JACOBS
UNIVERSITY

Jordan Ivanchev

Fast time scale modulation of pattern generators realized by Echo State Networks

Technical Report No. 28
March 2013

School of Engineering and Science

Fast time scale modulation of pattern generators realized by Echo State Networks

Jordan Ivanchev

School of Engineering and Science
Jacobs University Bremen gGmbH
Campus Ring 12
28759 Bremen
Germany

E-Mail: j.ivanchev@jacobs-university.de

Abstract

There are many attempts in the field of robotics and artificial intelligence towards achieving a well-functioning architecture that can control an agent in a stable and elegant manner, while allowing for mixing of behaviors. However, state of the arts robots are far away from the coordinated, complex and graceful behavior that we can observe in animals. A popular approach towards tackling complex motor control problems is the employment of pattern generators. Several architectures have shown good performance, modulating and switching between different patterns. However, those methods perform gradual changes rather than the fast and yet smooth modulations of a pattern for a short duration of time, characteristic for reflexive actions. In this report I present a practical study of a technique that achieves changes in amplitude, frequency and shift of a periodic signal in the time frame of less than a period.

Contents

1 Introduction **1**
1.1 Motivation and research landscape 1

2 Objectives **2**

3 Methods and Results **3**
3.1 Echo state networks 3
3.2 Training the pattern generator 4
3.3 Training of the controller: Addition of internal units 5
3.4 Qualitative observations on the modulation performance 7
3.5 Measuring the smoothness of the modulation transition 9

4 Discussion **11**
4.1 Insights of training a pattern generator 11
4.2 Challenges of fast time scale modulation and why this approach works 13
4.3 Evaluation of performance 15

5 Conclusion **16**

1 Introduction

1.1 Motivation and research landscape

Central pattern generator (CPGs) research has become a useful tool for realizing locomotion in robots. A survey of CPG inspired methods for control of legged robots [1] describes a number of attempts to tackle this complex task. In biological systems, pattern generators are mainly positioned in the spine and are responsible for motor control as discussed in [2]. More than that, according to [3], it is believed that they are controlled on a low level by neuromodulators that can change the dynamics of the generator in order to increase speed, change gait etc. As discussed in [2] those low level changes in the motor control are believed to come from the spine as well. It is interesting to see that the movement of one leg can be considered to be a result of the combination of various patterns controlling different muscles. In this case the architecture seems to be hierarchical, however for coordination of the movements of several legs the structure is not hierarchical any more according to [4]. When a change of pace or gate is needed, the hierarchical structure in the spine is almost solely responsible for that change, however when there is an obstacle another machinery is activated that should control the pattern generators together with their neuromodulators such that the obstacle is avoided while maintaining the desired behavior. [5], [6] and [7] all agree that this control system is not centralized and it seems that each leg has its own modulator structure.

Controlling or modulating a pattern generator in order to change the frequency, amplitude and offset of its output is an object of current research in robotics. One of the approaches is to implement and control the pattern generators using reservoir computing (RC). In this approach a generative reservoir is used to implement the pattern generator. An input to the reservoir may be used in order to control it as in [8]. Here the pattern generated is shown to be modulated using slowly varying input signal that changes the frequency. Also a "gait" change, meaning a qualitative change of the produced pattern, was demonstrated by using an input signal as a switch.

When a RC approach is used it is vital to note that the reservoir is high dimensional and has extremely complex nature which makes it challenging to analyse. When used for practical applications it is often treated as a black box. Another trait of this method is the fact that the training of the reservoir is computationally cheap compared to other pattern generating strategies. More than that, the reservoir approach is capable of generating very complex high dimensional patterns with ease.

In [9] a tracking controller that modulates an echo state network (ESN) pattern generator is described. The idea of this method is to design a feedback mechanism that interacts with the pattern generator allowing it to track a slowly varying control targets such as frequency, amplitude or shift, thus gradually modulating the pattern. However, a more practical, useful and biologically plausible type of control mechanism would only require a spike signal signifying that the pattern

should change in a specific way.

This report is structured as follows. The Objectives section will define the goals set for the controller. In the Methods and Results part the approach toward achieving fast time scale modulation will be described in detail and its functioning will be demonstrated. In the Discussion part attention will be drawn to some important factors about the training of the controllers, the results will be analysed and advantages and disadvantages of both approaches will be presented. As conclusion, some ideas for future development of fast time scale modulating architectures will be provided.

2 Objectives

Controlling amplitude and frequency of the pattern is considered to be low level control of a system. From a biological standpoint this action is realized by control mechanisms in the spinal cord. However, if successful, it provides a low level control of the locomotion. It is believed, that the high level control comes from the brain. Such commands ensure obstacle avoidance for example. The mechanism for this kind of control is different from the low level commands and does not stand above them in a hierarchy, but rather connects to the structure in its own way and modulates its output while the desired behavior maintained by the lower level controllers is still in tact. The result is an agent that can perform simultaneously a low level task while doing another one dictated by the high level control. This can lead to mixture of behaviors, which is highly desirable. An example for this could be a hurdles competition in which the competitor should run on the track while jumping over the obstacles. It is desired that the high level control should make the person jump, while still moving forward and then ensure that the system (runner) will return fast enough to its original state of running. In other words, the high level control needs to realize a transition to the previous state after the desired action is performed.

The goal of the developed technique is to train a mechanism that performs a fast time scale control of its output. The controller will not use as input the desired characteristics of the pattern but rather a pulse signal coming from a high level control structure. The idea is to add a behavior to an already existing one (modulating the controller), which means that it should be, for example, able to avoid any obstacle by exploiting high level commands, while preserving the behavior dictated by other modules of the architecture. The controller should be able to create an immediate change and then smoothly return the system to its original state before it took over. The instantaneous reaction of the network to the input can allow an agent to incorporate reflexive reactions to sensory input without introducing serious effect on other already present behaviors. This mechanisms can also be used as a pre-step to a more gradual change of some of the parameters of the pattern, while a slower time scale control mechanism takes over.

3 Methods and Results

In this section echo state networks will be introduced and the training of the pattern generator together with a detailed description of the controller will be presented. The learning process of the modules will be described and its performance will be demonstrated.

3.1 Echo state networks

Echo state networks (ESNs) are a type of recurrent neural networks (RNNs), which are an object of research in the field of reservoir computing (RC). The nature of RC is thoroughly explained and investigated in [10]. In short, a non-linear dynamical system with parameters that are called weights is used to expand the input in a high-dimensional space called a reservoir. The states of the reservoir are then linearly combined in a readout layer of neurons. This is the only layer of the system that needs to be learned as explained originally in [11] and [12].

An ESN consists of input layer, reservoir and output readout layer. The input weights and reservoir weights are randomly initialized and fixed. The reservoir neurons can have different activation function which leads to a non-linear system. Usually one uses the sigmoid function or the logistic sigmoid function. The readout weights are the only weights that are trained in the system. The output layer might also use an activation function depending on the task at hand. The fact that only one set of weights has to be learned allows for simple learning of the network using linear regression or different variations of it like ridge regression for example. The network states are updated in the following way:

$$x[n] = f(Wx[n-1] + W^{in}u[n] + W^{fb}y[n-1]) \tag{1}$$

$$y[n] = g(W^{out}x[n]) \tag{2}$$

where $x[n]$ is a vector containing the state of the reservoir at time step n, W is its weight matrix, $u[n]$ is the input vector at time step n. W^{in} are the weights of the input, W^{fb} are the feedback weights from the output to the reservoir, $y[n]$ is the output of the network at time step n, and W^{out} are the output weights. One might also decide to use leaky neurons as described in [13]. Here using a parameter called the leak rate, the dynamics of the network can be slowed down or accelerated in order to allow it to deal with more slowly or fast varying input. The equations in the case of a leaky integrator echo state network are:

$$x[n] = (1-\alpha)x[n-1] + \alpha f(Wx[n-1] + W^{in}u[n] + W^{fb}y[n-1]) \tag{3}$$

$$y[n] = g(W^{out}x[n]) \tag{4}$$

where α is the neuron's leaking rate. The function f is the activation function of the network and g is the activation function of the output neurons, which can be chosen by the designer of the network depending on the task at hand. The internal weight matrix needs to be scaled in order to guarantee the stability of the network while it still exhibits rich dynamics. The matrix needs to be rescaled such that its spectral radius guarantees the Echo State Property as initially stated in [12] and later further discussed in [14]. The echo state property basically means that the reservoir has fading memory which vanishes asymptotically. Other than the scaling of the internal weight matrix the reservoir dynamics depends on other factors. The input scaling, the bias term (a constant valued input to the network) and the actual input that is driving the reservoir are very important for the performance of the network and can change drastically its dynamics as examined in [15]. The scaling of the input weights might be used to steer the reservoir towards its saturation level, which will make it more non linear but will reduce the memory capacity as well as demonstrated in [16]. If the feedback weights are non-zero as in the case of pattern generators the stability of the network is no longer guaranteed. In [17] a solution which introduces state noise during training is proposed. If the echo state has leaky integrator units the echo state property is guaranteed depending on the respective coefficients as described in [13].

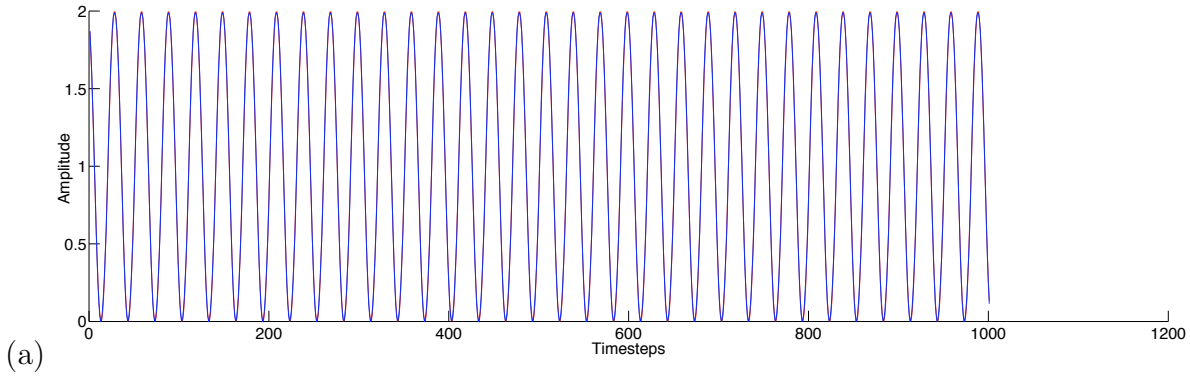
3.2 Training the pattern generator

There are several advantages of using pattern generators realized by echo state networks rather than coupled differential equations as in [18] or other RNNs. ESNs are more easily trained as shown in [19] and due to their high dimensionality can produce more complicated patterns as in [13] and [20] than other implementations of central pattern generators. In order to implement a pattern generator using the echo state network framework one has to use a generic network with no input and non-zero feedback weights. The update equations then become:

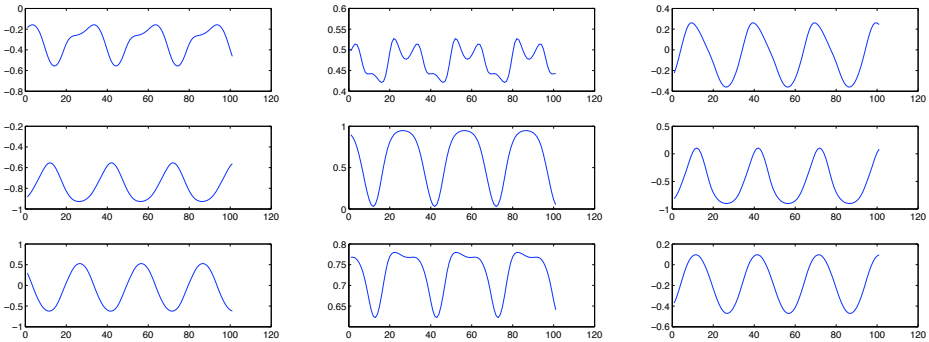
$$x[n] = f(Wx[n-1] + W^{fb}y[n-1]) \tag{5}$$

$$y[n] = g(W^{out}x[n]) \tag{6}$$

We assume a network size of N. The network state $x[n]$ at time step n is of size $N \times 1$ and is initially set to all zeros. A bias is added as the only input to the network providing a constant input of ones. The input weights W^{in} for the bias (size $N \times 1$) are drawn from a uniform distribution $[-0.5\, 0.5]$. The internal weight matrix W (size $N \times N$) is once more drawn from a uniform distribution over $[-1\ 1]$ and are then scaled so that the spectral radius is equal to 1.5. The feedback weights W^{fb} (size $1 \times N$) are as well drawn from a $[-1\ 1]$ uniform distribution. The connectivity of the network is set to 20% . For this task I use a network size $N = 100$ and the tanh activation function is used for the internal states. The training data is a simple sine wave with period $T = 30$ time steps.



(a)



(b)

Figure 1: Training of the pattern generator: a) target signal (blue) vs network output (red) during the testing phase. b) Activations of randomly chosen internal units of the PG during a test run.

A training data series of length 20000 time steps was generated. The output of the network is forced during the learning procedure, meaning that what actually comes into the network from the output unit through the feedback weights is the teacher signal. If we look at this from another angle, what is actually happening is that the network is trained on a one time step prediction task with its input arriving from the output unit. The output weights W^{out} (size $N \times 1$) are the only parameter to be calculated during the learning phase. I use standard ridge regression with Tychonov parameter $\alpha = 0.00001$, which results in normalized mean square training error (NRMSE) of 0.00034398. The mean absolute output weights size is 0.0627 . It becomes clear from Figure 1 and the error measures that the network has successfully learned to oscillate in the desired manner producing a sine wave at its output.

3.3 Training of the controller: Addition of internal units

This approach aims at training a mechanism that modulates either the frequency, the amplitude or the offset of the pattern. It does so by adding internal units to the network that do not interfere directly with the other units but only through

the output feedback. Only the output weights of this additional batch of neurons are trained. The training procedure goes as follows:

- b new internal units are added to the network. The internal weight matrix W is changed in the following way: The weights $W^{oldtonew}$ from the already existing units to the new ones and the weights $W^{newtonew}$ between the new units are drawn from a uniform distribution $[-0.10.1]$. The weights $W^{newtoold}$ from the new units to the old ones are set to 0, so that they do not interfere with the network's dynamics. The feedback weights $W^{fbtonew}$ from the output to the new neurons are also set to 0. The input weights $W^{intonew}$ of the bias are drawn from a uniform distribution $[-1\ 1]$. The output weights $W^{outfromnew}$ from the new units are initially set to 0.

- A teacher signal y is designed such that it oscillates in the desired manner. For example if a decrease in frequency is to be implemented by the controller, the teacher signal is simply a sinusoid with the same amplitude and offset as the original but with lower frequency.

- The PG is then run forcing y at the output. The internal states during this run are collected. The states of the new neurons are collected and stored in a matrix P. The output weights W^{outnew} of the new neurons are then calculated so that they produce as output the difference between the target signal y and the actual output of the network y_n during the forced run. Assuming the length of y and y_n is L, ridge regression is once again used for this task to minimize the error E:

$$E(P \times W^{outnew}, (y - y_n)) = \frac{\sqrt{\sum_{i=1}^{L}((y[i] - y_n[i]) - P[i] \times W^{outnew})^2}}{L} \tag{7}$$

- Once the additional output weights $W^{outfromnew}$ are computed the new functionality of the PG is ready for use. The equation describing the calculation of the output of network can be rewritten in the following form:

$$y[n] = g(W^{out}x[n] + m[n]W^{outfromnew}x^{new}[n]) \tag{8}$$

  where $x[n]$ is the state of the original network at time step n, $x^{new}[n]$ is the state of the newly added batch of neurons at time step n and $m[n]$ is the value of the modulation signal at time step n.

  In Figure 2 one can see a depiction of the described architecture and on Figure 3 is the above process described in graphics.

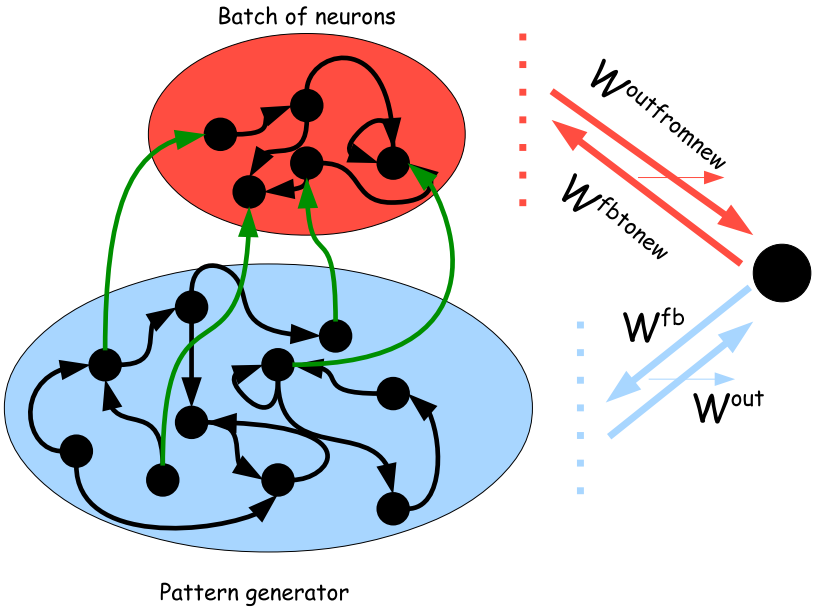


Figure 2: A depiction of the described architecture. It can be noticed that the only weights that are trained are W^{out} and $W^{outfromnew}$.

3.4 Qualitative observations on the modulation performance

In order to see how well the controller modulates a single parameter 3 experiments were designed:

The same already trained pattern generator was used for all three setups. The network was trained to produce a sinusoid with period 30 time steps, amplitude 1 and shift 1. A batch of 10 neurons was added to the network. Three different types of teacher signal were used to train the output weights of the newly added neurons. Table 1 documents the parameters of the teacher signals and the training errors.

Table 1: Teacher signals specifications and training results

	Period	Amplitude	Shift	NRMSE	Mean output weights
Exp. 1 (FM)	90	1	1	7.2078e-004	1.3986
Exp. 2 (AM)	30	3	1	0.1713	0.3155
Exp. 3 (SM)	30	1	2	4.5760e-004	0.9657

After the additional output weights are calculated a modulation signal is designed for each experiment.

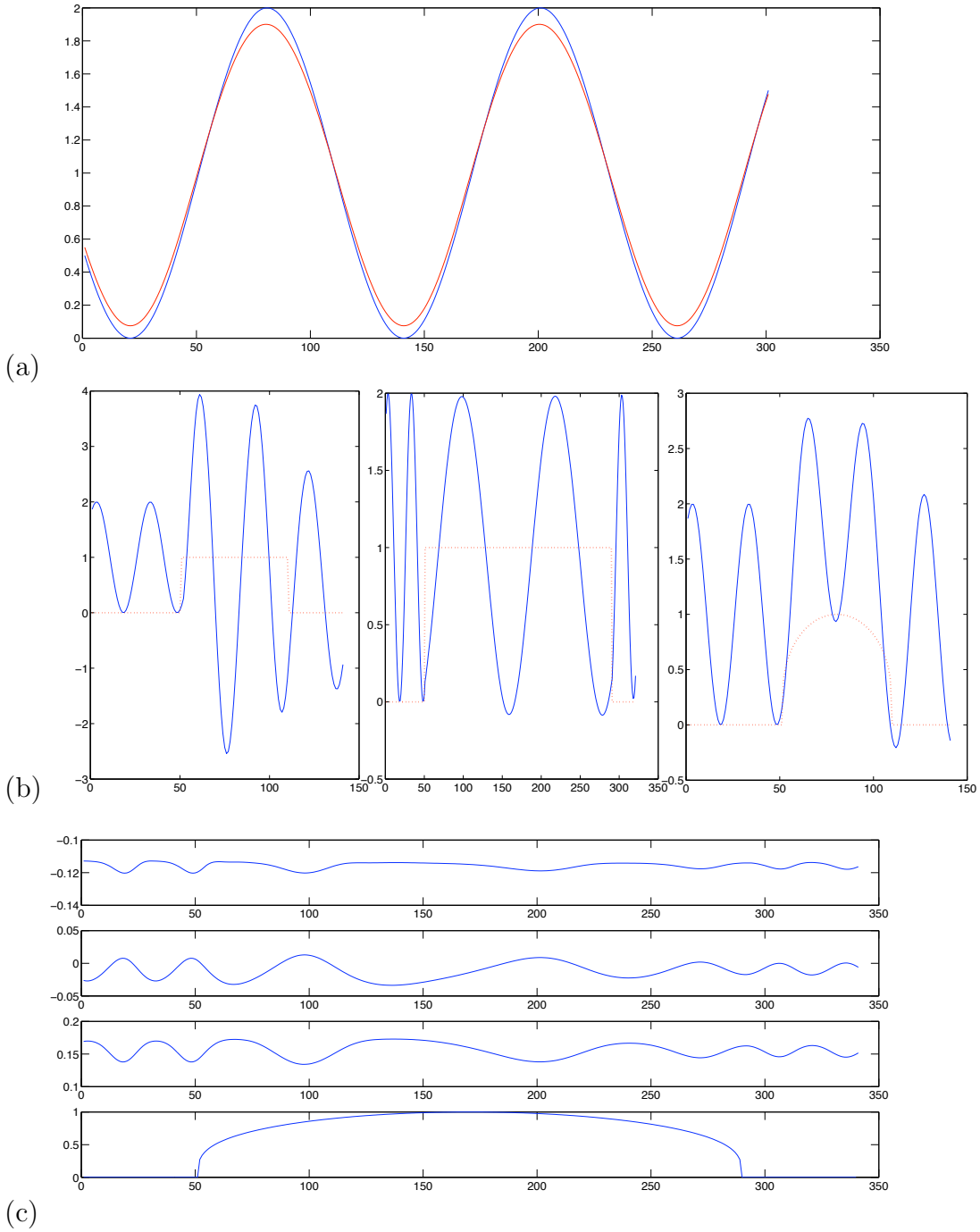


Figure 3: Training procedure and results: a) Comparison of the forced output of the network y (blue) and the actual output that the network produces (red). b) Output of the network when the controller is trained for respectively frequency, amplitude and shift modulation aligned with the modulation signal. The controller is activated at step 50. c) States of some of the neurons in the added batch at the moment where the controller, which is trained for frequency decrease is activated at step 50 aligned with the modulation signal.

$$m[n] = A_m sin(\frac{2\pi n}{2W})^{\alpha} \tag{9}$$

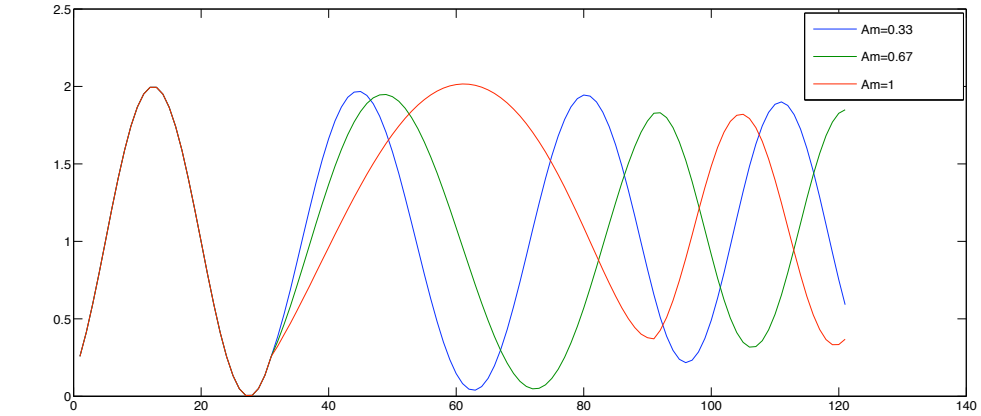
This is also known as a cosine (sine) window, where n varies from 1 to W and W is the width of the window. The parameter α is used to control the steepness of the window function and A_m is the amplitude of the modulation signal. Figure 6 demonstrates the change of the measure ρ as the parameter α is varied for frequency, amplitude and shift modulation. In the case of frequency and amplitude modulation a cosine window with parameter $\alpha = 0.1$ was used and for shift modulation $alpha = 0.3$. The network is then run for different values of the amplitude of the modulation signal for all three experiments. The results can be seen in Figure 4.

3.5 Measuring the smoothness of the modulation transition

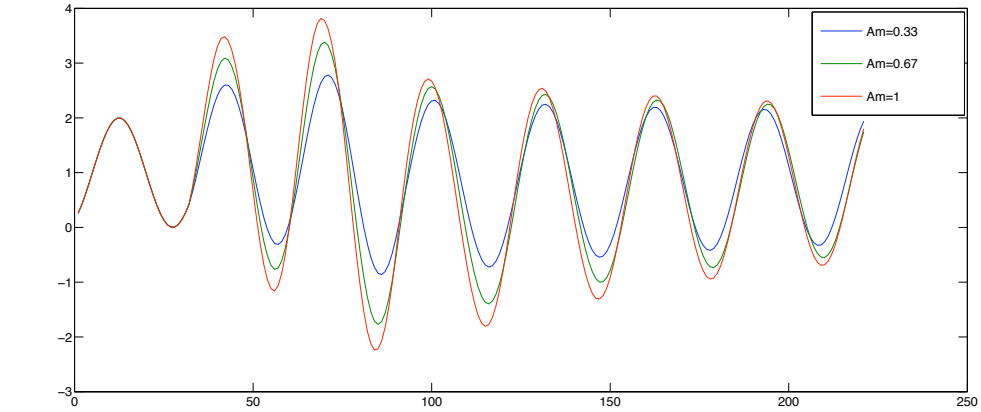
One measure that will be used to evaluate the performance of the controller is the smoothness of the transition when the controller is activated. In order to do that we look at the first and second derivatives of the signal around the point where the controller is activated. Assume we take N time steps before the transition and N time steps after the transition. Then we have a signal c of length $2N+1$. By taking the derivative of the signal c or rather a pseudo derivative $c'[i] = c[i] - c[i-1]$ (which is assumed to be oscillatory) it can be noticed that the amplitude of the oscillation in c' is of much lower magnitude compared to c. However, if there is any discontinuity present in the original signal, a spike in its derivative emerges and its magnitude becomes greater if the signal is further differentiated. In order to get an intuition of why this happens, assume that the signal c is a sinusoid $c = s + Asin(2\pi tf)$, where s, A and f are respectively the shift amplitude and frequency of the sinusoid, then its n-th derivative (assuming n is odd for simplicity) with respect to time is $c^{(n)} = \pm A(2\pi f)^n cos(2\pi ft)$. When a discontinuity is present the instantaneous frequency at this point f_d is much higher than the frequency of the signal f_c. It might even be assumed that the term $(2\pi f_d)$ is smaller than one, while $(2\pi f_c)$ is larger than one, which leads to the observed phenomenon. Of course, this is just an intuitive explanation. Taking the third derivative of c seems to be enough to make the spikes due to the discontinuity seem as outliers. Figure 5 depicts this process. As a result of that, the magnitude of the spikes can be used as a measure of the roughness of the transition. A new signal c_s is then composed of only the values of c''', which are larger than the standard deviation of c'''. Assume c_s has length l of the measure ρ is then defined as:

$$\rho = \sum_{i=1}^{l} |c_s(i)| \tag{10}$$

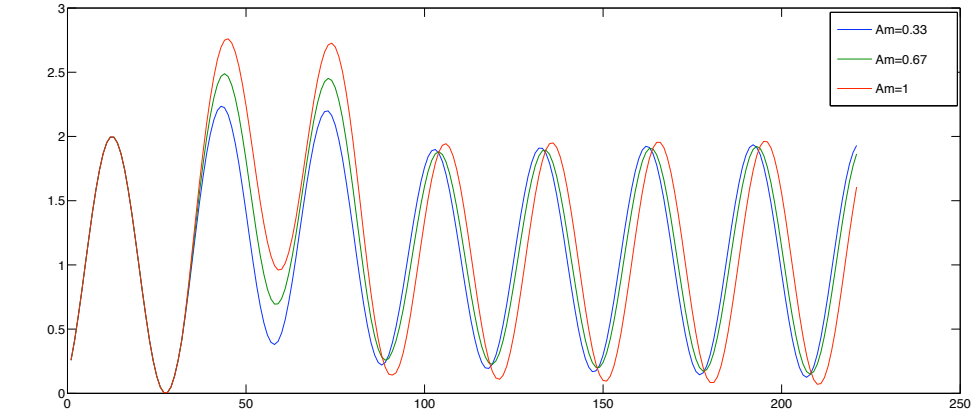
This is not an absolute measure of the transition quality, however it can be used to compare different transitions. In general, discontinuities in the original



a)



b)



c)

Figure 4: Varying the amplitude of the modulation signal for a) Frequency modulation b) Amplitude modulation c) Shift modulation. The modulation starts at time step 30 and ends at time step 90.

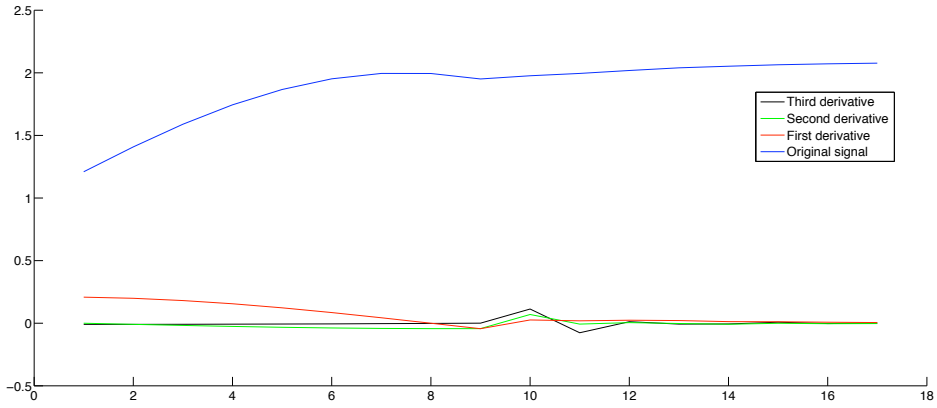


Figure 5: Derivatives of a signal with a discontinuity at the transition point.

signal as in Figure 5 are very rarely observed. Usually such events occur when the controller is activated around the maximum or minimum of the signal. The spikes that are observed in the derivatives are then a measure of how abruptly the signal changes. In order to use this measure the modulation signal $m[n]$ as defined in Equation 9 is used.

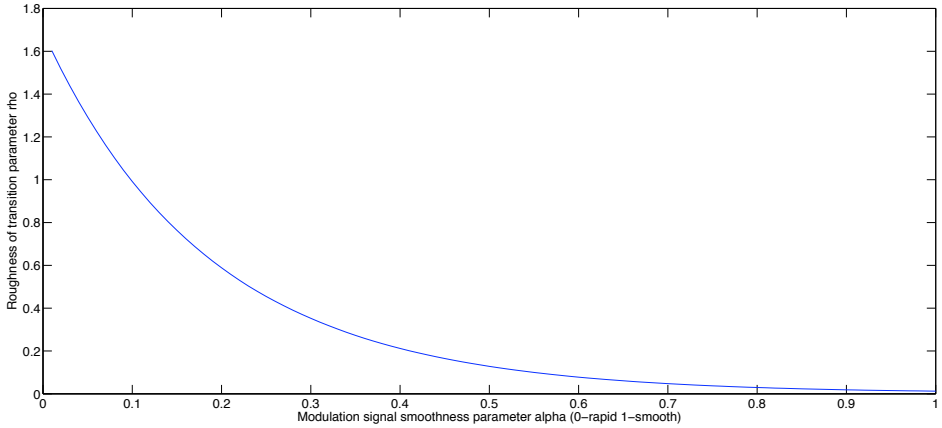
Figure 6 demonstrates the change of the measure ρ as the parameter α is varied for frequency, amplitude and shift modulation.

4 Discussion

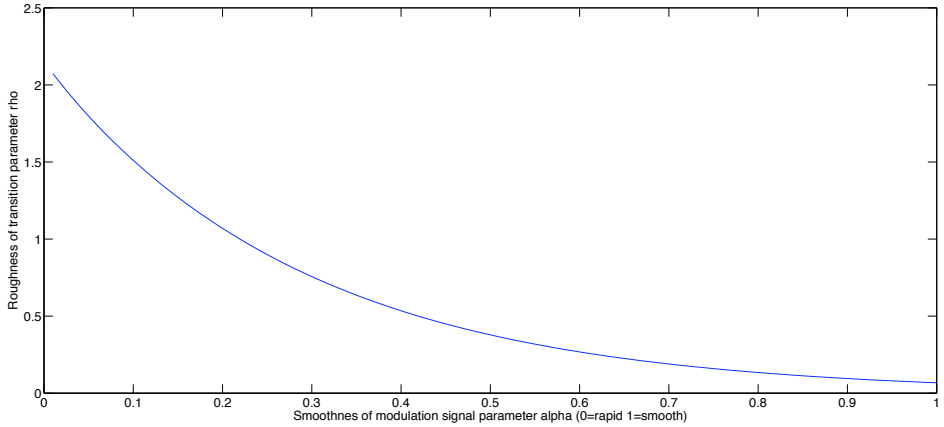
The discussion section will be structure as follows: at first some important and interesting insights into training the pattern generator will be presented. Next there will be a short discussion regarding the main challenges of fast time scale modulation. Thirdly, the performance of the approach will be evaluated and discussed.

4.1 Insights of training a pattern generator

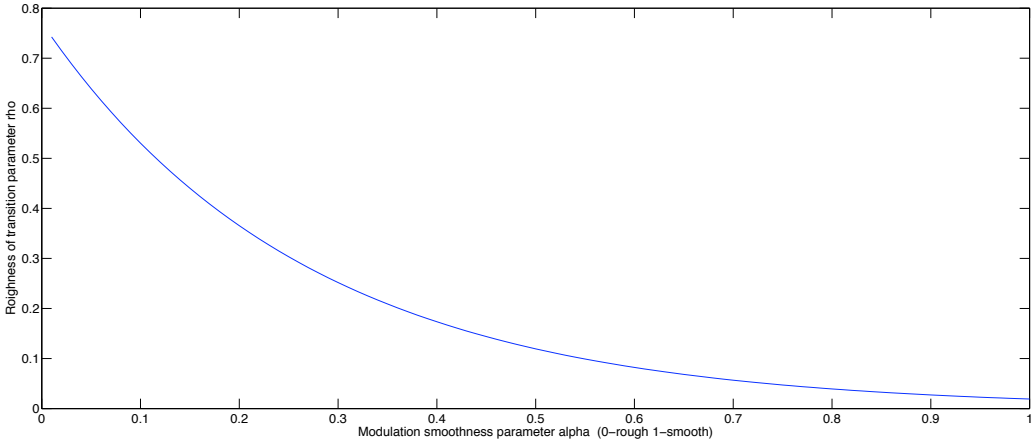
The pattern generator training method presented in the previous section is a very simple one. The reason for this choice is the objective that the control mechanisms should be able to perform well regardless of the PG they are presented with. However, attention should be given to some properties of the pattern generator that lead to better results. The most important factor is that the network needs to be flexible in order to be controlled. Flexibility here means that the generator should be able to easily change the parameters of its oscillation while still behaving in a stable manner and that the time it takes to return to its oscillation prior to the modulation should be minimal.



a)



b)



c)

Figure 6: Measures of smoothness of transition for the a) frequency modulation b) amplitude modulation c) shift modulation.

It was noticed that CPGs that exhibit lower amplitudes of the internal units and smooth sinusoid activations result in a network that cannot follow closely the forced signal and returns more slowly to its initial oscillation after the modulation. However, if the activations of the internal units have greater amplitude and occupy both the linear and non linear part of the activation function, the performance of the control mechanism is improved in the sense that the network can more easily replicate the forced signal and it returns almost instantly to its original oscillation when the controller is switched off. However, if we have a network with high amplitude pure sinusoid oscillations, the transition when the controller is activated is smoother. So there is a trade off between smoothness at the beginning of the modulation and the recovery time after the modulation. In the end I would recommend that it is more beneficial to train a network with high amplitude activations that have both linear and non linear parts. Those conditions can be obtained by gradually changing the spectral radius of the network until the point where the network starts behaving chaotically is reached. By keeping the spectral radius just below this value one can ensure a rich behavior. The bias input weights scaling and the feedback weights scaling also play a vital part here. They can be used to "steer" the network towards non linearity or to increase the magnitude of the oscillations of the units. In Figure 7 an example of two networks that both produce the same output is shown. One has a stable purely sinusoid activations of the internal units while the other exhibits the rich behavior that is desired.

4.2 Challenges of fast time scale modulation and why this approach works

When trying to implement a controller that should perform fast time scale modulation it should be: able to generate a modulated version of the pattern; able to perform smooth transition to the modulation; able return the network to its original behavior; stable.

A rigorous explanation of why adding the batch of neurons actually is able to perform well in some of those tasks is far away from the scope of this report. However, an explanation on the intuitive level can be attempted. The added neurons act as an observer or "sampler" of the original network. They do not interfere with the network's dynamics except through the feedback weights of the output unit, however they receive information about the state of the generator from their connections to the original units. Their activation closely resemble the oscillations of the original neurons in the original network. This gives them the potential to produce a linear combination of their activations that can mimic a signal that is also a result of the network's dynamics. Such a signal is the difference between the teacher signal and the actual output of the pattern generator during the forcing phase of the training. As it can be seen in Figure 3 the output of the network does not perfectly follow the target signal. This difference propagates to the network through the feedback weights and as a result the internal activations

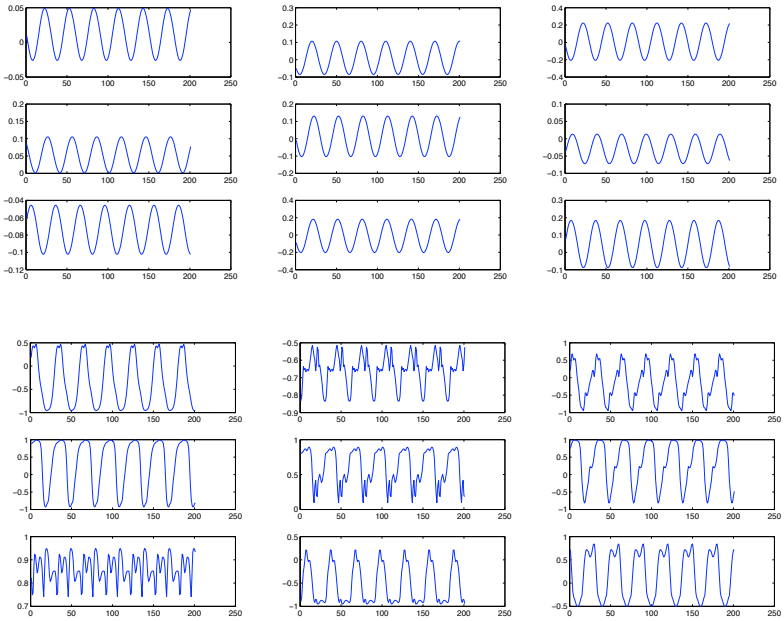


Figure 7: Activations of internal units of two networks producing the same output. a) This network has spectral radius of 0.1 input weights scaling of 0.1 and feedback weights scaling of 0.1. The activations are scaled, phase shifted copies of the output signal. b) This network has spectral radius of 1.5 input weights scaling of 1 and feedback weights scaling of 1. The activations are not copies of the output signal have high amplitude and cover both the linear and non linear part of the activation function.

of the units are slightly different, which leads to an even greater discrepancy between the output and the target. This avalanche effect results in the network returning to its normal state. By training a separate group of neurons to correct for that difference, the feedback that comes from the output unit is exactly right, so that the network will keep oscillating in the desired manner. Another approach can be taken by training the weights from the batch of neurons to correct the activations of the internal units. However, in this case the dynamics of the network is compromised and smaller errors in the computation of those weights can lead to unsatisfactory performance of the controller. The training procedure seems to do a bit more than just training the added batch of neurons to keep the network from returning to its original state. It learns how to bring a signal that is different from the target that was used for training from its original oscillation to the actual teacher signal.

The issues of the transitions can be solved by avoiding the activation of the controller near the peaks of the signal and by employing a smooth modulation signal. As demonstrated in Figure 6, the decreasing of the slope of the modulation signal reduces the roughness factor of the transition. This method however, works for the transitions to the modulation. Once the modulation is over, it seems like the controller fails to bring the network back to its original dynamics. There are no discontinuities present, however it might take several periods for the generator to return to its original oscillation. The length of the recovery period seems to depend mainly on the value of the output unit when the modulation is stopped and naturally on the pattern generator itself. The slope of the modulation signal does not seem to have any effect on that. A possible solution to the problem might be to calculate a second set of weights that perform the task of returning the network to its original oscillation from another oscillatory pattern. An easy way to do this would be to use the already computed weights that make the network oscillate in a modulated fashion (or in other words perform constant modulation on the signal), then force the original signal at the output unit, compute the difference between this teacher signal and the actual output and calculate the second set of weights that should be able to return the network to its original state.

The challenge of stability, unfortunately, depends mainly on the pattern generator that is modulated. If the network itself is operating on the edge of chaos it is quite difficult for the controller to avoid radical behavior. This is due to the fact that the network is simply not able to oscillate in the desired manner.

4.3 Evaluation of performance

The performance of the controller demonstrates that it is able to realize fast time scale modulation of a pattern. The change of frequency, amplitude or shift is instantaneous, meaning that it can be observed even in the first time step after the activation of the controller, which is what fast time scale modulation requires. An important advantage of the controller is that it can control just the frequency,

amplitude or the shift of the signal, while the parameter that should stay constant exhibits minimal change.

However, there are some issues that should be mentioned here, concerning this technique for modulation. First of all, even though the change in parameters of the signal is obvious and very well demonstrated, one should be careful when the activation of the batch of neurons should be started and more importantly ended. The phase of the pattern at which the activation is finishes is vital for the smooth transition of the network to its new state. This is mainly a problem for the frequency modulation task. A good practice that guarantees satisfactory results is to stop the modulation near a maximum value of the pattern. This usually ensures fast recovery of the generator. This might be seen as biologically plausible phenomenon. Just as the hurdles runner will not change the frequency of its step right in the middle of going over an obstacle, the network might also experience difficulties if it is expected to perform a similar unnatural task.

The transition between the normal state of the controller and the desired one cannot be both smooth and instantaneous at the same time. There is a trade off between the two and the choice of the smoothness of the modulation signal is entirely dependent on the application for which the controller is to be used.

The modulation signals that are used for the activations of the controllers should come from a higher control level in the architecture. It might be useful to see the mechanism presented in this report as a knob that can change the behavior of the generator. Once the high level controller has it at its disposal it might learn how to employ it and come up with different activation sequences in order to achieve the desired modulations.

5 Conclusion

A method for modulating an output of a pattern generator was demonstrated and discussed. It adds a batch of neurons to the pattern generator. Their output weights are trained so that when they are activated the frequency, amplitude or the shift of the signal can be modulated separately. The controller can very effectively change the a desired characteristic of the output, while leaving the other characteristics unchanged. The challenges that fast time scale modulation presents were discussed and their solutions in the case of the proposed approach were analysed. Measures of the performance of such a controller were defined, and a study on the effects of the modulation signal to the transition was carried out. The importance of the training of the pattern generator was noted and some observations were made about the desired dynamics of a reservoir in order for the controllers to perform at their best. However, the method should deliver satisfiable results regardless of the characteristics of the PG.

Acknowledgement. The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 – Challenge 2 – Cognitive Systems, Interaction, Robotics – under grant

agreement No 248311 – AMARSi.

References

[1] Q. Wu, C. Liu, J. Zhang, and Q. Chen. Survey of locomotion control of legged robots inspired by biological concept. *Science in China Series F: Information Sciences*, 52:1715–1729, 2009. 10.1007/s11432-009-0169-7.

[2] M. Dimitrijevic, Y. Gerasimenko, and M. Pinter. Evidence for a spinal central pattern generator in humansa. *Annals of the New York Academy of Sciences*, 860(1):360–376, 1998.

[3] M. Ronald and Harris-Warrick. Neuromodulation and flexibility in central pattern generator networks. *Current Opinion in Neurobiology*, 21(5):685 – 692, 2011. Networks, circuits and computation.

[4] H. Cruse. What mechanisms coordinate leg movement in walking arthropods? *Trends in Neurosciences*, 13(1):15 – 21, 1990.

[5] H. Cruse and S. Epstein. Peripheral influences on the movement of the legs in a walking insect carausius morosus. *Journal of Experimental Biology*, 101(1):161–170, 1982.

[6] B. Blaesing and H. Cruse. Stick insect locomotion in a complex environment: climbing over large gaps. *Journal of Experimental Biology*, 207(8):1273–1286, 2004.

[7] V. Duerr, J. Schmitz, and H. Cruse. Behaviour-based modelling of hexapod locomotion: linking biology and technical application. *Arthropod Structure Development*, 33(3):237 – 250, 2004. Arthropod Locomotion Systems: from Biological Materials and Systems to Robotics.

[8] T. Waegeman, F. Wyffels, and B. Schrauwen. Towards a neural hierarchy of time scales for motor control. In Tom Ziemke, Christian Balkenius, and John Hallam, editors, *From Animals to Animats 12*, volume 7426 of *Lecture Notes in Computer Science*, pages 146–155. Springer Berlin Heidelberg, 2012.

[9] J. Li and H. Jaeger. Minimal energy control of an esn pattern generator. Technical report, Jacobs University Bremen, 2011.

[10] D. Verstraeten, B. Schrauwen, M. DeHaene, and D. Stroobandt. An experimental unification of reservoir computing methods. *Neural Networks*, 20(3):391 – 403, 2007. Echo State Networks and Liquid State Machines.

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

[12] H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks. Technical report, German National Research Center for Information Technology, 2001.

[13] H. Jaeger, M. Lukosevicius, D. Popovici, and U. Siewert. Optimization and applications of echo state networks with leaky- integrator neurons. *Neural Networks*, 20(3):335 – 352, 2007. Echo State Networks and Liquid State Machines.

[14] H. Jaeger and Manjunath. Echo state property linked to an input: Exploring a fundamental characteristic of recurrent neural networks. *Neural Computation*, 25(3):671 – 696, 2012.

[15] D. Verstraeten and B. Schrauwen. On the quantification of dynamics in reservoir computing. In *Proceedings of the 19th International Conference on Artificial Neural Networks*, pages 985 – 994.

[16] D. Verstraeten, J. Dambre, X. Dutoit, and B. Schrauwen. Memory versus non-linearity in reservoirs. In *Neural Networks (IJCNN)*, pages 1 – 8.

[17] H. Jaeger. Tutorial on training recurrent neural networks, covering bptt, rtrl, ekf and the "echo state network" approach. Technical report, German National Research Center for Information Technology, 2002.

[18] S. Degallier, L. Righetti, S. Gay, and A. Ijspeert. Toward simple control for complex, autonomous robotic applications: combining discrete and rhythmic motor primitives. *Autonomous Robots*, 31:155–181, 2011. 10.1007/s10514-011-9235-2.

[19] H. Jaeger and H. Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304(5667):78–80, 2004.

[20] A. Krause, V. Duerr, B. Blaesing, and T. Schack. Multiobjective optimization of echo state networks for multiple motor pattern learning. In *18th IEEE Workshop on Nonlinear Dynamics of Electronic Systems, (NDES 2010)*, pages 190 – 193, 2010.