Graz, University of Technology

Institute for Theoretical Computer Science (IGI),

Graz University of Technology

A-8010 Graz, Austria

Master's Thesis

Temporally correlated exploration noise for reward-modulated learning of reservoir models

Submitted by:

Michael Steinegger

Supervisor:

Assoc. Prof. Dr. DI. Robert Legenstein

Graz University of Technology, Austria

4. January 2014

Technische Universität Graz

Institut für Grundlagen der Informationsverarbeitung (IGI),

Technische Universität Graz

A-8010 Graz

Masterarbeit

Zeitlich korreliertes Explorations Rauschen für Reservoir Modelle

Vorgelegt von:

Michael Steinegger

Betreuer:

Assoc. Prof. Dr. DI. Robert Legenstein

Technische Universität Graz, Österreich

4. Januar 2014

Abstract

In humans and monkeys, projection neurons in primary motor cortex act as the main signal source for spinal networks and thus act as the output stage of cortical motor control circuits. Recently computational models - called liquid state machines or reservoir computing - have emerged that mimic this structure through the use of readout neurons. Synaptic plasticity of such readout neurons can be achieved through reward-based learning strategies, such as the Exploratory Hebb (EH) Rule. In the EH-rule, changes in synaptic efficacies are driven by correlations between stochastic neuronal responses (neuronal noise) and a global reward signal that measures system performance on the task at hand. These measurements require the noise to have an immediate impact upon system behaviour. This thesis investigates the ability of such reservoir computing models to perform motor control tasks. In systems bound by physical mass constraints, inertia and friction effects might delay or filter rapidly changing noise. We employ temporal correlation of neuronal noise signals to mitigate these filtering effects. Our results show that movement of an agent can be successfully directed by using temporally correlated exploration noise for optimizing the weights of the readout neurons of a reservoir driven controller.

Kurzfassung

In Menschen und Affen agieren Projektions Neuronen des primären Motor Cortex als Hauptsignalquelle für spinale Netzwerke und agieren so als Ausgangsstufe für kortikale Motor-Regelkreise. Neuartige neuronale Rechenmodelle - genannt liquid state machines oder Reservoir Computing - imitieren diese Struktur durch die Benutzung von Readout Neuronen. Synaptische Plastizität solcher Readout Neuronen kann durch verstärkende Lernstrategien erreicht werden, wie die explorative Hebb (EH) Regel. In der EH-Regel werden Änderungen in synaptischer Verbindungsstärke durch Korrelationen zwischen stochastischen neuronalen Ausgangssignalen (neuronalem Rauschen) und einem globalen Belohnungssignal, welches den Systemerfolg misst, bewirkt. Diese Strategie erfordert, dass das Rauschen unmittelbare Wirkung auf das Systemverhalten hat. Diese Arbeit untersucht die Fähigkeit eines solchen Reservoir computing Modells, Motorkontrollaufgaben zu bewältigen. In Systemen in denen Massen bewegt werden müssen (z.b. Roboterarme), verzögern und filtern Trägheits- und Reibungseffekte schnell variierendes Rauschen. In dieser Arbeit werden zeitliche Korrelation von neuronalen Rauschsignalen benutzt um diese Filter Effekte zu schwächen. Unsere Resultate zeigen dass die Bewegung eines Agenten erfolgreich durch ein reservoir computing System gesteuert werden kann wenn zeitlich korreliertes Explorationsrau-

schen verwendet wird um die Gewichte der Readout Neurone zu optimieren.

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

....

(Place, Date) (Signature)

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

....

(Ort, Datum) (Unterschrift)

Contents

Contents ii

List of Figures iii

1 Introduction 1

2 Background 4

2.1 Liquid Computing 4

2.2 Reward modulated learning 10

2.3 Reward based learning of readouts 12

3 Model 15

3.1 Reservoir model 17

3.2 Exploration noise 19

3.3 Physical model 19

4 Results 29

4.1 Simulation results for 1D model 29

4.2 Simulation results for 2D model 34

4.3 Simulation results for robot arm 42

5 Discussion 46

5.1 Future work 48

5.2 Conclusion 48

A Time-discrete Formulas **49**

B Matlab Code **51**

Bibliography **57**

List of Figures

2.1 Reservoir 7
2.2 Reservoir with Feedback 8

3.1 Basic Model 16
3.2 Model of a 2-link robotic arm 21
3.3 Motor Control System 23
3.4 Robot Arm Feedback Kernels 28

4.1 Noise Amplitude Histograms 30
4.2 Noise Output 31
4.3 1D System Output 32
4.4 1D Targets reached during Training 33
4.5 1D System neuron activity 34
4.6 2D System 35
4.7 2D neuron activity 36
4.8 2D Targets reached during Training 37
4.9 Reconstruction of exploration noise z_ξ 39
4.10 Noise reconstruction with large τ_c 40
4.11 Performance of different τ_c 41
4.12 Robot arm movement trajectories during training 42
4.13 Robot arm movement trajectories during testing 43
4.14 Feedback and neuron activity during training 44
4.15 Target positions reached during training 45

Chapter 1

Introduction

Ever since the basic principles of computation in biological neurons have been known, there have been attempts to emulate it. Many different architectures based on networks of neural elements have been devised over time. The first computational model of a neuron was the "linear threshold unit (TLU)" or McCulloch-Pitts neuron named after its creators [McCulloch and Pitts, 1943]. Other models like the influential perceptron [Rosenblatt, 1962] soon followed, fuelling a surge of optimism about the capabilities of simple networks consisting of these models. In analytical work it was soon shown [Minsky and Papert, 1988] that the computational power of single layers of these neurons was severely limited in scope and the optimism faded. To overcome these inherent limitations, complexity of the network had to be increased, resulting in multi-layered neural networks [Rumelhart et al., 1986]. Another promising step that increased complexity was the introduction of recurrent connections between units in a neural network by [Hopfield, 1982]. Hopfield networks provided a model for associative memory by utilising point attractors. Such models opened up a whole field of attractor networks not only for modelling memory, but other biologically inspired processes like motor behaviour and classification tasks [Amit, 1989, Pearlmutter, 1995]. Increasingly difficult tasks forced system dynamics to become more complex, which in turn needed to be controlled

more rigorously to maintain system stability. Models were constructed to circumvent these complex dynamics issues [Maass et al., 2002, Jäger, 2001]. In these so-called Liquid Computing or Reservoir Computing models, a recurrently connected reservoir holds a high dimensional state, and a readout layer extracts desired information. In order to achieve this extraction by readout neurons, their weights are modulated by synaptic plasticity rules which try to approximate a desired output signal. This usage of prior knowledge about the desired output is referred to as supervised learning. In the context of biological systems it might prove problematical to assume prior knowledge about optimal output of a given network, especially in motor-control tasks where translation of desired movement trajectory into actual control signals plays a significant part.

Alternatively, weight modulation can be achieved through reward-based learning [Hoerzer et al., 2011]. The prior knowledge of desirable output is replaced by a measure of success in form of a reward function. The signal generated by this function indicates progress towards a desired state. It was shown by [Hoerzer et al., 2011] that such a reward modulated network is able to carry out a variety of quite complex computational tasks. However when considering motor control tasks, a significant problem arises through principal system behaviour. The learning rule employed by [Hoerzer et al., 2011] utilizes a temporally uncorrelated noise signal to perform a local search around the slowly changing control signal. This requires the noise signal to have immediate impact upon performance of the system. Real-world mechanical systems are however bound by inertia and as such tend to filter such noise through low-pass characteristics.

In the work presented here we show that this problem pertaining to motor control problems can be overcome by applying temporally correlated noise. By correlating the noise signal over time a lasting motor control impulse is able to overcome low-pass filtering and generate movement on which performance measurement by a reward mechanism can be made. This is shown in the simulation of an abstract two-dimensional (2D) motor control task. Varying degrees of noise correlation and their impact on performance are examined before we move on to a more realistic simulation of a 2-joint robot

arm.

This thesis is organized in the following way:

In Chapter 2 we will explore the concepts of reservoir computing and reward based learning in more detail. We will also take a look at the concept of supervised and unsupervised learning and how it relates to rewards and reward signals.

Chapter 3 gives a definition of our basic model which is used in our simulations. Two representations of physical objects, moving point masses and a 2 joint robot arm, are used and those are also presented.

Throughout Chapter 4 simulations based on this model are laid out and their results presented. The influence of time correlation on exploration noise is demonstrated. It is then tested in two tasks containing increasingly complex internal dynamics.

Chapter 5 concludes with a discussion of our results and the problems encountered.

Chapter 2

Background

2.1 Liquid Computing

In the process of modelling computational abilities of neural pathways, many network architectures and neuron models have been developed and tested. Artificial neural networks have come a long way in simulating finite state machines. But as one moves to more biologically plausible neuron models like spiking neurons, these approaches seize to work. It was shown that this can be achieved by introducing recurrent connections into networks of spiking neurons [Maass, 1996]. But the requirement of a synchronizing mechanism for all neurons weakens their relevance in simulating neural microcircuits. Another problem is the high dimensionality of recurrent networks formed by neural microcircuits. Through recurrent connections the internal dynamics of a system can become very difficult to control. So we want to explore a model for neural computing which does not rely on controlling these internal dynamics and is still able to operate with high dimensional states.
Consider a pool of liquid in motion. It consists of particles flowing around interacting with each other. The entirety of all particle movement vectors can be thought of the internal state of the liquid. By dropping an object into it all particles will change their flow accordingly. Thus the internal state of the liquid as a whole while it is not reverted to a resting state can be thought of

conserving memory about the input object. If we consider rapidly changing input, the internal state of the liquid will probably fluctuate and flow freely in a large state space that is not fixed to finite defined states but more in a liquid state. The goal in this liquid state model is not to extract information about the input from the attractors of the resting state, but rather from the flow through the state space of the liquid state [Maass et al., 2002]. This has the added benefit of preserving the temporal dimensions of the input. In natural occurring signals information about the temporal structure naturally plays a big role and can be crucial in a number of tasks . By using integrate-and-fire neurons which operate in the continuous time domain it can be shown that such an aforementioned liquid state model can effectively encode temporal features of complex input signals [Buonomano and Maass, 2009].
The final task of such a liquid computing model is to extract relevant data from the trajectory of changes in the internal state of the liquid. It can be shown that this data through a high dimensional state space can be separated linearly and processed by networks of readout neurons if their synaptic weights are subject to adaptation [Buonomano and Maass, 2009, Hoerzer et al., 2011].

2.1.1 Liquid State Machine

In this model a single layer of readout neurons z is employed to interpret the internal state s of a randomly interconnected network of neurons which we will call reservoir R (see Figure 2.1). This internal state is generated at time t by all the neurons in R through a filter $\mathcal{F}$ in response to an continuous input signal u:

$$\mathbf{s}(t) = (\mathcal{F}\mathbf{u})(t) \tag{2.1}$$

Commonly called operators in mathematics, $\mathcal{F}$ maps a temporal input stream $\mathbf{u}(\bullet)$ onto the output stream $\mathbf{s}(\bullet)$. The internal state $\mathbf{s}(t)$ is dependent on preceding inputs $\mathbf{u}(s)$ for $\mathbf{s} \leq t$. When R is implemented by a neural circuit this filter can be realised by recurrently connecting some or all of the neurons, providing the capability of fading memory [Maass et al., 2002].

In contrast to finite state machines these liquid states do not have defined discrete transitions between them and are not constructed for a specific task. Rather, the resulting state at time t emerges as a product of the previous state and the input at that time. To obtain desirable output a memoryless readout function z is used. This readout maps the observed internal state $\mathbf{s}(t)$ at time t onto the output $\mathbf{y}(t)$:

$$y_j(t) = z_j\left(\mathbf{s}(t)\right) \tag{2.2}$$

In a neural circuit this can be implemented by a single layer of L readout neurons. In some cases this can even be reduced to a single neuron. These readout neurons z_j look as follows:

$$z_j(t) = \sigma\left(\sum_i w_{j,i} s_i(t)\right) \tag{2.3}$$

The transfer function σ is typically linear, but may sometimes be chosen as a sigmoid function(usually the *tanh* function). While the reservoir R can be generic or randomly chosen, the readout layer z has to be specifically constructed for the given task. Multiple readout units can be used in parallel on the same reservoir for different tasks which require information about the same input sequence [Hoerzer et al., 2011]. The basis of computational power for these liquid state machines can be found within 2 properties presented in [Maass et al., 2002].

The *separation property* of a liquid filter $\mathcal{F}$ addresses the amount of separation between the trajectories of internal liquid states $s(t)$ that are caused by two different input streams $u(\bullet)$. The ability to distinguish between different input streams and represent them accordingly in the internal state is essential for computation and a good separation capability enables to differentiate between input streams at time t with significant differences lying further in the past before t.

The *approximation property* of a readout z addresses its capability to distinguish between different internal states and transform them into given target outputs.

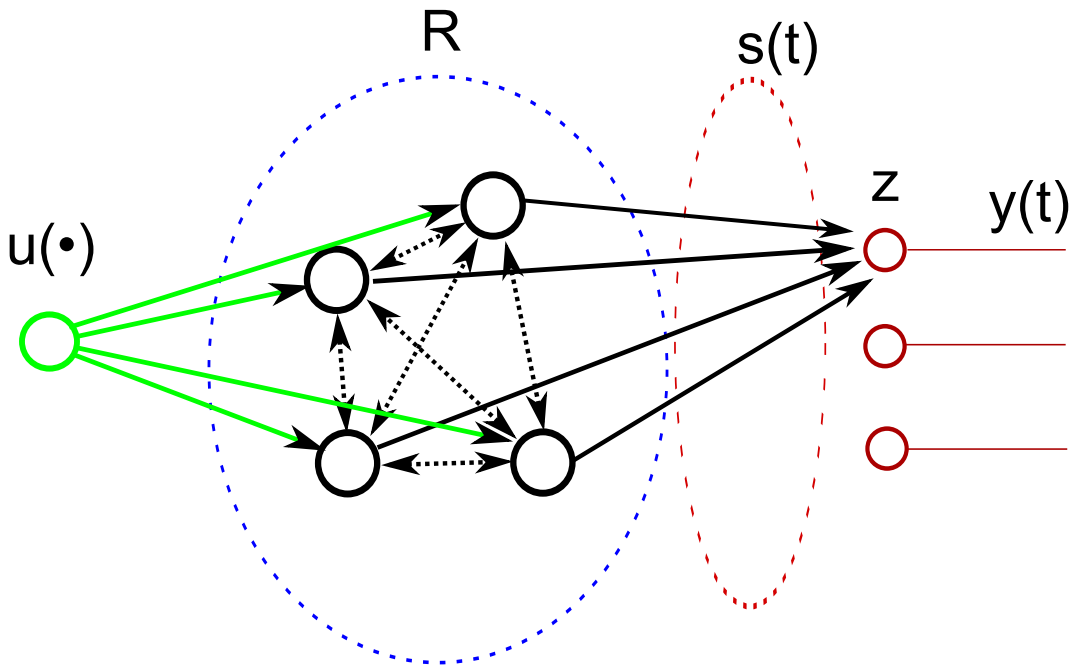

Figure 2.1: Liquid computing model with a reservoir R consisting of recurrently connected neurons, and a layer of readout units z. At every point in time t an internal state $s(t)$ is created by R from the input $u(\bullet)$ and recurrent connections. $s(t)$ is then transformed by z into the output $y(t)$.

On the basis of these properties a universal approximation theorem for liquid state machines was formulated by [Maass et al., 2002]. According to the theorem, if a liquid state machine possesses both separation and approximation property it can approximate any time invariant fading memory filter. Providing the reservoir with feedback from the readout as shown in Figure 2.2 overcomes the limitation of fading memory [Maass et al., 2007]. In the absence of noise the resulting computational model is capable of any conceivable digital or analogue computation on time-varying inputs, and still has significant computing power in noisy conditions.

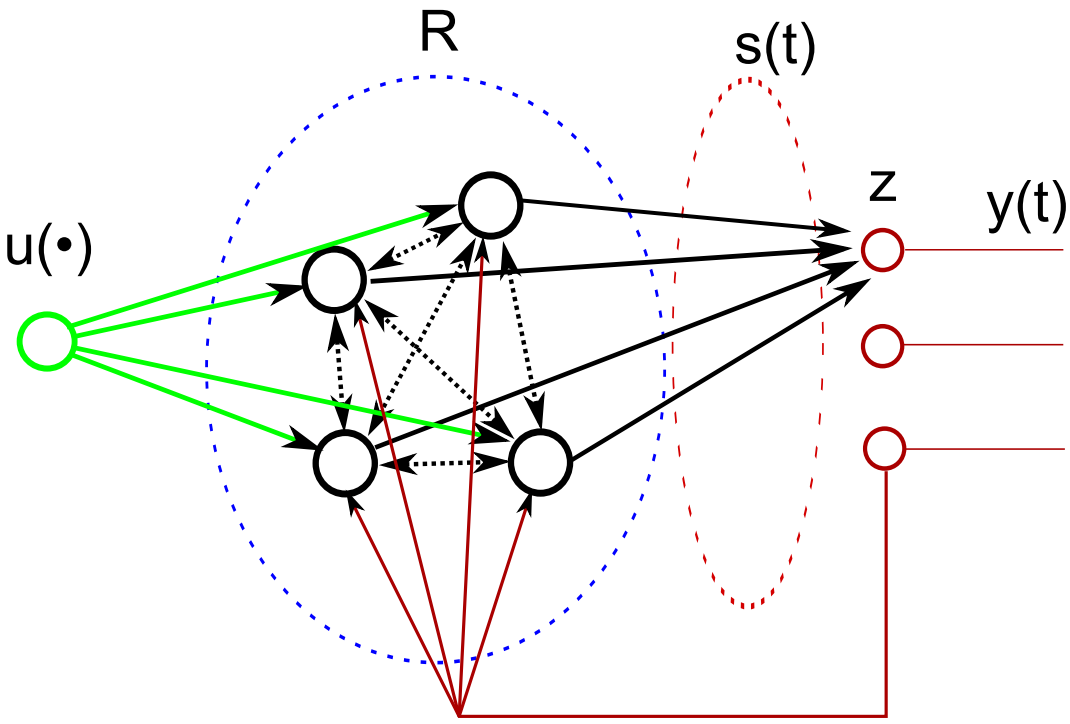

Figure 2.2: Liquid computing model like in Figure 2.1 but with a feedback loop. In addition to the input stream $u(\bullet)$ the reservoir R is provided with the output $y(t)$. Every internal neuron of R might be connected with the output from every readout unit in z to create complex feedback dynamics.

2.1.2 Echo State Network

Another approach for computing with reservoirs is the Echo State Network (ESN) proposed by [Jäger, 2001]. The model basically consists of a discrete-time recurrent neural network with K input units u, N internal network units s and L output units y. Looking back at the liquid computing model in Fig. 2.1 the internal units together can be thought of the reservoir R and the output units as the readout z. The term "Echo State" refers to the internal state $s[n]$ being viewed as an echo of previous inputs.

$$\mathbf{s}[n] = E\left[n, n-1, n-2, ...\right] \tag{2.4}$$

with the discrete function E being equivalent to the continuous liquid filter $\mathcal{F}$ introduced earlier. The update of each internal unit s_j is computed according to

$$s_j[n+1] = f(\sum_k w^{\text{in}}_{j,k} u_k[n+1] + \sum_i w_{j,i} s_i[n] + \sum_l w^{\text{back}}_{j,l} y_l[n])) \tag{2.5}$$

where f is the output function of the unit which is typically sigmoid. $\mathbf{W}$ here is the weight matrix for connections between the internal network units, $\mathbf{W}^{in}$ a matrix collecting the input weights and $\mathbf{W}^{back}$ is the matrix containing weights for feedback connections from the output units. The output is given by

$$\mathbf{y}[n+1] = f^{\text{out}}\left(\langle \mathbf{w_j}^{out}, \mathbf{x}[n+1] \rangle\right) \tag{2.6}$$

with $\mathbf{x}[n]$ being a concatenation of input, internal state and output vectors:

$$\mathbf{x}[n] = \begin{pmatrix} \mathbf{u}[n] \\ \mathbf{s}[n] \\ \mathbf{y}[n-1] \end{pmatrix} \tag{2.7}$$

f^{out} describes the output function of the output units and $\mathbf{W}^{\text{out}}$ is a matrix containing all weights that lead to the output unit. The vector $\mathbf{x}$ is used in the inner product $\langle \mathbf{w}^{out}_j, \mathbf{x}[n+1] \rangle$. Note that with this definition even output units may be recurrently connected as well as there being direct connections from input to output units. For the network to be able to being used in a reservoir computing sense it is required to have the *echo state property* defined and examined extensively by [Jäger, 2001]. Furthermore a range of corresponding properties are defined which are *uniformly state contracting*, *state forgetting* and *input forgetting*. Possessing any of these properties leads to the network being able to have echo states. From these properties it can be intuitively stated that a reservoir possesses the echo state property if it is able to wash out any information pertaining to initial condition asymptotically.

2.2 Reward modulated learning

In modelling the behaviour of animals and humans, the concept of reward and reward based learning methods play a crucial role. Conditioning experiments rely on pairing a possible action of the subject to a specific reward or punishment. In Psychology this is known as Thorndike's Law of Effect [Thorndike, 1911]. The concept of reward is also used in machine learning and economic decision making [Sutton and Barto, 1998]. Rewards are typically objects or states which attain positive or negative value by some kind of evaluating system or process. These can be used to increase the likelihood of behaviour leading to the same or similar reward inducing states. Usually this takes the form of some kind of learning procedure, where multiple trials of the same or similar problem are faced with varying behaviour. A reward signal then acts as a kind of teacher influence in determining the most favourable behaviour over the course of the trials. The reward can be induced manually, like giving out money by an experimenter for each question correctly answered. It can also come naturally like the sensation of a full stomach after having eaten.

There are also fundamental problems regarding using such reward signals to facilitate learning. One is known as the "credit assignment problem". With more complex tasks and behavioural variety, it becomes unclear which part of a specific behavioural pattern triggers a reward response. Especially in a natural environment, not every object or movement is relevant for solving a problem, and therefore subject to adjustment for reward attainment. Another related problem evolves around temporal delay. Typically there is a time window between the required behaviour and the reward response. The length of this time period and the nature of the task in producing distracting elements may strongly interfere with the ability to map a specific behaviour to the corresponding reward response.

In some reward based learning methods the information about the learning process is obtained in a reward prediction error signal. This error signal naturally requires a prediction to be made about occurring reward impulses and thus an estimate of the state of the environment. As any action taken

will likely result in a change in the environment, these too have to be estimated and accounted for. The exploration and selection of effective actions is therefore critical to learning success. The resulting difference between prediction of the evolved environmental state and the obtained reward can then be used as performance measure to make future predictions more accurate with respect to the environmental data on which the prediction was based. Research suggests that this form of learning behaviour based on predicting reward stimuli takes place in the human brain (see [Schultz, 2007] for a review). Various neuromodulators influence synaptic plasticity. Especially the function of dopaminergic neurons is often likened to generating reward signals for the human brain [Schultz, 2007]. Dopaminergic neurons located in the mid-brain not only seem to activate in bursts after food and liquid rewards, but also encode a prediction error in the form of:

$$\text{Dopamine Response} = \text{reward occurred} - \text{reward predicted} \tag{2.8}$$

Other research about synaptic plasticity has postulated Spike Timing Dependent Plasticity (STDP) as a plausible model about how single presynaptic and postsynaptic neurons influence each other [Bi and Poo, 1998, Markram et al., 1998]. STDP models a spiking-neuron specific form of Hebbian Learning [Hebb, 1949], where the fundamental principle is the temporal correlation of neuronal firing times. Specifically the weight of a synapse between 2 neurons changes according to the timing difference of presynaptic spike arrivals and postsynaptic spikes. Although it is difficult to show this theoretical concept in practical studies there are signs that suggest involvement of dopamine signals in synaptic plasticity [Pawlak et al., 2010]. Because as discussed above dopamine is a strong candidate for supplying some form of reward signal, this leads to the consideration of a global reward signal as a gating or modulation mechanism for synaptic plasticity rules (see e.g. [Legenstein et al., 2008] for models). This is often called reward-modulated Hebbian Learning. The corresponding learning rules can be called 3 factor learning rules after their dependence on pre- and postsynaptic activity $x(t)$

and $y(t)$ as well as modulation $M(t)$ (e.g. by a reward signal):

$$\Delta w(t) = \eta x(t) y(t) M(t) \tag{2.9}$$

In the following experiments we will use one of these rules, the Exploratory Hebb Rule (EH) from [Legenstein et al., 2010b]. The 3 factors here are the output of the readout neurons, an exploratory noise signal $\xi(t)$ and the modulatory reward signal $M(t)$. The output of the readout neurons is perturbed by the exploration signal. The exploration noise establishes a search space around the input, which is considered to be changing much more slowly so it can be considered constant (in contrast to STDP). The resulting correlation between $\xi(t)$ and $M(t)$ provides a performance measure on which the learning of the readout weights can be based.

2.3 Reward based learning of readouts

For a reservoir in order to successfully compute its task, suitable connection weights have to be found. We have established previously that the connections of the reservoir units themselves can be randomly generated. The question is now how to obtain readout weights suitable for a given task. Along with the definition of echo state networks, [Jäger, 2001] provided a training method consisting of 3 steps.
In Step 1 the reservoir is constructed and all connections are established randomly.
During Step 2 the network is driven by input of a training set. The training set consists of a input sequence $\mathbf{u}[1]$, $\mathbf{u}[2]$,... ,$\mathbf{u}[N]$ and corresponding target outputs $\mathbf{y}^*[1]$,$\mathbf{y}^*[2]$,...,$\mathbf{y}^*[n]$. If there are feedback connections from the output back into the reservoir, the target outputs have to be injected back into the reservoir, called "teacher forcing". During this training the resulting network states $\mathbf{s}[1]$, $\mathbf{s}[2]$,...,$\mathbf{s}[N]$ are collected. Lastly in Step 3 the optimal output weights are computed by finding weights which minimize the error between output gained from the collected states $\mathbf{s}[n]$ and the desired output $\mathbf{y}^*[n]$. Due to the simple nature of the readout this can be done with using

linear regression.
For this method it is necessary to acquire and provide a suitable training set. For some problems this might become cumbersome or outright impossible. Also while in the context of artificial neural networks this type of supervised learning poses no problem, in the context of biological systems and neuroscience more plausible methods of obtaining readout weights might be desirable. Thus we try to apply reward based learning. In a reward based approach, only a reward signal and a final target state have to be chosen. Thus there is no teacher-provided data on optimal output the network should give, and no solution to reach the target has to be known beforehand.
The amount of supervision is determined by the complexity and information the modulating reward signal $M(t)$ provides. As an example consider a 2-dimensional plane on which an agent moves towards a target point. A reward signal for the agent could consist of only the Euclidean distance between its momentary position. Alternatively it could incorporate the angle between its movement vector and the direction to the target, providing the agent with much more spatial information.
With the EH Rule it can be shown [Hoerzer et al., 2011] that learning is possible even with the most basic signal. In contrast to e.g. [Sussillo and Abbott, 2009] who use similar network models and experiments with fully supervised learning rules, [Hoerzer et al., 2011] use only following reward

$$M(t) = \begin{cases} 1 & \text{if } P(t) > \bar{P}(t) \\ 0 & \text{if } P(t) \leq \bar{P}(t) \end{cases} \tag{2.10}$$

with $P(t)$ as a measure of system performance as follows

$$P(t) = -\sum_{i=1}^{L} \left(z_\xi^i(t) - f_i(t) \right)^2 \tag{2.11}$$

and $\bar{P}$ a low-pass filtered moving average of $P(t)$. $z_\xi^i(t)$ is the output of neuron i and $f_i(t)$ the target value for $z_\xi^i(t)$. The target values themselves are not known to the network. The only information accessible for the learning rule is whether the overall output of all readout neurons was an improvement over

the recent performance or not. This is arguably the least possible information under which goal directed learning is still possible. However the target still consists of a complete trajectory with every point in time being assigned an optimal value. During large stages of the simulation it requires only small corrections accounting for small changes of the target, assuming a grade of continuity. In our experiments we will use similar learning and reward procedures, but with the target being a distant end state which the model must reach. In a 2 or more dimensional space this target state is reachable by many trajectories which the readout has to construct on its own. A reward that is based only on the achievement of a relatively long-fetched goal is a further step of removing teacher influence on the learning process. In the following chapters we want to demonstrate how this might be achieved.

Chapter 3

Model

In this chapter we will present a basic model on which our simulations are based. The model consists of 3 major components, as shown in Figure 3.1. The first component is a reservoir of recurrently connected neurons with a single layer of linear readout units. As discussed in the previous chapter its purpose is to collect information about the dynamics of the second component, a model of a moving physical mass, inside its state space. The readout units then generate a control signal which drives the physical object, generating movement. The information provided by the object is its own position state, which is fed back into the reservoir. The third component generates temporally correlated noise and adds it to the readout signal from the reservoir. This noise is essential for the learning process.
As already stated, the task of every simulation is for the reservoir to lead the equation system to a specific target state. In our simulations the target state always represents a specific position for the modelled physical object and no demands are made for specific velocity values. The desired control signal that drives the model on a trajectory leading from the initial position to that target is unknown throughout the simulation. It must be found by the model by moving around and receiving a reward signal. This reward changes proportionally with the distance of the object to the target, essentially telling the model how far away it is from its goal.

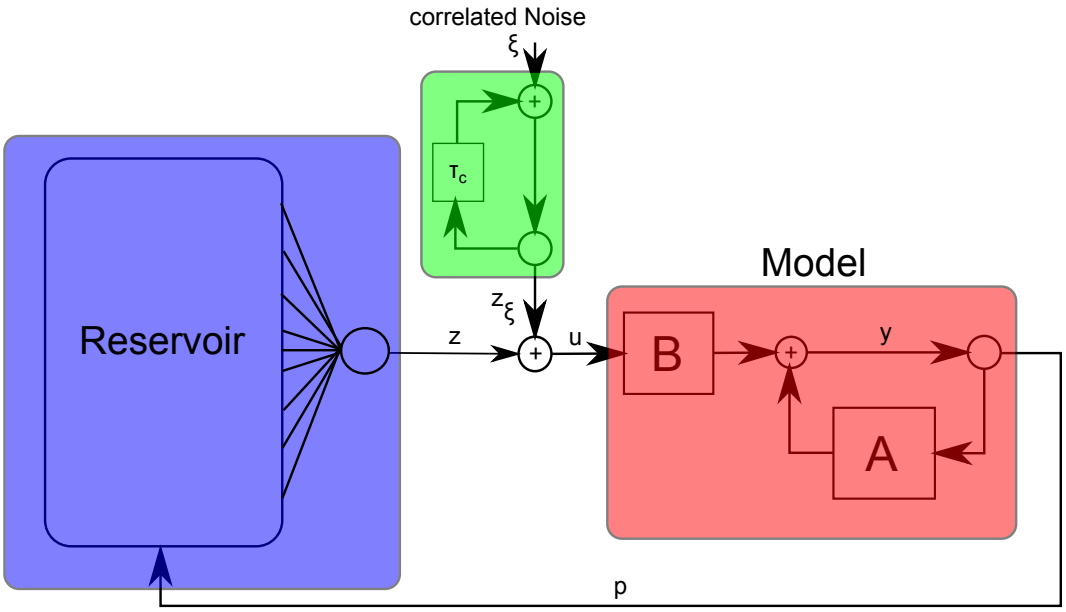

Figure 3.1: The System diagram consists of three main components. (blue) The reservoir of non-linear functions stores information about the model dynamics which it gains from the feedback signal $p(t)$. Its readout units create the output signal $\mathbf{z}(t)$ to control the model. (green) The recurrent loop constructs the time-correlated noise signal $\mathbf{z}_\xi$ from the uncorrelated source noise ξ and the noise correlation time constant τ_c. Together with the output of the reservoir readout $\mathbf{z}(t)$,z_ξ forms the control signal u. (red): The physical model is driven by the noisy reservoir output $\mathbf{u}(t)$. Generally it consists of a system of differential equations modelling a device we want to control like an electric motor or a moving physical object. The position of the model is returned to the reservoir as feedback signal $p(t)$.

Typically the control signal is interpreted as acceleration value, manipulating position through changing the velocity of the physical model. Thus the model can be viewed as integrating the control signal before feeding it back to the reservoir. As the movement of the modelled mass is subject to inertia and friction, the velocity can be considered to have a low-pass characteristic. The goal of these simulations is to demonstrate the ability of time correlated noise to overcome the difficulties of low-pass filtering and integration of the

control signal. It must be able to effect changes in the position of the object so the reward signal can give meaningful information about the target and inject it into the weights of the readouts. With these weight changes the reservoir has to create a control signal strong enough to overcome the random travelling of the noise to guide the model to the target. This also places constraints on the noise amplitude.
Throughout the rest of the chapter we want to provide a more precise mathematical definition of the model. All definitions here are stated in continuous time as we still perceive the model in a biological context. The simulation results presented in the next chapter are obtained using discretized models. The equations for these discrete models can be found in Appendix A.

3.1 Reservoir model

The reservoir is implemented as an array of fully connected neurons. The internal dynamics is given by

$$\tau \dot{x}_i(t) = -x_i(t) + \lambda \sum_{j=1}^{N} w_{ij}^{\text{rec}} r_j(t) + \sum_{j=1}^{M} w_{ij}^{\text{fb}} y_j(t) \tag{3.1}$$

where τ is the membrane constant. The state $x_j(t)$ of the j-th neuron represents its membrane potential at the soma at time t. Parameters w_{ij}^{rec} and w_{ij}^{fb} denote the weights for recurrent connections within the reservoir and feedback connections from the model to the network neurons respectively. Their values were randomly chosen to be between -1 and 1. The firing rate of the j-th reservoir neuron r_j at time t is given by

$$r_j(t) = \tanh(x_j(t)) + \xi_j^{\text{state}}(t), \tag{3.2}$$

where $\xi_j^{\text{state}}(t)$ models zero mean noise on the firing rate of the neuron. This noise is uniformly distributed between -0.05 and 0.05. As further on we will use multiple readout neurons, the following formulas are used for a generic number of readouts. The signal from the readout neurons $\mathbf{z}(t)$ is computed

by a simple sum of the weighted firing rates of all reservoir neurons r_j at time t.

$$z_i(t) = \sum_{j=1}^{N} w_{ij}^o r_j(t) \tag{3.3}$$

w_{ij}^o are the readout weights for the i-th output unit from the j-th reservoir unit. The control signal $\mathbf{u}(t)$ is obtained by adding correlated noise $z_\xi(t)$ to $\mathbf{z}(t)$:

$$\mathbf{u}(t) = \mathbf{z}(t) + \mathbf{z}_\xi(t) \tag{3.4}$$

The exploration noise $z_\xi(t)$ is detailed further below in Equation 3.8. The readout weights w_{ij}^o were initialized to very small values drawn from a uniform distribution with zero mean, and then adapted throughout training with a variation of the Exploratory Hebb (EH) rule [Legenstein et al., 2010a]. The distribution was ranged between -0.01 and 0.01.

$$\Delta w_{o,ij}(t) = \eta \, (R(t) - \bar{R})(t) \, (u_i(t) - \bar{u}_i(t)) r_j(t) \tag{3.5}$$

where R is a reward function for measuring momentary system performance. The term $u_i - \bar{u}_i$ tries to reconstruct the correlated noise z_ξ from the control signal u. The readout weights are thus changed according to the momentary change of the reward function $R - \bar{R}$ that results from the momentary change of the control signal by the exploration noise. η denotes the learning rate. The two running average values $\bar{R}(t)$ and $\bar{u}(t)$ are computed by

$$\tau_z \frac{d}{dt} \bar{u}(t) = (u(t) - \bar{u}(t)) \tag{3.6}$$

for the control signal $u(t)$ and

$$\tau_\alpha \frac{d}{dt} \bar{R}(t) = (R(t) - \bar{R}(t)) \tag{3.7}$$

for the reward signal $R(t)$. τ_α and the readout output filter time constant τ_z are suitably chosen constant values.

3.2 Exploration noise

The generator of the exploratory noise is the second component. Because of the low-pass characteristic of the inertia model,temporally uncorrelated noise is expected to be ineffective for usage with the learning rule. Therefore temporally correlated exploration noise $\mathbf{z}_\xi$ is introduced into the control signal.

$$\frac{d}{dt}\mathbf{z}_\xi(t) = -\frac{z_\xi(t)}{\tau_c} + \xi \tag{3.8}$$

where ξ is a vector of uniformly distributed random variables with zero mean. The range was set between $-a$ and a, with a referred to as the noise amplitude. The value of a is dependent on the experiment. $\mathbf{z}_\xi(t)$ is of the same dimension as the control signal $\mathbf{u}(t)$ so the noise signals for each dimension of the model, while temporally correlated, are independent of each other. The noise correlation time constant τ_c governs the time over which the noise remains correlated. It has significant influence on the effect of the noise on the system and on training success.

3.3 Physical model

Throughout the simulation 2 different architectures are used. The first model represents a moving object with a position in an environment and a velocity. All its mass which is set to 1 kg is concentrated in a single point. The influence of inertia gives it a low-pass filter characteristic for the control signal $z(t)$. Its dynamics are based on a set of differential equations

$$\frac{d}{dt}\mathbf{y}(t) = A\mathbf{y}(t) + B\mathbf{u}(t) \tag{3.9}$$

where the state vector $\mathbf{y}$ models both position $\mathbf{p}$ and velocity $\mathbf{v}$.

$$\mathbf{y}(t) = \begin{pmatrix} \mathbf{p}(t) \\ \mathbf{v}(t) \end{pmatrix} \tag{3.10}$$

At first we use this model in one-dimensional space to show the difference between correlated and uncorrelated noise. In a second experiment we add a second spatial dimension. On a single axis, a reward signal based on direction always points straight at the target or away from it. A point mass moving on a plane has much more margin of error as it can move towards the target on a trajectory which does not actually lead to the target. As such at least 2 dimensions are needed here to accurately test the performance of the architecture.
After this 2 simulations we use the model of a robot arm with 2 joints to simulate a physical model with more complex internal dynamics (shown in Figure 3.2). The robot arm consists of 2 joints with motors inside the end points. Here the control signal from the reservoir acts as torque for the 2 motors inside the joints. The goal is for the endpoint of the arm to reach a target point on a 2 dimensional plane. Although the dimensionality is the same as in the previous simulation, the influence of the two mass points of the joints on each other require internal states of much more complexity. For the rest of this chapter we will present the internal dynamics of the physical models in the three simulations in detail.

3.3.1 1D Model

In the one-dimensional model the reservoir has to control the velocity of a mass point on a infinite line so that the position assumes a specific target value on the line. The dynamics parameter matrices A and B are given as

$$A = \begin{pmatrix} 0 & 1 \\ 0 & \tau_f \end{pmatrix}, \; B = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{3.11}$$

with τ_f being the friction coefficient. The reward function was chosen as a simple measure of distance between the target and the momentary position of the model. Thus it looks as follows

$$R(t) = - \left| p(t) - p_{\text{target}} \right| \tag{3.12}$$

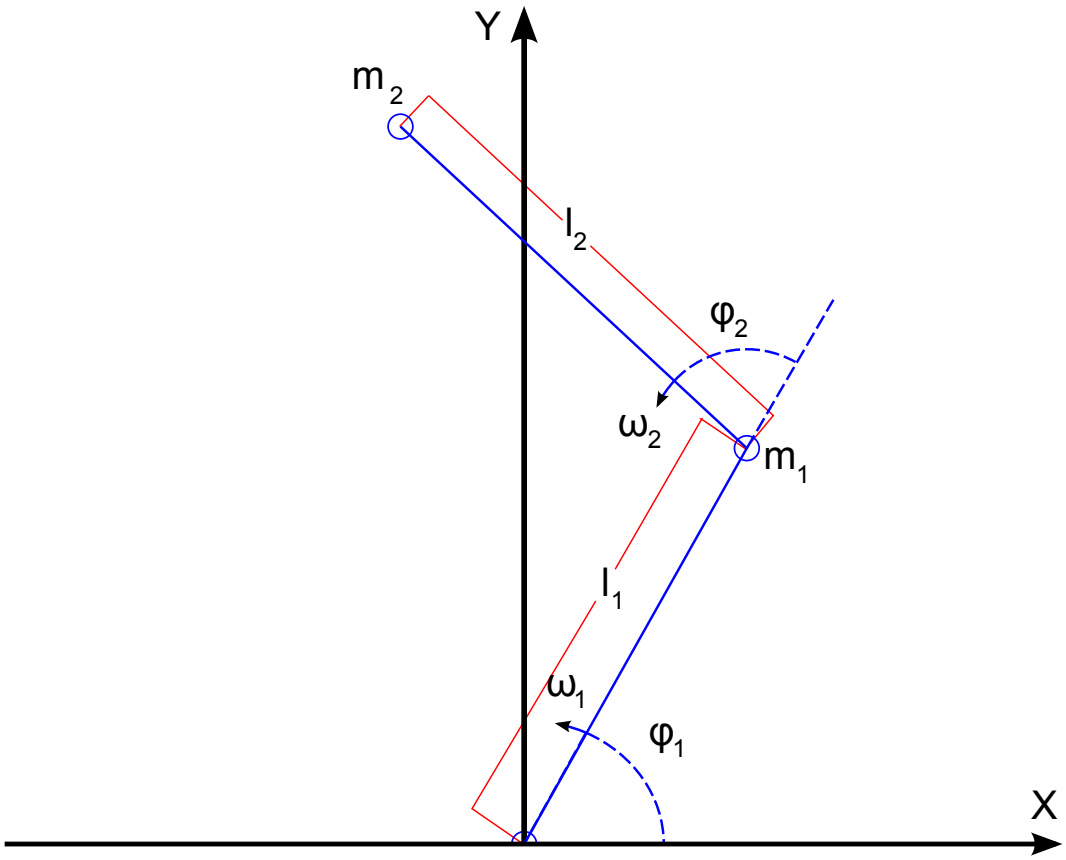

Figure 3.2: Model of a 2-link robotic arm. The mass is concentrated in 2 points for simplicity. The points for this masses m_1, m_2 are located in the 2 joints of the links. Movement is controlled by motors in the joints at m_1 and m_2. They act on the acceleration $\delta\omega_1, \delta\omega_2$ on the two angles ϕ_1, ϕ_2.

where p_{target} was the target position value of the model. The function is always negative with its highest value of zero reached when the model has reached the target. This is to ensure that the momentary change in the reward signal given by $R(t) - \bar{R}(t)$ is always positive when the model moves towards the target and the distance lessens, and negative otherwise. The weights are then updated according to this momentary change as shown in Equation 3.5. The feedback signal back into the reservoir consists of the model position $p(t)$ and was connected randomly to all reservoir neurons.

3.3.2 2D Model

The system was then expanded by a second dimension with separate noise and input. It still represents a mass point where the velocity is controlled by the reservoir to reach a given target point on a plane. The control signal for each dimension is held separately, so $u_1(t)$ controls $v_1(t)$ and $u_2(t)$ controls $v_2(t)$. Two separate readout units with distinct weights are used to construct $\mathbf{z}(t)$. A and B are given as

$$A = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\tau_f & 0 \\ 0 & 0 & 0 & -\tau_f \end{pmatrix}, \; B = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \tag{3.13}$$

while the reward function R stays the same as in Equation 3.12 but with 2-dimensional points $\mathbf{p(t)}$ and $\mathbf{p}_{\text{target}}$.The weights for both readouts are updated using this global R. Again as feedback into the reservoir both dimensions of $\mathbf{p(t)}$ were used as separate signals with own connections and weights.

3.3.3 Model of a robot arm

After exploring the boundaries of the system parameters with the two-dimensional model a last simulation with a completely different model architecture was made. This time the simulation of a 2-joint robot arm was chosen for its realistic dynamics of different masses influencing each other establishing a bigger state space. Also guiding a robot arm is a straightforward application and it's use is easily understood. Thus the last simulation task was to try to guide this robot arm to a target endpoint.
The arm is modelled like a double inverted pendulum with 2 links joined together at one end as shown in Figure 3.3. The first link is joined to the ground. The end of the second link acts as the reference point for reaching the goal. The links 1 and 2 have lengths $l_1 = l_2 = 1m$ and masses $m_1 = m_2 = 1kg$. For simplicity m_1 and m_2 are considered to be concen-

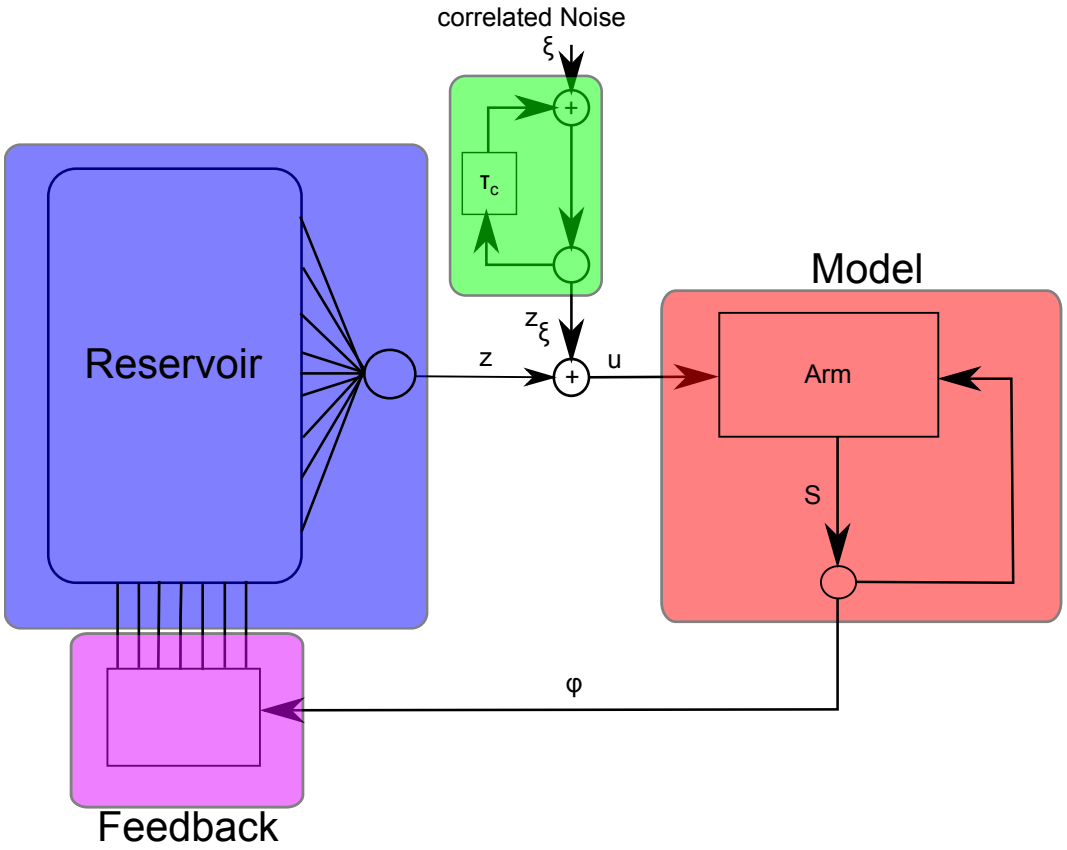

Figure 3.3: The system was modified from Figure 3.1. (violet) A feedback component was added that sparsely encodes the 2 dimensional feedback signal into 20 dimensions via radial basis functions. (red) The controlled system was changed to the simulation of a 2-link robotic arm with a 4 dimensional internal state S(See Equation 3.14).

trated in the joints. The two motors in the joints of the arm create a torque for the angular movement of the 2 links of the pendulum to each other and to the ground. The robotic arm was modelled with 4 internal variables, the 2 angles $\phi_1(t), \phi_2(t)$ of the joints and the angular velocities $\omega_1(t), \omega_2(t)$. For simplicity the values for both angles were restricted to $\phi_{min} = 0$ and $\phi_{max} = 1\pi$ without taking into account link width or physical joint limits. Figure 3.2 shows how they relate to the model. Together they form the

internal state $\mathbf{s}$ of the model defined as follows:

$$\mathbf{s}(t) = \begin{pmatrix} \phi_1(t) \\ \omega_1(t) \\ \phi_2(t) \\ \omega_2(t) \end{pmatrix} \tag{3.14}$$

The activation function of the readout neurons was also changed because of the sensibility of the model. Because of the radial nature of the problem space continuously strong control signals could cause the arm to strongly push against the maximum angles. Therefore a scaling factor was suitably chosen to adjust the reservoir output strength. Also the maximal torque that the 2 motors generate is limited to $F_{\max} = 20Nm$. To avoid saturation effects that could severely affect the function of the exploration noise z_ξ this maximum torque was accounted for in a limiting function applied to the control signal $\mathbf{u}(\mathbf{t})$. Thus a tangens hyperbolicus function was chosen to limit the signal while preserving continuity of the function.

$$u_i(t) = \tanh\left(\beta \sum_j w_{i,j} x_j(t) + z_\xi(t)\right) F_{max} \tag{3.15}$$

β denotes a scaling constant which was empirically chosen to keep the reservoir output from saturating the tanh function. The signal u from the reservoir acts as control signal for the motor torques, thus generating the angular acceleration $\dot{\omega}_1, \dot{\omega}_2$. The Formula for the acceleration can be derived from the kinematics of the robot arm. The points p_1, p_2 denote the x and y coordinates

of the 2 masses m_1, m_2.

$$\begin{aligned}
\mathbf{p_1} &= \begin{pmatrix} l_1 \cos\phi_1 \\ l_1 \sin\phi_1 \end{pmatrix} \\
\mathbf{p_2} &= \mathbf{p_1} + \begin{pmatrix} l_2 \cos(\phi_1 + \phi_2) \\ l_2 \sin(\phi_1 + \phi_2) \end{pmatrix} \\
\dot{\mathbf{p_1}} &= \begin{pmatrix} -l_1\dot{\omega_1} \sin\phi_1 \\ l_1\dot{\omega_1} \cos\phi_1 \end{pmatrix} \\
\dot{\mathbf{p_2}} &= \dot{\mathbf{p_1}} + \begin{pmatrix} -l_2(\omega_1 + \omega_2)\sin(\phi_1 + \phi_2) \\ l_2(\omega_1 + \omega_2)\cos(\phi_1 + \phi_2) \end{pmatrix}
\end{aligned} \tag{3.16}$$

For simplicity of the simulation the gravitational force was omitted. Thus the kinetic energy T and the potential energy U resolve to

$$\begin{aligned}
T = &\frac{1}{2}\left(m_1 l_1^2 + m_2 l_1^2 + m_1 l^2 + m_2 l_1 l_2 \cos(\phi_2)\right)\omega_1^2 + \\
&\frac{1}{2} m_2 l_2^2 \omega_2^2 + \left(m_2 l_2^2 + \frac{1}{2} m_2 l_1 l_2 \cos\phi_2 \omega_1 \omega_2\right) \\
U = &\, 0
\end{aligned} \tag{3.17}$$

Using the method of Lagrange we arrive at the following equation for the acceleration $\dot{\omega}(t)$ from the input signal $\mathbf{u}$.

$$\dot{\omega}(t) = H^{-1}(t)\left(\mathbf{u} - C\omega\right)(t) - f\omega \tag{3.18}$$

with

$$\begin{aligned}
H(t) &= \begin{pmatrix} \frac{1}{3}l_1^2 m_1 + \frac{1}{3}l_2^2 m_2 + m_2 l_1^2 + \frac{1}{2}l_1 l_2 m_2 cos(\phi_2(t)) & \frac{1}{3}l_2^2 m_2 + \frac{1}{2}l_1 l_2 m_2 cos(\phi_2(t)) \\ \frac{1}{3}l_2^2 m_2 + \frac{1}{2}l_1 l_2 m_2 & \frac{1}{3}l_2^2 m_2 \end{pmatrix} \\
C(t) &= \begin{pmatrix} -l_1 l_2 \sin(\phi_2(t))\,\dot{\omega_2}(t) & -\frac{1}{2}l_1 l_2 \sin(\phi_2(t))\,\dot{\omega_2}(t) \\ \frac{1}{2}l_1 l_2 \sin(\phi_2(t))\,\dot{\omega_1}(t) & 0 \end{pmatrix}
\end{aligned} \tag{3.19}$$

A generic friction term f was introduced in Equation 3.18, acting on the current acceleration in each time step. This limitation of acceleration should act as discouragement of high velocities like the friction terms in Equations

3.11 and 3.13. Without it the joint velocities would quickly outgrow the influence of the exploration noise z_ξ on the direction of movement. The target was a circle with a radius of $0.1m$ around a target point defined by ϕ_{target}. The reward function was chosen to be the normalized product

$$R(t) = \frac{\mathbf{v}^*(t)\,\mathbf{v}(t)}{|\mathbf{v}^*(t)|\,|\mathbf{v}(t)|} \tag{3.20}$$

of current speed $\mathbf{v}(t)$ and the distance to the target $\mathbf{v}^*(t)$. Both had to be transformed into cartesian vectors from the angles $\phi_1(t), \phi_2(t)$ and angular velocities $\omega_1(t), \omega_2(t)$.

$$\mathbf{v}^*(t) = \phi_{\text{target}} - l_1 \begin{pmatrix} cos(\phi_1)(t) \\ sin(\phi_1)(t) \end{pmatrix} + l_2 \begin{pmatrix} cos(\phi_1(t) + \phi_2(t) \\ sin(\phi_1(t) + \phi_2(t) \end{pmatrix} \tag{3.21}$$

$$\mathbf{v}(t) = \omega_1(t) \begin{pmatrix} cos(\phi_1(t) + \frac{\pi}{2}) \\ sin(\phi_1(t) + \frac{\pi}{2}) \end{pmatrix} + \omega_2(t) \begin{pmatrix} cos(\phi_1(t) + \phi_2(t + \frac{\pi}{2}) \\ sin(\phi_1(t) + \phi_2(t) + \frac{\pi}{2}) \end{pmatrix} \tag{3.22}$$

The weight update also changed from Equation 3.5 to

$$\Delta w_{o,ij}(t) = \eta\, R^*(t)\, (z^i_\xi(t)) r_j(t) \tag{3.23}$$

where $R^*(t)$ was determined by

$$R^*(t) = \begin{cases} 1 & \text{if } R(t) \geq \bar{R}(t) \\ -1 & \text{if } R(t) < \bar{R}(t) \end{cases} \tag{3.24}$$

which is similar to the reward used by [Hoerzer et al., 2011] shown in Equation 2.11. There is an additional reason for using $R^*(t)$ rather than $R(t)$. With shrinking distance between the end effector of the arm and the target also $R(t)$ may diminish by several magnitudes. This has a scaling effect on the weight change which can, in concert with low values of η lead to no significant weight updates at all if the arm is already near the target. At every time step the joint angles $\phi(t)$ where fed back into the reservoir. Because of the sensibility of the system with regard to the control signal one has to

be careful with feedback signal shaping. In this model we used radial basis functions to sparsely encode the feedback signal. Each of the 2 feedback signals were fed to 10 function kernels. The centres of the kernels μ_i were uniformly distributed between 0 and 2π to match the restriction of the joint angles. The following kernel function was used

$$\mathbf{o_i}(\mathbf{t}) = \exp^{\frac{(\phi(t)-\mu_i)^2}{2\rho^2}} \tag{3.25}$$

and the standard deviation ρ was set to 0.5. The resulting feature vectors $\mathbf{o}(t)$ were then fed back to the reservoir neurons $\mathbf{x}(t)$ by the weight matrix W^{fb} as in Equation 3.1. Like before the weights had values randomly assigned between -1 and 1, but this time only 25% of the weights were allowed to have values different from zero. The distribution of these non zero values was also random.

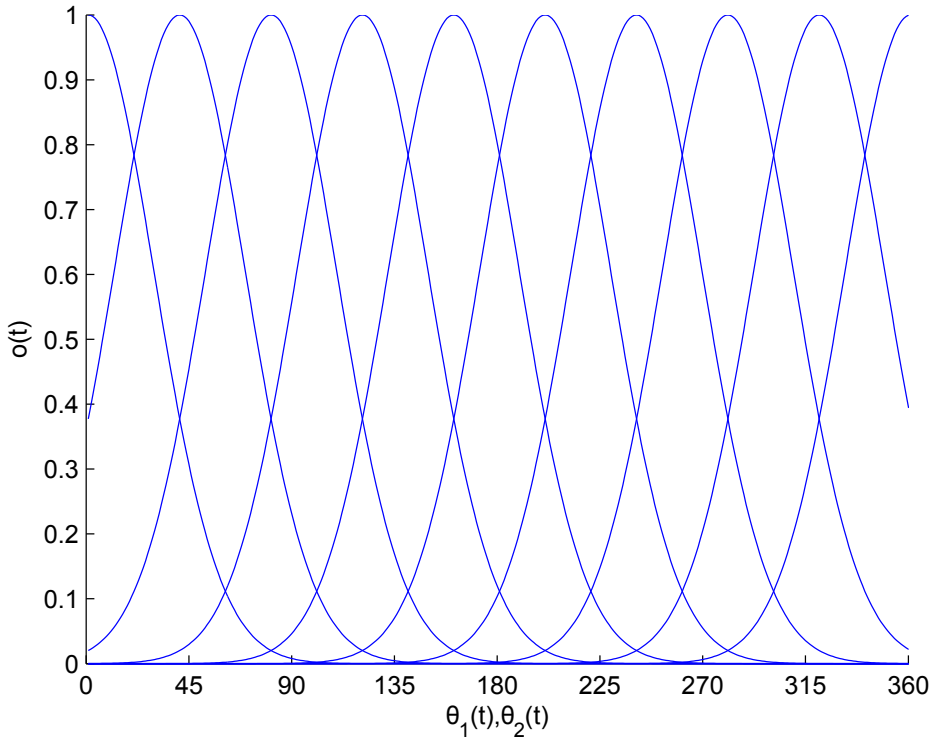

Figure 3.4: Radial basis function kernels for the Feedback signal $\mathbf{o}(\mathbf{t})$. As the angles of the two joints were constrained the 10 kernels that were used are evenly spread out over the interval 0 to 2π radians or 0° to 360° as shown in this figure.

Chapter 4

Results

4.1 Simulation results for 1D model

4.1.1 Correlation of exploration noise

The simulation of the first model, the 1-dimensional mass, was divided into two parts. In the first part the readout output z was set to zero. To show the difference between time-correlated and uncorrelated noise, the system position $p(t)$ and velocity $v(t)$ were recorded over a time of 100 seconds and then compared. The system state $\mathbf{s}(\mathbf{t})$ was recorded at first without correlated noise, then with noise correlation time constant τ_c set to 5 s. The amplitude of the uncorrelated source noise ξ was fixed differently so that the noise signal z_ξ would have the same variance both with and without time-correlation as seen in Figure 4.1. For the simulation time step $\Delta t = 1$ms the amplitudes were fixed at 0.5 for the correlated signal and 24 for the uncorrelated one. As seen in Figure 4.1 the width of the distribution of $v(t)$ doubles for correlated input noise. The standard deviations behave in the same way. The more significant difference lies in the position values $p(t)$ as seen in Figure 4.2. The non-zero mean value of local velocity spikes causes the position to vary significantly. In the absence of other input only the correlated noise generates the movement necessary for the reward function

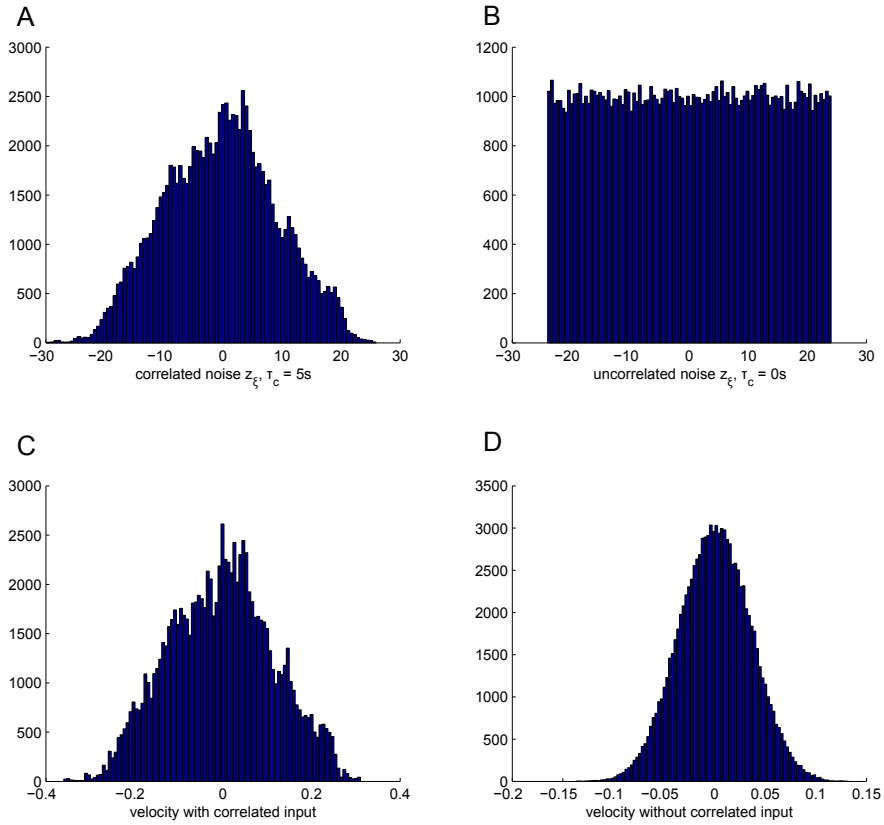

Figure 4.1: Histograms of noise and velocity amplitudes with and without time correlation. A and B show the amplitude distribution of noise that is either temporally correlated with a τ_c of 0.5s in A or completely uncorrelated in B over the course of 100s (or 100000 timesteps). The amplitude for the noise source was set 48 times higher in case of the uncorrelated noise to reach the same maximum amplitudes. In C and D the respective velocity distributions for the 1D model with correlated (in C) and uncorrelated noise (D) as input is shown. The maximum velocity driven by correlated noise more than doubles, and the distribution loses its symmetry which allows the possibility of long term movement of a low-pass filtered model and thus exploration.

to return meaningful values to update the readout weights with respect to the position change generated by the noise. If we take a look back at the weight update function in Equation 3.5 this reflects the desired behaviour on the condition that the input from the reservoir changes so slow as to be nearly constant in nature.

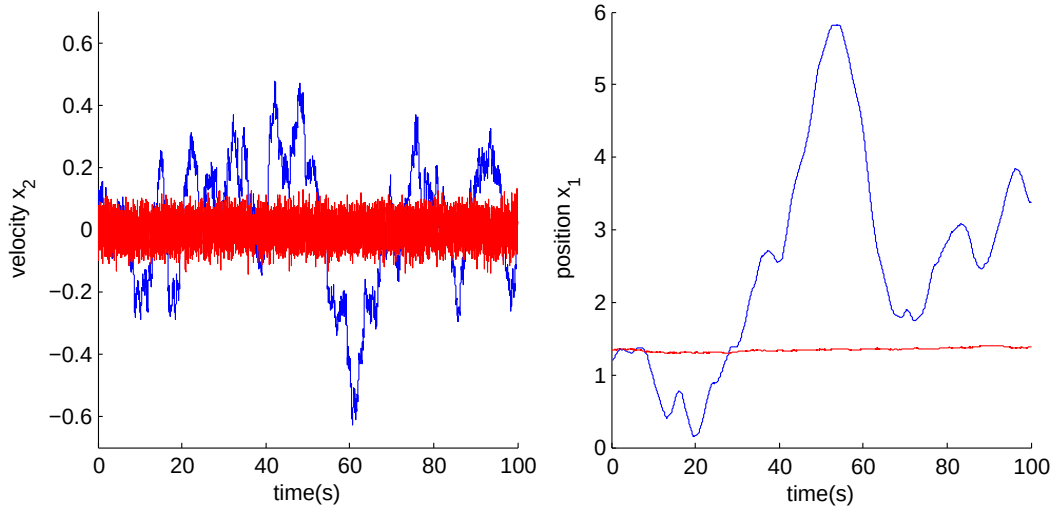

Figure 4.2: Recorded output shows the advantage of using time correlated noise. Left: Velocity recorded over 100 seconds. Right: Position values during the same time. Without using temporally correlated noise (red) position of the model remains constant. Using temporally correlated noise yields not only higher velocities, but also stable velocity values over enough time to change the position significantly over the course of the simulation.

4.1.2 System Feedback

With these values for correlation time and noise amplitude the full system was simulated to show the impact of noise correlation on learning performance. Also this simulation should prove that the reservoir was capable of producing a readout signal smooth enough to satisfy the "nearly constant" condition in the time frame of the noise correlation time constant. The averaging time

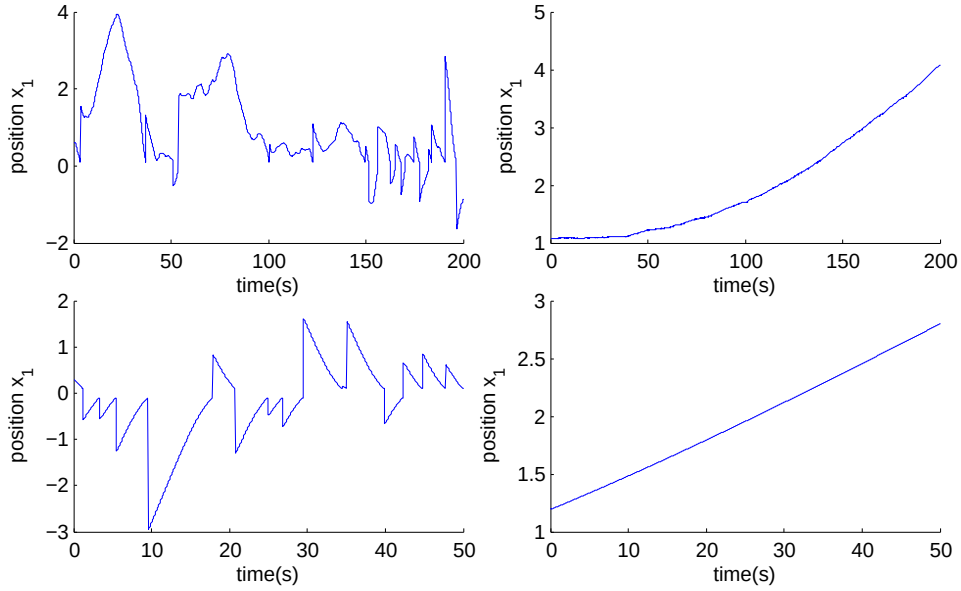

Figure 4.3: Simulation of position (p_1) over time. Top: Training phase for the model. Bottom: Testing phase. Left: Simulation with temporally correlated noise z_ξ. Right: noise correlation time constant τ_c was set to 0s. On the Left side the testing phase shows much progress during the whole period resulting in very stable performance in the training phase. On the right side the weak correlation of the noise does not have much effect on the monotonous noiseless readout signal z. This results in simple acceleration in an arbitrary direction.

constant τ_z was set to $0.1995s$ and the averaging constant τ_α in Equation 3.7 at $0.0045s$ for the running average for the reward function $\bar{R}$. The noise correlation time constant τ_c remained at $5s$. The time constant for the friction τ_f was set to $0.012s$. This provided significant resistance to movement like a rough or soft surface. The learning rate η was fixed at 0.0005. The noise ξ was set to 0 during the testing period to see if the network could reach the target without aiding movement generation of the noise signal. As a strong noise signal could eventually reach the target without the readout signal and thus inhibit learning, this was necessary to measure the performance of the reservoir readout. The internal state $\mathbf{y}$ was initiated with a random value

between -1 and +1 for the position $p(t)$ and 0 for the velocity $v(t)$. The value chosen for p_{target} was 0. If the distance between p_{target} and $p(t)$ would be under 0.1 during the simulation, it would reset $\mathbf{y}(t)$.During training the number of resets in a given time frame provided a good indicator for learning progress (For a typical progression see Figure 4.4). The total number of resets during testing could be taken as abstract performance measure. To show the

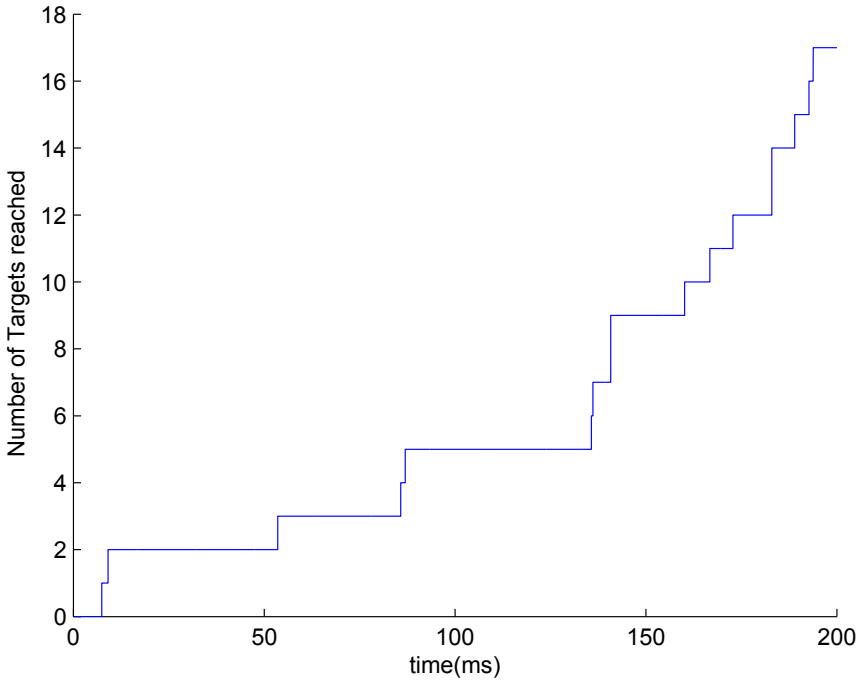

Figure 4.4: Number of Times the target was reached by the simulated object over the course of training. The influence of learning can be clearly seen towards the end.

importance of time correlated noise, the system was allowed to learn for 200 s before a 50 s testing period. In Figure 4.3 sample resulting behaviour of typical simulated systems during training and testing is shown. In Figure 4.5 activity of random neurons inside the reservoir during this training is shown. The average number of resets over 20 simulation was 12.25 with standard deviation of approximately 4 resets. Only 1 in 20 tests failed to produce any target approaching behaviour. Without correlated noise there was no success

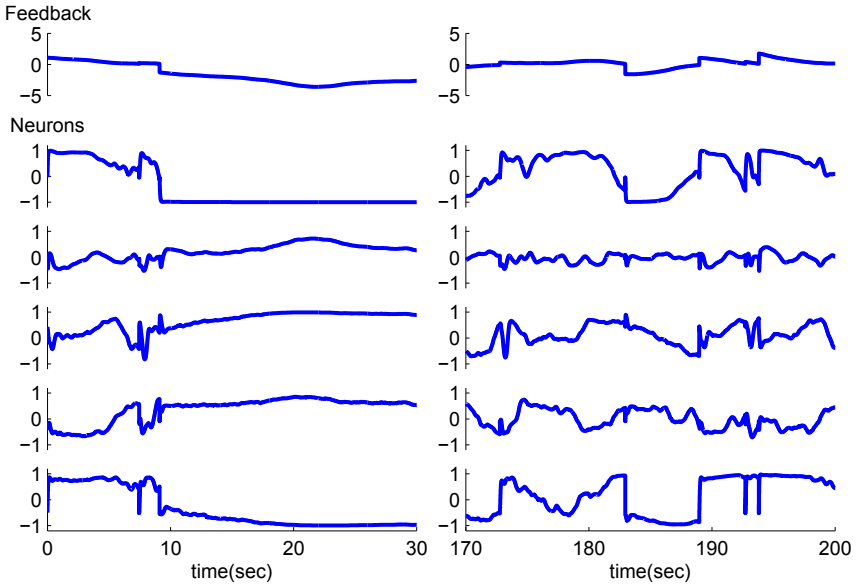

Figure 4.5: Activity of feedback signal and 5 different reservoir neurons over time during training. (Left) Activity in the first 30 seconds of training. (Right) Activity of the same neurons after 170 seconds of training in the last 30 second period. The feedback signal back to the reservoir consisted of the position of the object on a one dimensional line over time.

of reproducing any directed movement of the object. Adjusting η in this case showed no significant success. Either the system became unstable with high noise driving velocity in random directions, or the input amplitude became to weak to induce meaningful learning feedback. Either way the weights could not adapt significantly to reproduce target directed movement with or without noise present during testing.

4.2 Simulation results for 2D model

As the significance of the one-dimensional model is relatively limited we tried to show that the same conclusions are true with a more complex set-up. Additionally, we then examined the effects of different length of time

correlation of noise on learning performance. After that we took a look at the reconstruction mechanism of the noise signal z_ξ for learning to see if it still holds under different noise correlation conditions.

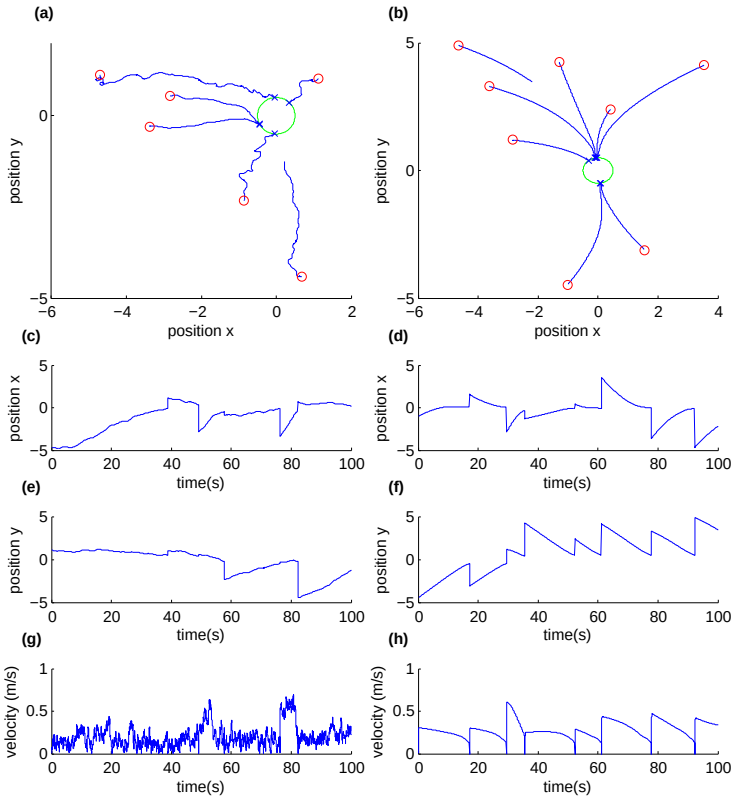

Figure 4.6: Simulation of a 2-dimensional system. Left/Right: Training/Testing. (a) and (b) show the position trajectories of the system vector $\mathbf{y}(t)$. The little red circles show the randomly generated starting point, from which the system tries to reach the target zone marked with a green circle. Blue x markers show where the trajectories reach the zone. Below, (c) (d) (e) and (f) show the x and y trajectories separately over time. The discontinuous steps show where the position is being reset after the system reaches the target zone. (g) and (h) show the absolute value of the velocity of the system.

In the second simulation the physical model explained in Section 3.3.1 was

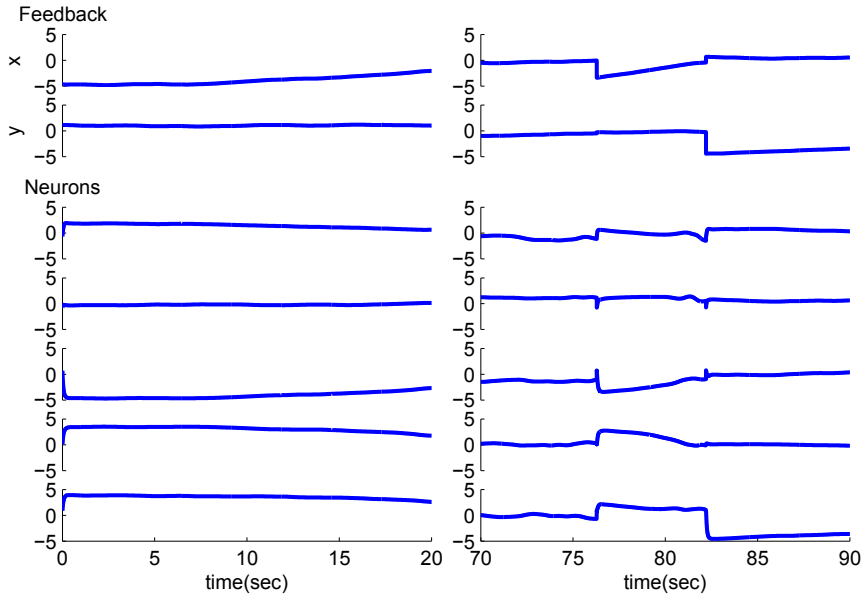

Figure 4.7: Activity of feedback signals and 5 different reservoir neurons over time during training. (Left) Activity in the first 20 seconds of training. (Right) Activity of the same neurons after 70 seconds of training. Feedback consisted of the two dimensions of the object position over time.

expanded with a second dimension (see Section 3.3.2). It was then tested with various noise settings to show the difference in performance with increasing noise correlation times. As in the first simulation the test consisted of setting the model to a random position on a flat plane. The noise induced into the control signal $z(t)$ should then be able generate movement. The resulting feedback had to excite the reservoir as well as provide the reward function with enough position changes to be able to affect the weight update. The weights of the readout would then be adapted so that the control signal would guide the model towards a predetermined target position. This time the initial values for the model position variables p_1 and p_2 were chosen between $10m$ and $-10m$ and the goal area was set up between $0.5m$ and $-0.5m$ in both axes. Thus it was possible for the model to start inside the target area, as well as up to $14m$ away from it. This also meant the direction from

the initial position to the target area was not constrained in any way to avoid imprinting a specific direction into the readout weights. The learn rate was kept at 0.0005 and the noise amplitude was fixed at 0.5 for the correlated signal. The friction constant τ_f in Equation 3.13 was set to $0.0195s$ to again provide the model with significant resistance comparable to a rough surface. At first we established a basis for the parameters that produced stable and preferable behaviour like in the one dimensional model. The noise correlation time constant τ_c was set to $0.5s$ while τ_z was set to $5s$. τ_α was left at $0.0045s$. With this settings the model was trained for 100 seconds before undergoing testing for another 100 seconds. Like in the one dimensional case the noise $\xi(t)$ was set to zero during testing. Initial positions during testing were again randomly determined as they were during training. In Figure

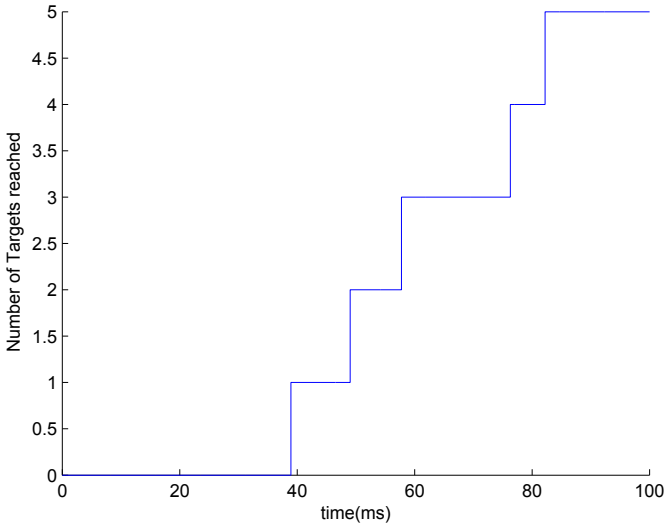

Figure 4.8: Number of Times the target was reached by the simulated object over the course of training. As in the one dimensional case learning progress is visible by a sharp increase in successes towards the end.

4.6 the resulting trajectories of one such simulation can be seen. The two dimensional object moved much more slowly, in part due to the higher friction. This is also reflected in reservoir activity, shown in Figure 4.7. As the

reward R remained a single global signal averaging over the distance to the target in both dimensions, the weights of each of the two readouts got less information about their specific dimensional distance to the target.
With higher values of the noise correlation time constant τ_c there is a high chance of weight distribution becoming unbalanced. This typically results in fast divergence of the control signal z. To counter this either the weights would have to be normalized, or the control signal would have to be limited. Another issue of high values of τ_c is the reconstruction of the temporally correlated noise z_ξ from the filtering of the readout output $u - \bar{u}$ in Equation 3.5. Figure 4.9 shows that for a given value of τ_c a corresponding value of the filtering time constant τ_z must be chosen. It also shows the behaviour of z as limiting factor. A higher τ_c with respect to a given z leaves a narrower band of values for τ_z. As seen in Figure 4.10, reconstruction might even become impossible with high enough τ_c. Additionally the behaviour of z is dependent on the circumstances of the simulation environment and initial parameters, success in learning meaningful behaviour of the model may become very circumstantial. Performance of the simulated model were measured in two ways. The first was integrating the root mean square error (RMSE) of the distance between the position of the moving mass and the target over the whole testing period. This is useful to obtain a single value for a whole simulation, but is only comparable between simulations with the same τ_c. A second value is the numbers of times the target was reached during testing. Still different parameters cause objects to move faster, reaching the target more often during a given testing period despite using similar trajectories. Performance values for reconstructing z_ξ with different values of τ_c and τ_z are shown in figure 4.11. As expected, the RMSE values rise with the rising amplitudes of the time correlated noise z_ξ. The average successes reaches a maximum at the value of τ_c which is still small enough to be able to reconstruct z_ξ, but large enough for the system to be able to explore the environment quickly. Above this value additionally the deviation of the Success Rate climbs, as the learning success of the model increasingly dependent on good starting conditions of initial system values $\bar{p}$ and weight distributions. Another question stemmed from the reservoir output $u(t)$ itself as it had to react smoothly

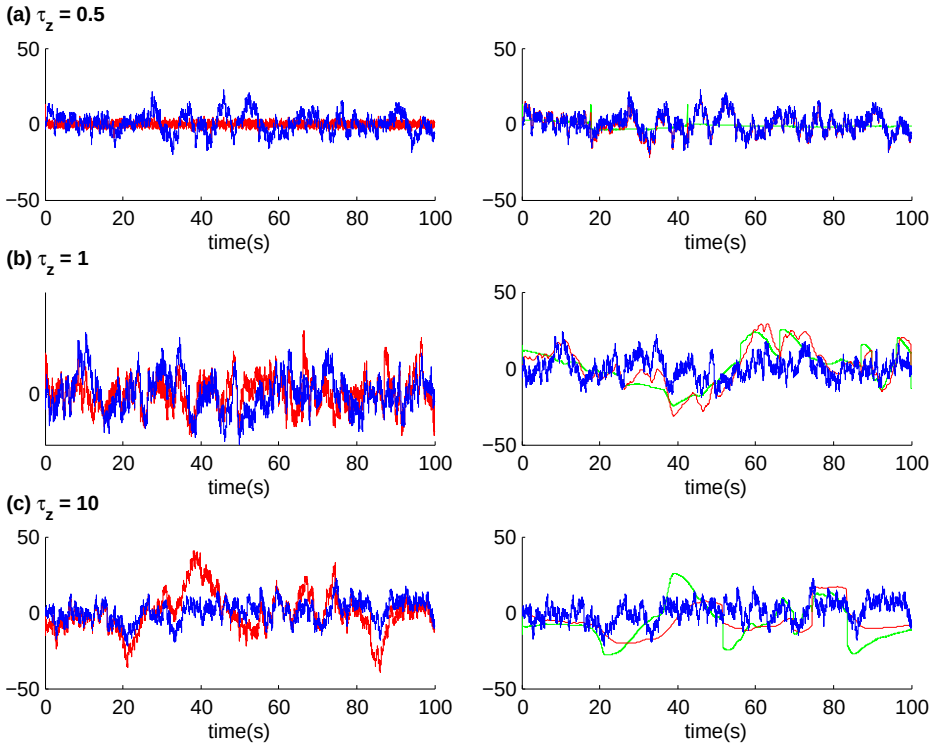

Figure 4.9: Reconstruction of exploration noise z_ξ, depends on readout output filter time constant τ_z. Shown are reconstructions for $\tau_z = 0.5s$ (a), $\tau_z = 1s$ (b), and $\tau_z = 10s$ (c) with constant noise correlation time constant $\tau_c = 1s$. Left: Reconstruction $u - \bar{u}$ (red) and original noise signal z_ξ (blue). Right: Filtered readout output $\bar{u}$ (red), noiseless readout output z (green), and noise z_ξ (blue). In (a) the filter time constant τ_z is too low, so the filtered readout output $\bar{u}$ follows the noisy readout output u. In (b) the filtered readout output $\bar{u}$ is mostly influenced by the noiseless readout output z resulting in a good reconstruction of the exploration noise z. (c) shows that a filter time constant τ_z set too high also filters changes of the noiseless readout output z and fails to reconstruct any signal.

enough that the "nearly constant" condition for the noisy position change would not be affected, yet react fast enough to correct higher velocities towards wrong directions. In the one dimensional case this was relatively easy

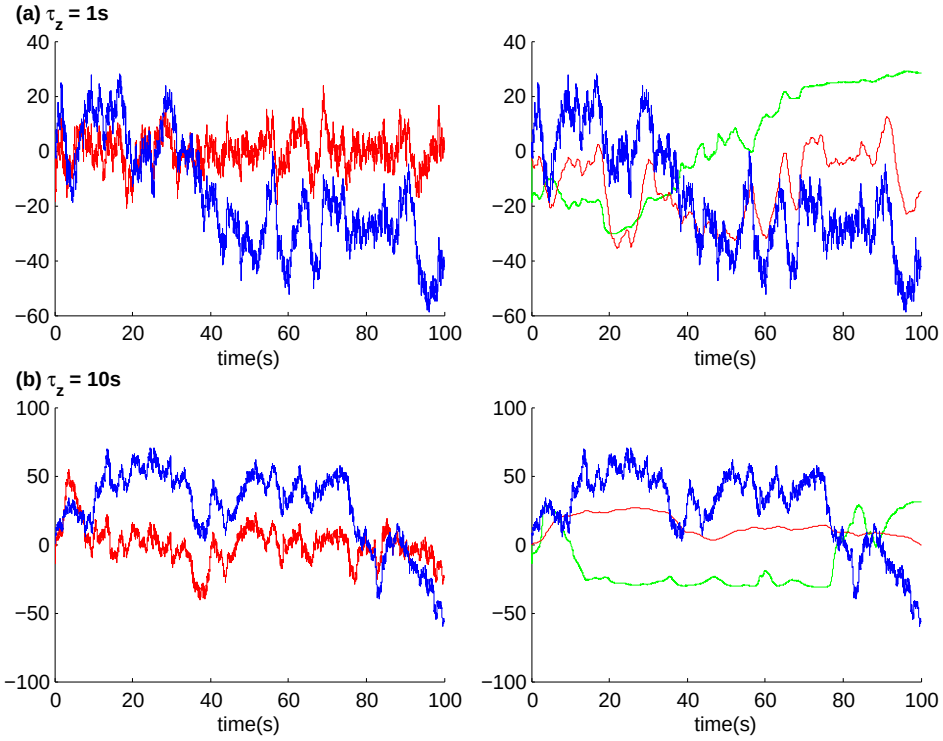

Figure 4.10: Reconstruction of z_ξ(red) for 2 different values of the readout output filter time constant τ_z with a higher value of the noise correlation time constant $\tau_c = 50s$. Shown are reconstructions for $\tau_z = 1s$ and $\tau_z = 10s$. (a) In this case the high τ_c makes the slope of z_ξ much slower, and on the right side $\bar{u}$(blue) is influenced by it as much as by z (green). The effect causes $u - \bar{u}$ (blue) on the left to differ significantly from z_ξ. (b) Below is shown that a higher value of τ_z filters both signals.

as the simulated object either moved towards the target or away from it. In the expanded model there was the possibility to pass by the target where the reward function could change from positive to negative value in a short time frame. The model had to be able to reverse the velocity completely in very short time as to not move away much further, or move very slowly in the first place. The maximum velocity of the object was only limited indirectly by

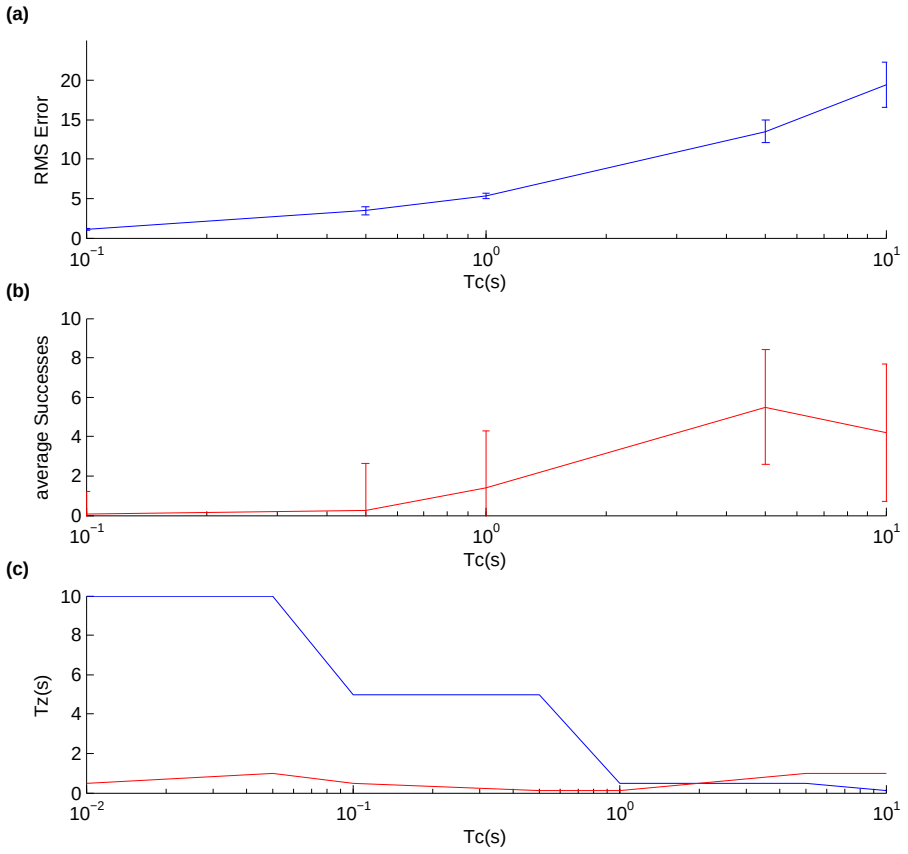

Figure 4.11: Performance measures for a range of noise correlation time constant values τ_c. (a) shows RMS Error, (b) averaged Numbers of the Simulation finding the target area and (c) shows the corresponding values of the readout output filter time constant τ_z for (a) (blue) and (b) (red) for each value of τ_c which produced the shown result. Success rate peak at the largest value of τ_c able to reliably reconstruct z_ξ. The values were averaged over 20 simulations, with both testing and training phase durations of 100s.

the friction. These considerations formed the constraints of distance values between initial and target position as well as friction coefficients and source noise amplitude strength. The question remains how these constraints could be possibly loosened or completely overcome.

4.3 Simulation results for robot arm

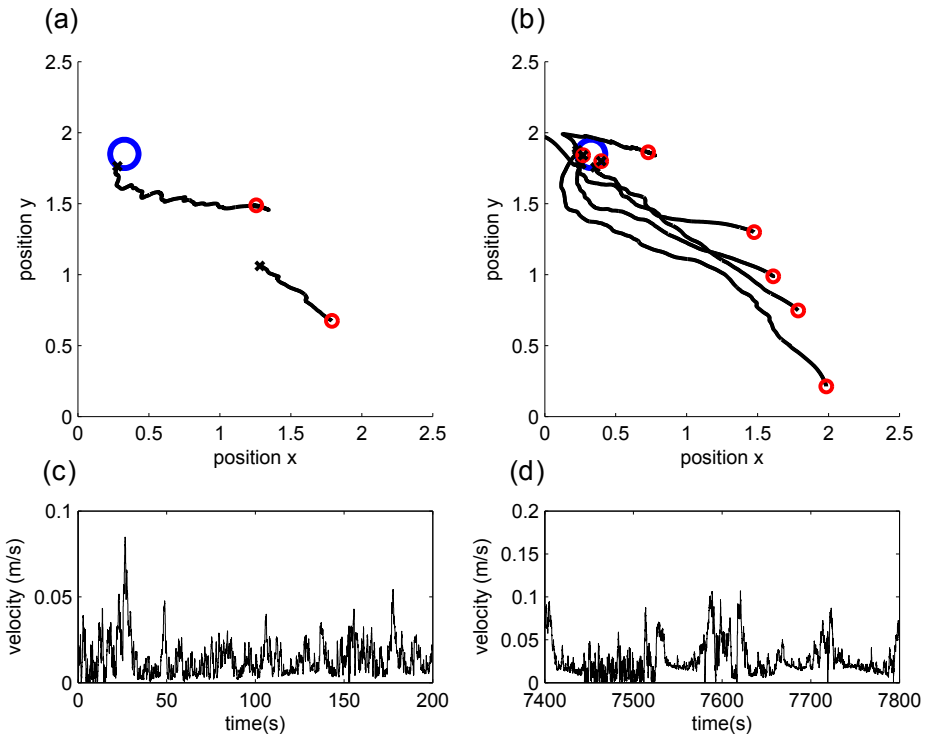

Figure 4.12: Movement trajectories and velocity during beginning and ending of training period. On the left side (a) trajectory of end point movement and (c) corresponding velocity during the first 200s of training. On the right side (b) trajectories and(d) velocity is shown later during the training period. Of note is the considerably higher velocity later during the training. The blue circle denotes the area around the target and red circles show Initial Positions.

For a more plausible motor control experiment, the 2D model was replaced with the simulation of a 2-link robotic arm presented in Chapter 3.3.3. A schematic of the arm is shown in Figure 3.2. Feedback into the reservoir was encoded by an additional module consisting of several radial basis kernels as discussed in Chapter 3.3.3. The goal of the experiment is for

the endpoint m_2 of the robot arm to reach a defined target point p_t within reach of the arm. As p_t was defined in cartesian space, the reward function was computed in a similar way as distance between the end point of the arm and the target point(see Equation 3.20). This allowed for ambiguity in target angles for the arm as 2 very different points $\phi(t)$ in angular space could result in the same reward value. The simulation was discretized with time steps of

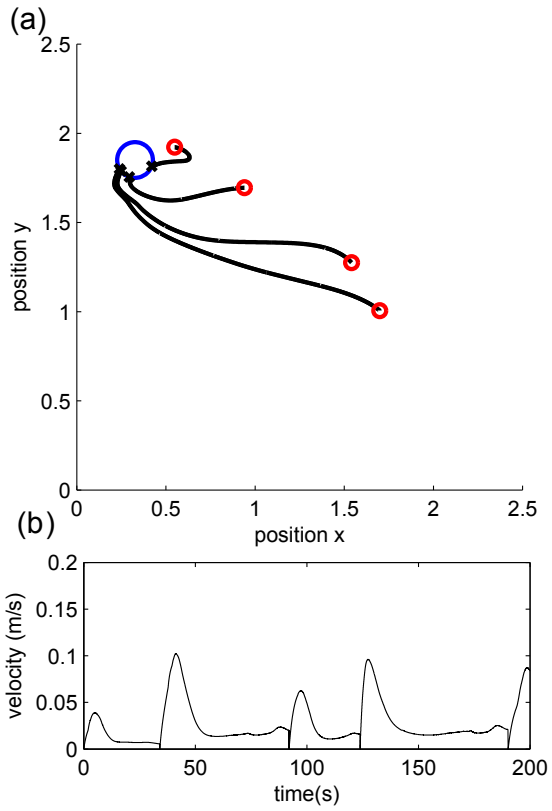

Figure 4.13: Movement trajectories and velocity during testing. (a) The arm reaches the target repeatedly and is set back to another random initial position several times. (b) Velocity values during the $200s$ testing period. The model was trained for $8000s$ with $\tau_c = 0.05s$ before testing.

$\Delta t = 0.0005s$. The time constant τ of the reservoir (from Equation 3.1) was also set to $\tau = 0.0005s$ to avoid problems with different discretization times

between model and simulation. The friction constant f was set to 0.5 which was thought to be slightly higher than normal for such a robot arm but still plausible. The initial position of the arm is determined by 2 uniformly distributed random variables determining the angles. This random variation of each angle was limited between 0° and 80° to reduce training time and stability issues. Because these angles are linked together, the initial positions of the end point of the arm are distributed along a circular ring with declining density towards the boundaries of the ring. Figure 4.12 shows movement of

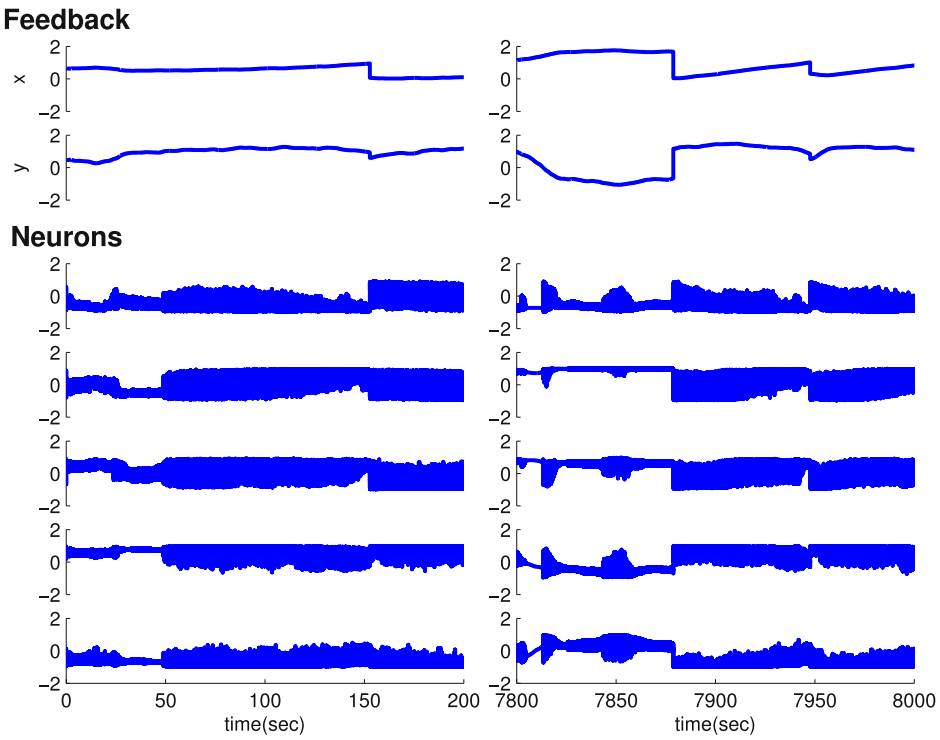

Figure 4.14: Feedback Signal and Activity of Reservoir Neurons during Training. (Left) Activity during the first 200s of training. (Right) Activity during the last 200s of training period. The feedback signal consists of the x and y cartesian dimensions of the end-point position $\mathbf{p}(t)$ of the arm model before being transformed by radial basis functions.

the arm end-point in two different time periods during training where this distribution is visible. The model was trained for $8000s$ with τ_c of $0.05s$ and a noise amplitude ξ_{max} of 0.0001. The constant β used in Equation 3.15 to scale the readout outputs was set to 0.01. Movement during testing of the trained model is shown in Figure 4.13. Neural activity inside the reservoir differs from the simple models, as seen in Figure 4.14. Performance during training of the model remained relatively constant after some time. The whole progress of targets reached during training is presented in Figure 4.15. It also shows that during some periods it took the model particularly long to reach the target. This could mean outlying initial position values were chosen.

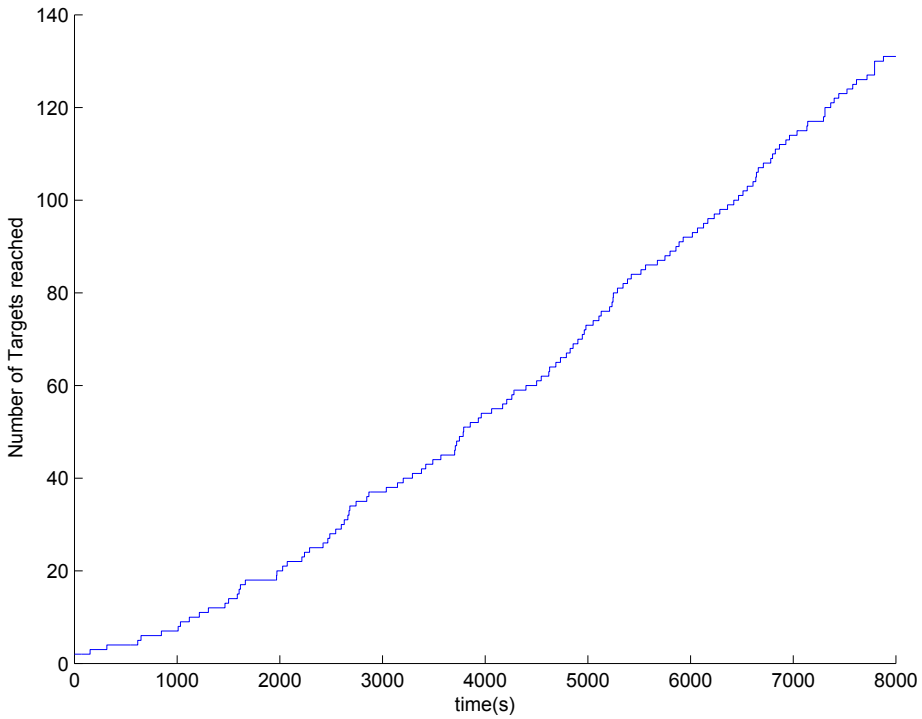

Figure 4.15: Performance measured in number of times the target position was reached over time during the training of $8000s$.

Chapter 5

Discussion

In this work we successfully utilized a learning rule that uses exploration noise as an exploratory signal for parameter adaptation to solve motor control tasks. It has already been shown that this learning rule, the EH-rule proposed by [Legenstein et al., 2010b], is able to simulate experimental results from [Jarosiewicz et al., 2008]. It is also able to perform a variety of specialized computational functions [Hoerzer et al., 2011]. Since the main driving force of weight changes is the correlation between network performance and noise, it is critical that noise can lead to virtually immediate changes in the performance signal. Additionally, it was assumed in [Legenstein et al., 2010b] and [Hoerzer et al., 2011] that the noise is temporally uncorrelated. In combination, these two assumptions are problematic for real-world motor control. Since all mechanical systems have some inertia, fast varying readout noise may be filtered out by the effectors. As a result, performance improvements cannot be estimated and communicated to plasticity mechanisms. We show through simulations that this problem can be overcome by readouts with temporally correlated noise statistics. Temporally correlated exploration noise lead to long-lasting perturbations of motor commands and in turn to clearly visible perturbations of effector trajectories in space. The effectiveness of these perturbations with regard to performance can be estimated e.g. through visual feedback and signalled via the reward signal $R(t)$.

This principle works very well in the case of our simulation of a simple 2D motor control task, which can be viewed as an agent traversing rough terrain to reach a certain target. High kinetic friction stemming from a surface acts as an additional inhibition to changes made by the exploration noise. Adding temporally correlated noise overcame this inhibition and enabled the performance signal to enact useful changes in readout weights.
In the case of a simulated robot arm this proved to be much more difficult as high maximum torques and low friction values allowed for very fast movement. This in turn necessitated very fine tuning of control signal and noise amplitudes to avoid issues with stability and very high velocities from which the system could not recover. Additionally the interconnection of the two joints generates an additional velocity correlated perturbation whose internal dynamics are completely unknown to the network. This not only acts as additional noise on the control signal, it may also weaken the correlation between any movement generated by the exploration noise and the resulting reward signal. Despite these complex issues, controlling the arm model proved to be possible, as long as arm velocity remained fairly low. In general motor control of a complex object might be influenced by many different factors, of which inertia and friction related effects are only part. For some tasks these two effects might not even have significant influence. Thus, the usefulness of temporal correlation of exploration noise depends heavily on the actual task and model constraints.
When taking into consideration a reconstruction of the exploration noise from the control signal for use in the EH-rule, an upper limit for the correlation constant τ_c becomes apparent. This was done by [Legenstein et al., 2010b] so the system wouldn't need specific knowledge of the noise. We believe this limit to be specific to the dynamics of a given system which might pose a problem when simulating some models. It might be possible for the lowest τ_c to be higher then the upper limit, making noise reconstruction impossible.

5.1 Future work

Motor control tasks can take on diverse forms of which we only considered the most generalized models. When pertaining to practical use, there are still several issues that need to be addressed. One challenging issue concerns the environment, which can often consist of various different obstacles and areas with varying friction and elevation. This leads to the question whether a single static time correlation coefficient τ_c value is sufficient for the exploration noise to be effective in changing surroundings. One could easily imagine a separate readout layer controlling the value of τ_c , but that would leave the question of how the weights of this layer would be found. Such mechanism would also alleviate the need of finding a suitable τ_c manually.
Other issues arise from the internal dynamics of the controlled object which need to be countered by the control signal. Depending on the physical model of the agent a separate phase in the learning process might be needed, where specific prearranged impulses of motor activity allow the model to experience the reaction of the object. Another solution might be to separate the reservoir into one part encoding the environment and one part which only counteracts the model.

5.2 Conclusion

Real-world motor control often places many restrictions on learning mechanisms and underlying models. This intensifies if one takes into account models of biological computation such as neural networks and unsupervised learning paradigm. In this work we provide first proof of principle that generic cortical networks can attain functions essential for motor control through a biologically plausible reward-modulated Hebbian learning rule. We employ noise with temporal correlations, as opposed to previous models [Hoerzer et al., 2011, Legenstein et al., 2010b], which is used to overcome restrictions inherent in physical models. Using this noise, we show that it is difficult but possible to control a model of a multi-joint arm.

Appendix A

Time-discrete Formulas

While the model was constructed in a continuous time space, the the actual computer simulations had to be conducted with a discrete abstraction. The discretized equations that were used are given here. For the 1D and 2D models of a moving singular mass a discretization time step of $\Delta t = 1$ms was used. In the simulation of the robot arm a time step of $\Delta t = 0.5$ms was chosen. In all simulations both the reservoir and the physical model were simulated with the same time step. The internal state of the reservoir thus updated in the following way instead of Equation 3.1:

$$x_i[n+1] = \left(\frac{1-\Delta t}{\tau}\right) x_i[n] + \lambda \sum_{j=1}^{N} w_{ij}^{\text{rec}} r_j[n] + \sum_{j=1}^{M} w_{ij}^{\text{fb}} y_j[n] \tag{A.1}$$

For the two moving averages $\bar{u}$ and $\bar{R}$ this amounted to

$$\bar{u}[n] = \exp^{-\frac{\Delta t}{\tau_z}} \bar{u}[n-1] + (1 - \exp^{-\frac{\Delta t}{\tau_z}}) u[n] \tag{A.2}$$

in place of Equation 3.6 and

$$\bar{R}[n] = \exp^{-\frac{\Delta t}{\tau_\alpha}} \bar{R}[n-1] + \left(1 - \exp^{-\frac{\Delta t}{\tau_\alpha}}\right) R[n] \tag{A.3}$$

for Equation 3.7. In case of the exploration noise the random variable vector

ξ was discretely drawn at each time step and added up.

$$\mathbf{z}_\xi[\mathbf{n}] = \exp^{-\frac{\Delta t}{\tau_c}} \mathbf{z}_\xi[n-1] + \xi \tag{A.4}$$

Like the reservoir states, the states of the physical models also had to be discretized. For the simple 1D and 2D models Equation 3.9 was replaced by

$$\mathbf{y}[n+1] = A\mathbf{y}[n] + B\mathbf{u}[n] \tag{A.5}$$

with the following parameter matrices A and B for the one-dimensional model:

$$A = \begin{pmatrix} 1 & \Delta t \\ 0 & \exp^{-\frac{\Delta t}{\tau_f}} \end{pmatrix},\ B = \begin{pmatrix} 0 \\ \Delta t \end{pmatrix} \tag{A.6}$$

For the two-dimensional model the matrices A and B look the following way:

$$A = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & \exp^{-\frac{\Delta t}{\tau_f}} & 0 \\ 0 & 0 & 0 & \exp^{-\frac{\Delta t}{\tau_f}} \end{pmatrix},\ B = \begin{pmatrix} 0 \\ 0 \\ \Delta t \\ \Delta t \end{pmatrix} \tag{A.7}$$

In case of the robot arm model this allowed us to bypass numerical integration methods and compute the positions and velocities strictly from travel distances and acceleration $\dot{\omega}$ which was computed by

$$\dot{\omega}[n] = H^{-1}[n]\left(\mathbf{u}[n] - C\omega[\mathbf{n}-\mathbf{1}]\right) - f\omega[n] \tag{A.8}$$

Appendix B

Matlab Code

This appendix contains code used for the Simulation of the two dimensional model. Code for the 1D model is the same except for dimensions. For the simulation of the robot arm, the reward function and model computation was changed accordingly, as documented in chapter 3.

```
function [M,wo,wf,zt,ztempt,z_t,rew,rew_,zxit, pos, pos_test,
    vel,vel_test, trials_pos, trials_pos_test,wo_norm,
    trials_time, trials_time_test,res_units] = regler2D_script(
    corr,rec,eta,velmax,avgrew,traintime,testtime,dt,Tc,Tz)
use_testing_noise = 0;  %noise during testing
use_correlation = corr; %Use correlating noise or not [0,1]
use_random_init = 0;    %Output Weights start random or 0
use_real_noise = rec; %reconstruct noise
reset = 1;              %Reset state if target reached
%Simulation Time
nsecs_train = traintime;
nsecs_test = testtime;
simtime = 0:dt:nsecs_train-dt;
simtime_len = length(simtime);
simtime2 = nsecs_train:dt:nsecs_train+nsecs_test-dt;
simtime2_len = length(simtime2);
%Reservoir Initialization
tau = 0.05;
N = 1000; %Reservoir Size
```

```
17 p = 0.1;
18 g = 1.5; % g greater than 1 leads to chaotic networks.
19 scale = 1.0/sqrt(p*N);
20 M = sprandn(N,N,p)*g*scale;
21 M = full(M);
22 nRec2Out = N;
23 state_noise = 0.05; % state noise level
24 init_noise = 1e-2;
25 if (use_random_init == 0)
26     wo = zeros(nRec2Out+1,2); %readout weights
27 else
28     wo = init_noise*2.0*(rand(nRec2Out+1,2) - 0.5);
29 end
30 dw = zeros(nRec2Out+1,2); %differential weight change
31 wf = 2.0*(rand(N,4) -0.5); % feedback weights
32 %Memory Allocation
33 zt = zeros(2,simtime_len);
34 z_t = zeros(2,simtime_len);
35 zxit = zeros(2,simtime_len);
36 ztempt = zeros(2,simtime_len);
37 pos = zeros(2,simtime_len);
38 vel = zeros(2,simtime_len);
39 rew = zeros(1,simtime_len);
40 rew_ = zeros(1,simtime_len);
41 wo_norm = zeros(1,simtime_len);
42 realrew = zeros(1,simtime_len);
43 zt_test = zeros(2,simtime2_len);
44 pos_test = zeros(2,simtime2_len);
45 vel_test = zeros(2,simtime2_len);
46 zxit_test = zeros(2,simtime2_len);
47 trials_time = zeros(1,100);
48 trials_time_test = zeros(1,100);
49 trials_time(1) = 1;
50 trials_time_test(1) = 1;
51 %system variable initialization
52 pos_spread = 10;
53 initpos = pos_spread*(2.0*rand(2,1) - 0.5);
54 y0 = [initpos;0;0];
55 x0 = 0.5*randn(N,1);
```

```
z0 = 0.5*randn(2,1);
y = y0; x = x0;
r = tanh(x);
z_ = z0;
zxi = [0;0];
R_ = - sqrt((y(1)-pt(1))^2 + (y(2)-pt(2))^2);
%System
Tr_ = -0.001/log(avgrew);
Tr = -0.001/log(1-avgrew);
Tz_ = Tz;
pt = [0,0]; %Target
if(use_correlation == 1)
    noise = 0.5; %noise amplitude
    c = exp(-dt/Tc); %Time dependence of noise
else
    noise = 24.0;
    c = 0;
end
A = [1, 0, dt, 0;
    0, 1, 0, dt;
    0, 0, exp(-dt/velmax), 0;
    0, 0, 0, exp(-dt/velmax)];
B = [0; 0; dt; 0];
C = [0; 0; 0; dt];
%test variable initialization
trials_pos = 0;
ti = 0; tj = 0;
for t = simtime
    ti = ti+1; tj = tj+1;
    %Compute reservoir output
    x = (1.0-dt/tau)*x + M*(r*dt/tau) +wf*([y(1);y(2);0;0]*dt/
        tau); %feedback of position only
    r = tanh(x)+2.0*state_noise*(rand(size(r))-0.5);
    ztemp = wo'*[r;1];
    %Add Noise
    xi = noise*2.0*(rand(2,1)-0.5);
    zxi = c*zxi + xi;
    z = ztemp + zxi;
    %Compute model output
```

```
94      y = A*y + B*z(1) + C*z(2);
95      %Compute reward function and moving averages
96      R = - sqrt((y(1)-pt(1))^2 + (y(2)-pt(2))^2);
97      R_ = exp(-dt/Tr_)*R_ + exp(-dt/Tr)*R;
98      fR = R-R_;
99      if(z_ == [0;0])
100         z_ = z;
101     else
102         z_ = exp(-dt/Tz_)*z_ + ( 1- exp(-dt/Tz_))*z;
103     end
104     %Update weights
105     if (use_real_noise == 1)
106         dw = fR*eta*[r;1]*(zxi)';
107     else
108         dw = fR*eta*[r;1]*(z-z_)';
109     end
110     wo = wo + dw;
111     % Store the output of the system
112     ztempt(:,ti) = ztemp;
113     zt(:,ti) = z;
114     z_t(:,ti) = z_;
115     pos(:,ti) = y(1:2);
116     vel(:,ti) = y(3:4);
117     zxit(:,ti) = zxi;
118     rew(ti) = R;
119     rew_(ti) = R_;
120     realrew(ti) = fR;
121     wo_norm(ti) = norm(wo);
122   res_units(:,ti) = x(1:10);
123     %Reset model reached the target
124     err = sqrt((y(1)-pt(1))^2 + (y(2)-pt(2))^2);
125     if (err < 0.5)
126         trials_pos = trials_pos + 1;
127         trials_time(trials_pos+1) = ti;
128         tj = 0;
129         if (reset)
130             initpos = pos_spread*(rand(2,1) - 0.5);
131             y = [initpos;0;0];
132             R_ = - sqrt((y(1)-pt(1))^2 + (y(2)-pt(2))^2);
```

```
133                 z_ = [0;0];
134                 x = x0;
135                 r = tanh(x);
136             end
137         end
138     end
139
140     %% Testing
141     ti = 0; tj = 0;
142     trials_pos_test = 0;
143     if(reset)
144         initpos = pos_spread*(rand(2,1) - 0.5);
145         y = [initpos;0;0];
146         x = x0;
147         r = tanh(x);
148     end
149     for t = simtime2
150       ti = ti+1; tj = tj+1;
151         % Simulate Reservoir
152         x = (1.0-(dt/tau))*x + M*(r*(dt/tau)) + wf*([y(1);y(2)
                ;0;0]*dt/tau);
153         r = tanh(x);
154         zopt = wo'*[r;1];
155         %Add noise
156         xi = noise*2.0*(rand(2,1)-0.5);
157         zxi = c*zxi + xi;
158         if(use_testing_noise)
159             z = zopt + zxi;
160         else
161             z = zopt;
162         end
163         %Compute System output
164         y = A*y + B*z(1) + C*z(2);
165         %Check if target is reached
166         if (sqrt((y(1)-pt(1))^2 + (y(2)-pt(2))^2) < 0.5)
167             trials_pos_test = trials_pos_test + 1;
168             trials_time_test(trials_pos_test+1) = ti;
169             tj = 0;
170             %reset model and reservoir position and state
```

```
171             if (reset)
172                 initpos = pos_spread*(rand(2,1) - 0.5);
173                 y = [initpos;0;0];
174                 x = x0;
175                 r = tanh(x);
176             end
177         end
178 
179         zt_test(:,ti) = z;
180         pos_test(:,ti) = y(1:2);
181         vel_test(:,ti) = y(3:4);
182         zxit_test(:,ti) = zxi;
183     end
```

Bibliography

[Amit, 1989] Amit, D. J. (1989). *Modeling brain function: the world of attractor neural networks*. Cambridge University Press. (Cited on page 1.)

[Bi and Poo, 1998] Bi, G. and Poo, M. (1998). Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *J Neuroscience*, 18(24):10464–10472. (Cited on page 11.)

[Buonomano and Maass, 2009] Buonomano, D. and Maass, W. (2009). State-dependent computations: Spatiotemporal processing in cortical networks. *Nature Reviews in Neuroscience*, 10(2):113–125. (Cited on page 5.)

[Hebb, 1949] Hebb, D. O. (1949). *The Organization of Behavior*. Wiley, New York. (Cited on page 11.)

[Hoerzer et al., 2011] Hoerzer, G., Legenstein, R., and Maass, W. (2011). Emergence of complex computational structures from chaotic neural networks through reward-modulated hebbian learning. *none*. (Cited on pages 2, 5, 6, 13, 26, 46 and 48.)

[Hopfield, 1982] Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proc. Nat. Acad. Sci. USA*, 79:2554–2558. (Cited on page 1.)

[Jäger, 2001] Jäger, H. (2001). The "echo state" approach to analyzing and training recurrent neural networks. GMD Report 148, German National Research Center for Information Technology. (Cited on pages 2, 8, 9 and 12.)

[Jarosiewicz et al., 2008] Jarosiewicz, B., Chase, S. M., Fraser, G. W., Velliste, M., Kass, R. E., and Schwartz, A. B. (2008). Functional network reorganization during learning in a brain-computer interface paradigm. *Proc. Nat. Acad. Sci. USA*. in press. (Cited on page 46.)

[Legenstein et al., 2010a] Legenstein, R., Chase, S., Schwartz, A., and Maass, W. (2010a). A reward-modulated hebbian learning rule can explain experimentally observed network reorganization in a brain control task. *J Neurosci*, 30(25):8400–8410. (Cited on page 18.)

[Legenstein et al., 2008] Legenstein, R., Pecevski, D., and Maass, W. (2008). A learning theory for reward-modulated spike-timing-dependent plasticity with application to biofeedback. *PLoS Computational Biology*, 4(10):1–27. (Cited on page 11.)

[Legenstein et al., 2010b] Legenstein, R., Wilbert, N., and Wiskott, L. (2010b). Reinforcement learning on slow features of high-dimensional input streams. *PLoS Computational Biology*, 6(8):e1000894. (Cited on pages 12, 46, 47 and 48.)

[Maass, 1996] Maass, W. (1996). Lower bounds for the computational power of networks of spiking neurons. *Neural Computation*, 8(1):1–40. (Cited on page 4.)

[Maass et al., 2007] Maass, W., Joshi, P., and Sontag, E. D. (2007). Computational aspects of feedback in neural circuits. *PLoS Computational Biology*, 3(1):e165, 1–20. (Cited on page 7.)

[Maass et al., 2002] Maass, W., Natschlaeger, T., and Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560. (Cited on pages 2, 5, 6 and 7.)

[Markram et al., 1998] Markram, H., Wang, Y., and Tsodyks, M. (1998). Differential signaling via the same axon of neocortical pyramidal neurons. *PNAS*, 95:5323–5328. (Cited on page 11.)

[McCulloch and Pitts, 1943] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133. (Cited on page 1.)

[Minsky and Papert, 1988] Minsky, M. and Papert, S. (1988). *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA. (Cited on page 1.)

[Pawlak et al., 2010] Pawlak, V., Wickens, J. R., Kirkwood, A., and Kerr, J. N. (2010). Timing is not everything: Neuromodulation opens the STDP gate. *Frontiers in Synaptic Neuroscience*, 2:146. (Cited on page 11.)

[Pearlmutter, 1995] Pearlmutter, B. A. (1995). Gradient calculation for dynamic recurrent neural networks: a survey. *IEEE Trans. on Neural Networks*, 6(5):1212–1228. (Cited on page 1.)

[Rosenblatt, 1962] Rosenblatt, J. F. (1962). *Principles of Neurodynamics*. Spartan Books, New York. (Cited on page 1.)

[Rumelhart et al., 1986] Rumelhart, D., Hinton, G., and Williams, R. (1986). Learning representations by back-propagating errors. *Nature*, 323:533–536. (Cited on page 1.)

[Schultz, 2007] Schultz, W. (2007). Behavioral dopamine signals. *Trends in Neuroscience*, 30:203–210. (Cited on page 11.)

[Sussillo and Abbott, 2009] Sussillo, D. and Abbott, L. F. (2009). Generating coherent patterns of activity from chaotic neural networks. *Neuron*, 63(4):544–557. (Cited on page 13.)

[Sutton and Barto, 1998] Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA. (Cited on page 10.)

[Thorndike, 1911] Thorndike, E. L. (1911). *Animal intelligence: Experimental Studies*. Macmillan. (Cited on page 10.)