i, when the control is enabled. The driving input was shift-modulated as in Figure 1b (bottom). By coarse manual experimentation, control gains $\gamma_1 = \gamma_2 = \gamma_3 = 80$, $\gamma_4 = 160$ were found to work well enough for demonstration purposes.

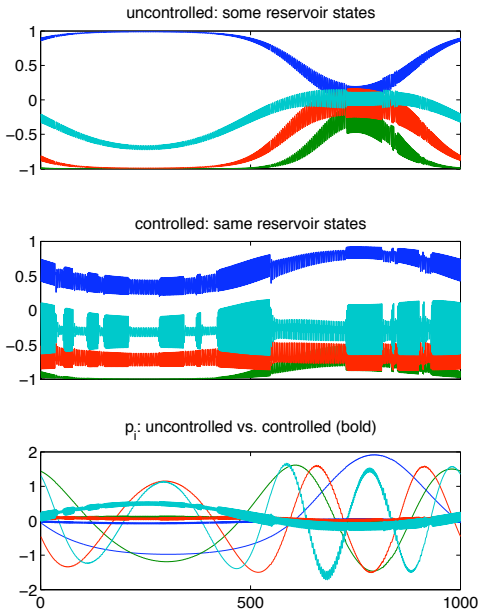


Figure 6: Comparing the controlled with the uncontrolled reservoir. Top: some reservoir state traces in uncontrolled condition (repeated from Figure 3b). Center: same states traced in controlled condition. Bottom: online measured p_i in controlled and uncontrolled conditions.

An inspection of Figure 6 shows that the control functions to a large extent, although not to perfection. In order to quantify the control efficiency, I calculated the mean energy ME (i.e. squared amplitudes) of the four p_i in the controlled and uncontrolled conditions. The damping ratios $ME_{uncontrolled}/ME_{controlled}$ were found to be 489, 174, 205, and 11.

Finally, I used the reservoir states from the controlled run to compute new output weights for the payload task, using the same regularization constant as used for the native system (compare Section 2.1). Figure 7 is the analog of the two lowest panels in Figure 3.

An NRMSE of 0.33 was obtained, the same as for the uncontrolled reservoir *driven by unmodulated input*. The payload performance is virtually identical to that undistorted case.

3 Internalizing the control

The control loop adds significant reservoir-external computational machinery to the core ESN. I will now show how its functionality can be "compiled into" the reservoir dynamics proper, by recomputing the reservoir weights $\mathbf{W}$ and input weights $\mathbf{W}^{in}$, obtaining new such weights $\mathbf{W}_{eq}, \mathbf{W}^{in}_{eq}$. When the network is rerun with these new weights, and the control loop removed, it will perform almost identically as with the old weights and the control being activated. I will call the

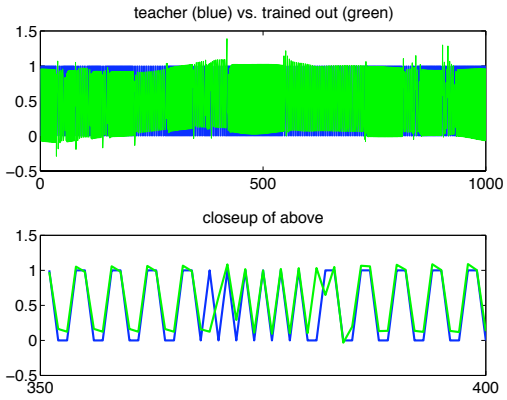


Figure 7: The payload task performance of the controlled reservoir. Analog of two bottom panels in Figure 3.

process of obtaining $\mathbf{W}_{eq}, \mathbf{W}^{in}_{eq}$ from $\mathbf{W}, \mathbf{W}^{in}$ the *equilibration* of the reservoir.

The idea behind equilibration is simple. Let $\mathbf{x}(n)$ be the reservoir state signal obtained from a run where the input was distorted and the control switched on, i.e. using (4). Then, recompute the input and internal weights such that they optimally reproduce $\mathbf{x}(n+1)$ from $u(n+1)$ and $\mathbf{x}(n)$ in a least mean square sense:

$$\mathbf{W}_{eq}, \mathbf{W}^{in}_{eq} = \\ \arg\min_{\tilde{\mathbf{W}}, \tilde{\mathbf{W}}^{in}} \; E\left[\left(\tanh^{-1}(\mathbf{x}(n+1)) - (\tilde{\mathbf{W}}\mathbf{x}(n) + \tilde{\mathbf{W}}^{in}u(n+1)\right)^2\right]. \quad (6)$$

I computed a solution to this LMS problem with ridge regression (using a regularization constant $\alpha = 0.2$), based on state and input data from a 5000 step run with distorted input and the controller activated. The new weights were inserted into the reservoir system, and it was re-run with distorted input, without the control loop – that is, the update equation

$$\mathbf{x}(n+1) = \tanh\left(\mathbf{W}_{eq}\mathbf{x}(n) + \mathbf{W}^{in}_{eq}u(n)\right) \quad (7)$$

was used.

Figure 8 shows that the resulting (uncontrolled!) reservoir dynamics is virtually identical to the dynamics obtained in the controlled condition (using distorted input in both simulations).

Not surprisingly, when the payload task output weights were recalculated from the equilibrated network dynamics, the task performance yielded an all but identical NRMSE – to be precise, it was now 0.331 as compared to 0.332 for the controlled network, and 0.333 for the native network on undistorted input.

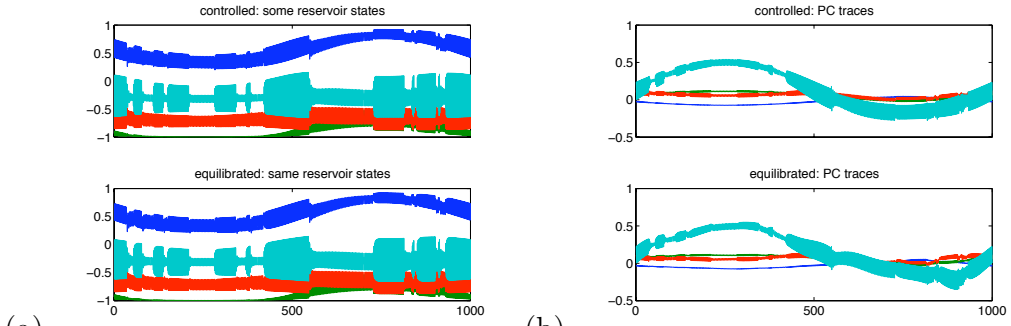


Figure 8: Comparing reservoir dynamics in the controlled vs. equilibrated conditions; (a): some internal reservoir state traces, (b): the four slow measured p_i. Left upper panel is repeated from Figure 6.

4 Discussion

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

- It often occurs in signal processing applications that the input is subject to variation on a timescale that is slower than the natural timescale(s) of the "payload" processing task.
- When the processing is done by an Echo State Network, the slow variation in the input will induce slow variations in the reservoir dynamics.
- A natural way to monitor this slow reservoir variation is by measuring some leading principal components p_i of the smoothed reservoir state signals.
- If one could somehow suppress these PCs, one may hope that the reservoir dynamics becomes invariant (to a certain degree) against the slow variation in the driving input, with obvious benefits for training the payload output in the usual reservoir computing fashion.
- One straightforward approach to cancel these PCs is by adding compensating components $\mathbf{c}_i$ to the network states, where these compensating $\mathbf{c}_i$ are scaled with the current amplitude of the to-be-suppressed p_i. This leads to a proportional controller that self-controls the reservoir dynamics.
- A working solution to determine such compensating $\mathbf{c}_i$ is to set them to the correlation coefficients of the p_i with the reservoir states.
- The controlled network's dynamics now exhibits important invariances against input distortion. A payload task trained on the controlled network, using distorted input, is solved with equal quality as the same task trained on the native network, tested with undistorted input.

- By a final twist, the controller's action on the reservoir states can be "internalized", or "compiled into" the reservoir, leading to new network weight parameters. These engender a network dynamics which is virtually identical to the controlled network's, but without the controller acting.

The "training" of the controller boils down to estimating the principal components p_i of the smoothed reservoir states. This is an unsupervised learning task. I solved it here by the canonical offline estimation of PCs from smoothed state signals harvested in a trial run. It could likewise be effected by online adaptive methods for PC estimation.

I would like to point out the natural connection of this approach with the idea of homeostasis. In a nutshell, one could say (or hope), that *a neural signal processing module can immunize itself against slow variation in the input by entertaining a "homeostatic" control of its internal dynamics.*

An intriguing aspect of the approach is that it is unnecessary to analyse the nature of the input, in the sense of creating a model of the slow variations in it. Concretely, in the demo example in this report, there was no need for the self-controlling reservoir to detect that the variation in the input was a shift. While the first extracted PC (the blue lines in Figure 4) roughly follows the input shift, it is however a nonlinear transformation of the sinewave-shaped shift dynamcis, reflecting the nonlinear (saturation-related) impact of the input on the reservoir states. The other PCs reflect slow response components intrinsic to the reservoir rather than the nature of the external slow variation.

As a proof-of-principle demonstration, this approach was illustrated in this report with what is likely the simplest kind of slow variation in input signals, namely a slow drift in offset.

This report marks only a starting point for further investigations, which are being pursued within the EU FP7 projects ORGANIC and AMARSi in the machine learning group at Jacobs University. Questions which are under investigation comprise the following:

- Use more refined controllers than a simple proportional one, e.g. by adding integral terms to the error.

- Investigate slow reservoir observations that are not linear combinations of smoothed reservoir states. Particularly interesting candidates are smoothed energies of states and – in tasks that involve oscillatory data – smoothed frequency measurements.

- Carry out a formal analysis of the efficiency, stability, and timescale separation issues of this method.

- Investigate how the simultaneous controls for the p_i interact with each other: I found that if only one of these PCs is controlled for, the control is more efficient than when other PCs are controlled simultaneously (not reported).

- Extend the approach to systems that are not input-driven signal transducers, but autonomous pattern generators. Slow observables of interest would then be observables of the generated output, for instance, shift, amplitude or frequency of an oscillatory output. The presented control method then would be adapted to enable the pattern-generating reservoir to control these properties of its output pattern. This would be of potential interest for neural motor control.

- Generalize the control mechanism by admitting modulation of other reservoir quantities besides the unit activations. For instance, one could envision a controller that modulates the reservoir weights. Such different target quantities for control action require different training methods for the controller. In preliminary studies, I used a numerical estimate of the gradient of the slow observable to be controlled w.r.t. the controllable network parameters to obtain analogs of the control vectors $\mathbf{c}_i$. In a pattern generation setting, this method made it possible to control the frequency, amplitude and/or shift of a periodic pattern generated by the network.

References

[1] 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.

[2] 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.