Design strategies for weight matrices of Echo State Networks

Tobias Strauß[1], Welf Wustlich[1], Roger Labahn[1]
[1] University of Rostock, Germany

January 20, 2012

Keywords: Echo State Networks, recurrent neural networks

This article develops approaches to generate dynamical reservoirs of ESNs with desired properties reducing the amount of randomness. It is possible to create weight matrices with a predefined singular value spectrum. The procedure guarantees stability (Echo State property) and minimizes the impact of noise on the training process. The final reservoir types were already known in the literature but related input weights were out of the scope of previous articles. But our experiments show, that well chosen input weights can improve performance.

1. Introduction

A considerable problem in theory and practice of artificial recurrent neural networks (RNN) is the training of recurrent connections. Various gradient descent algorithms (e.g. BPTT or RTRL, see Jaeger (2002)) have been developed, but they suffer from slow convergence and local minima. An approach to avoid these problems is the Echo State approach (see e.g. Jaeger (2001a)) which was introduced at the beginning of this millennium. The *Echo State Networks* (ESN) consist of three layer: An input layer, a hidden layer with recurrent connections (called *dynamical reservoir* (DR)), and the output layer. In contrast to ordinary networks, a fundamental concept of ESNs is to train only the output connections, all other connections remain fixed. This simple but effective training approach ensures a good performance on many tasks (see e.g. Lukoševičius and Jaeger (2009)).

State of the art in creating dynamical reservoirs is to choose the connections randomly, and afterwards, to scale the spectral radius of the related weight matrix to (often slightly) lower 1. This is done to satisfy the necessary condition of the stability property namely the "Echo State property". A point of criticism of this method is that it does not ensure stability since the sufficient condition is much more restrictive and often unregarded. But practically more relevant is that random matrices damp "echos" with different "directions" unequally. This in turn means that certain parts of the signal might vanish fast compared to other parts. We aim to construct reservoirs preserving a variety of "echos", but the literature does not provide a construction algorithm for such reservoirs.

An approach of creating reservoirs with certain dynamics is given in Ozturk et al. (2007). There the authors investigate the average state entropy and motivate a maximization of this value to obtain better performing reservoirs. In contrast, we analyze the linear algebraic properties of a given weight matrix. We intend to adjust all eigenvalues and singular values. Furthermore, this article is targeted at a deeper insight of the internal dynamics of these reservoirs at least in the linear case.

In the next section, we briefly introduce the backgrounds of Echo State Networks. Afterwards in section 3, we present methods to create weight matrices with desired properties. Those weight matrices shall be equipped with predefined singular values and eigenvalues, easily verifiable stability and a long short term memory. In section 4, we analyze the impact of disturbances on Echo State Networks: Here we show that our new matrices are more robust against noise than randomly created ones. We have also tested the new reservoirs on standard experiments with convincing success and describe the observations in section 5. There we also investigate different ways of connecting the inputs to the reservoir.

2. Echo State Networks

Echo State Networks (as well as most other neural networks) consist of neurons and links. Each neuron has a time dependent activation. We distinguish three kinds of neurons:

- input units providing external activations $\boldsymbol{u} \in \mathbb{R}^K$
- hidden units with internal activations $\boldsymbol{x} \in \mathbb{R}^N$
- output units which generate the systems output signal $\boldsymbol{y} \in \mathbb{R}^L$

The connections between neurons are weighted. We collect these weights in weight matrices which are usually denoted by $\boldsymbol{W}$ and some superscript referring to the location of the respective connections. The activations are calculated by

$$\begin{aligned}\boldsymbol{x}(n+1) &= f(\boldsymbol{W}\boldsymbol{x}(n) + \boldsymbol{W}^{in}\boldsymbol{u}(n+1) + \boldsymbol{W}^{back}\boldsymbol{y}(n)) && (1)\\ \boldsymbol{y}(n+1) &= f^{out}(\boldsymbol{W}^{out}(\boldsymbol{u}(n+1), \boldsymbol{x}(n+1), \boldsymbol{y}(n))). && (2)\end{aligned}$$

The functions $f : \mathbb{R} \to \mathbb{R}$ and $f^{out} : \mathbb{R} \to \mathbb{R}$ are applied component-wise. Typically, f is a sigmoid function (like tanh). We assume $f^{out} = \mathrm{id}$ throughout this article. The hidden layer is called *dynamical reservoir (DR)*. Once initialized, the weight matrix $\boldsymbol{W}$ of the DR remains constant like most of the other weights (e.g. weights of the input $\boldsymbol{W}^{in}$ or

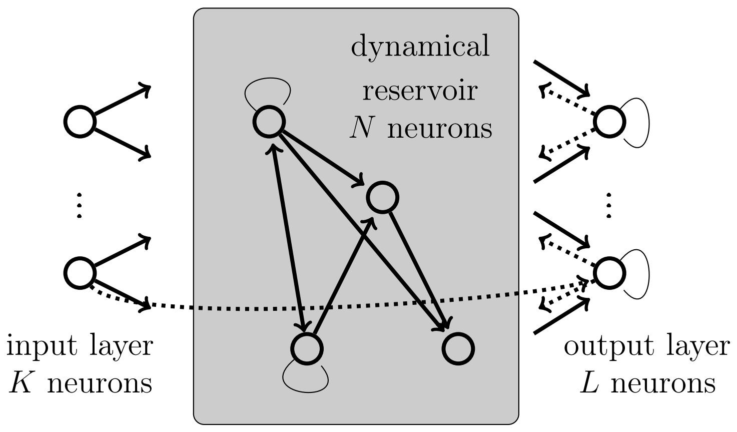

Figure 1: Layers of an Echo State Network. Dashed connections are ignored in our theoretical investigations. Feedback (output-to-hidden) connections will be used only in section 5.3.

feedback connections $\boldsymbol{W}^{back}$). Only the output weights $\boldsymbol{W}^{out}$ are trained usually by a fast linear regression algorithm. The states and the targets are collected row-wisely in the matrices $\boldsymbol{S}$ and $\boldsymbol{T}$, respectively. Then (if $f^{out} = \mathrm{id}$) we have to solve

$$\boldsymbol{S}(\boldsymbol{W}^{out})^T = \boldsymbol{T}. \tag{3}$$

Definition 1 (Jaeger (2001a)). Let $\boldsymbol{u}(n) \in U$ and $\boldsymbol{x}(n) \in A$ with compact spaces U and A. Assume that the network has no output feedback connections. Then, the network has Echo States if the network state $\boldsymbol{x}(n)$ is uniquely determined by any left-infinite input sequence $\overline{\boldsymbol{u}}^{-\infty}$. More precisely, this means that for every input sequence $\ldots, \boldsymbol{u}(n-1), \boldsymbol{u}(n) \in U^{-\mathbb{N}}$, for all state sequences $\ldots, \boldsymbol{x}(n-1), \boldsymbol{x}(n)$ and $\ldots, \boldsymbol{x}'(n-1), \boldsymbol{x}'(n) \in A^{-\mathbb{N}}$, where for any $i \in \mathbb{Z}$

$$\boldsymbol{x}(i) = f(\boldsymbol{W}^{in}\boldsymbol{u}(i) + \boldsymbol{W}\boldsymbol{x}(i-1)) \tag{4}$$

and

$$\boldsymbol{x}'(i) = f(\boldsymbol{W}^{in}\boldsymbol{u}(i) + \boldsymbol{W}\boldsymbol{x}'(i-1)) \tag{5}$$

it holds that $\boldsymbol{x}(n) = \boldsymbol{x}'(n)$ for any n.

Theorem 2 (Jaeger (2001a)). Assume $f = \tanh$ and a network without output feedback.

(a) Let the weight matrix $\boldsymbol{W}$ satisfy $\sigma_{max} = \Lambda < 1$, where σ_{max} is it's largest singular value. Then $\|\boldsymbol{x}(n+1) - \boldsymbol{x}'(n+1)\|_2 < \Lambda \|\boldsymbol{x}(n) - \boldsymbol{x}'(n)\|_2$ for all inputs $\boldsymbol{u}(n+1)$, for all states $\boldsymbol{x}(n), \boldsymbol{x}'(n) \in [-1,1]^N$. This implies Echo States for all inputs $\boldsymbol{u}(n+1)$, for all states $\boldsymbol{x}(n), \boldsymbol{x}'(n) \in [-1,1]^N$.

(b) Let the weight matrix $\boldsymbol{W}$ have a spectral radius $\rho(\boldsymbol{W}) > 1$, where $\rho(\boldsymbol{W})$ is an eigenvalue of $\boldsymbol{W}$ with the largest absolute value. Then the network has an asymptotically unstable null state. This implies that it has no Echo States for any input set U containing 0 and admissible state set $A = [-1, 1]^N$.

See Jaeger (2001a) for the proof. Obviously, Theorem 2 provides a necessary and a sufficient condition for the Echo State property.

To avoid a necessary but trivial case-by-case analysis we assume both one-dimensional input and output (i.e. $L = K = 1$). Therefore, some matrices become vectors which we denote by small bold letters, i.e. $\boldsymbol{w}^{in}$ instead of $\boldsymbol{W}^{in}$ or $\boldsymbol{w}^{out}$ instead of $\boldsymbol{W}^{out}$. The higher dimensional case can be treated in almost the same way.

We usually will refer to the linear case, $f = \mathrm{id}$, to allow some analysis. Experiments show that improvements derived from the linear case usually also work quite well in the nonlinear case (e.g. $f = \tanh$).

3. Construction

As already mentioned in the introduction, the dynamical reservoir typically is initialized randomly and stays untrained. Jaeger (Jaeger (2002)) suggested the following procedure:

1. Randomly generate an internal weight matrix $\boldsymbol{W}_0$.

2. Normalize $\boldsymbol{W}_0$ to a matrix $\boldsymbol{W}_1$ with unit spectral radius by putting $\boldsymbol{W}_1 = (1/|\rho(\boldsymbol{W}_0)|)\,\boldsymbol{W}_0$, where $|\rho(\boldsymbol{W}_0)|$ is the spectral radius of $\boldsymbol{W}_0$.

3. Scale $\boldsymbol{W}_1$ to $\boldsymbol{W} = \alpha \boldsymbol{W}_1$, where $\alpha < 1$, such that finally $\boldsymbol{W}$ has a spectral radius of α.

The above procedure does not state anything about the singular values. Therefore, the Echo State property is not ensured. Naturally, it arises the question about stable reservoirs. Furthermore, only the absolute value of the maximal eigenvalue is known which induces the question about a design for reservoirs with user-defined properties. One approach to answer these questions was developed in Ozturk et al. (2007). They suggested two different methods but the second is out of focus since it is a bias learning method. The first one concerns the steps 1 and 2 of Jaeger's reservoir construction. Instead choosing of a random matrix $\boldsymbol{W}_0$, the authors generate a matrix with predefined eigenvalue spectrum using the rational canonical form:

$$\boldsymbol{W}_0 := \begin{pmatrix} -a_1 & -a_2 & \cdots & -a_{N-1} & -a_N \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & 0 \\ 0 & \cdots & 0 & 1 & 0 \end{pmatrix} \tag{6}$$

Due to the particular structure of $\boldsymbol{W}_0$, the characteristic polynomial of this matrix has the coefficients contained in the first row of $-\boldsymbol{W}_0$, i.e.

$$p(\lambda) = \det(\lambda \boldsymbol{I} - \boldsymbol{W}_0) = \lambda^N + a_1\lambda^{N-1} + a_2\lambda^{N-2} + \cdots + a_N\lambda^0. \tag{7}$$

Thus, all eigenvalues are (and especially the spectral radius is) determined by the first row of $\boldsymbol{W}_0$. Consequently, a normalization to spectral radius 1 is unnecessary if the a_i are chosen properly. The first row of $\boldsymbol{W}_0$ is calculated by a maximization of the average state entropy through a uniform eigenvalue distribution in the complex plane. A neural network equipped with such a reservoir weight matrix is called *ASE-ESN*.

Our approach is similar: Instead of Jaeger's step 1 and 2, we specify classes of matrices with spectral radius 1 from which $\boldsymbol{W}_0$ is taken.

3.1. Sparse and orthogonal reservoir matrices

Due to Theorem 2, the Echo State property is fulfilled if the singular values are smaller than 1 and the activation function f is Lipschitz continuous with Lipschitz constant 1. The Echo State property guarantees that the outputs of the neural net are basically determined by the sequence of inputs. To calculate the singular values of any matrix $\boldsymbol{M} \in \mathbb{R}^{n\times n}$, we can use its singular value decomposition (SVD): There exist unitary matrices $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{C}^{n\times n}$ such that $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$ where $\boldsymbol{\Sigma}$ is a diagonal matrix containing the singular values on the main diagonal and $*$ denotes the Hermitian transpose. On the other hand, if we start with any diagonal matrix $\boldsymbol{D}$ and multiply it by any orthogonal matrices from left and from right, the singular values remain unchanged. This means, $\boldsymbol{D}$ and $\boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$ have the same singular values because $\boldsymbol{D}^T\boldsymbol{D}$ and $\boldsymbol{V}\boldsymbol{D}^T\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T = \boldsymbol{V}\boldsymbol{D}^T\boldsymbol{D}\boldsymbol{V}^T$ are similar and, hence, have the same eigenvalues. The outline is now as follows: We start with any diagonal matrix with maximal singular value 1 and obtain $\boldsymbol{W}_0$ by consecutive multiplications by orthogonal matrices. Then also $\boldsymbol{W}_0$ has a maximal singular value of 1 and the Echo State property is satisfied if $0 \leq \lambda < 1$ where $\boldsymbol{W} = \lambda\boldsymbol{W}_0$ as in step 3.

Furthermore, we propose to choose all singular values equal to 1. In this case, also the absolute values of all eigenvalues equal 1. Since the identity matrix $\boldsymbol{I}$ satisfies all desired properties, we may start with $\boldsymbol{D} = \boldsymbol{I}$. Self connecting neurons may be not well motivated biologically. In the theory of artificial neural networks, self connecting neurons are even not permitted sometimes (e.g. in Hopfield neural networks, see Hopfield (1982)). Therefore, we first multiply $\boldsymbol{I}$ by any random permutation matrix $\boldsymbol{P}$ to shift nonzero entries away from the main diagonal. Note that $\boldsymbol{P}$ is orthogonal, i.e. the desired spectral properties remain true.

So far, every neuron has exactly one input and one output connection within the reservoir. To generate some more connections, we multiply $\boldsymbol{P}\boldsymbol{I} = \boldsymbol{P}$ by other orthogonal matrices $\boldsymbol{U}$ and $\boldsymbol{V}$. It is computationally important to use sparse weight matrices $\boldsymbol{W}$. Dense matrices consume many floating point operations which slow down the algorithm during the working phase. Unfortunately, random $\boldsymbol{U}$ and $\boldsymbol{V}$ typically generate dense matrices $\boldsymbol{W}_0 = \boldsymbol{U}\boldsymbol{P}\boldsymbol{V}^T$. To avoid too many nonzero entries, we start with $\boldsymbol{W}_1 = \boldsymbol{P}$, and multiply $\boldsymbol{W}_{i-1}$ step-by-step by sparse orthogonal matrices $\boldsymbol{Q}_i$ to generate only few nonzero entries. I.e. $\boldsymbol{W}_i = \boldsymbol{Q}_i \boldsymbol{W}_{i-1}$ or $\boldsymbol{W}_i = \boldsymbol{W}_{i-1}\boldsymbol{Q}_i$. A set of sparse matrices is for example the set of two-dimensional (Givens) rotation matrices $\boldsymbol{Q}(h,k,\varphi)$ defined by

$$\boldsymbol{Q}(h,k,\varphi) := \begin{array}{cc} \left(\begin{array}{ccccccc} \ddots & & & & & & \\ & 1 & & & & & \\ & & \cos(\varphi) & & -\sin(\varphi) & & \\ & & & 1 & & & \\ & & & \ddots & & & \\ & & & & 1 & & \\ & & \sin(\varphi) & & \cos(\varphi) & & \\ & & & & & 1 & \\ & & & & & & \ddots \end{array}\right) & \begin{array}{c} \\ \\ \leftarrow h\text{th row} \\ \\ \\ \\ \leftarrow k\text{th row} \\ \\ \\ \end{array} \end{array} . \quad (8)$$

We stop this procedure if $\boldsymbol{W}_i$ reaches a predefined density and obtain

$$\boldsymbol{W}_{r+l+1} = \boldsymbol{Q}(h_{i_l}, k_{i_l}, \varphi_{i_l}) \cdot \dots \cdot \boldsymbol{Q}(h_{i_1}, k_{i_1}, \varphi_{i_1}) \boldsymbol{P} \boldsymbol{Q}(h_{j_1}, k_{j_1}, \varphi_{j_1}) \cdot \dots \cdot \boldsymbol{Q}(h_{j_r}, k_{j_r}, \varphi_{j_r}). \quad (9)$$

Now, we set $\boldsymbol{W}_0 = \boldsymbol{W}_{r+l+1}$ and continue with step 3 of Jaeger's procedure.

Thus, we obtain sparse and orthogonal reservoir matrices to which we refer as *SORM* in the following. All singular values and the absolute values of the eigenvalues of the SORM are known since they are determined by the construction. At least in the linear case the damping of our RNN is constant and also the stability (Echo State property) is easy to verify.

Algorithm SORM:

1. Permute the rows of I to avoid small cycles. Then the singular values as well as the absolute values of the eigenvalues of the resulting matrix all are 1.
2. Choose some rotation matrix $\boldsymbol{Q}(h,k,\varphi)$ and multiply it from left or from right by the current weight matrix.
3. Repeat step 2 until the desired target density is reached.

3.2. CyclicSORMs

Remember that we assume one input neuron, i.e. the input weight matrix $\boldsymbol{w}^{in}$ is a vector. In section 4 we will show that the matrix

$$\boldsymbol{M} := (\boldsymbol{w}^{in} | \boldsymbol{W}\boldsymbol{w}^{in} | \dots | \boldsymbol{W}^{N-1}\boldsymbol{w}^{in}) \quad (10)$$

has a direct influence on the impact of errors in the following sense: If the condition number of $\boldsymbol{M}$ is very high or the rank of $\boldsymbol{M}$ is low, then the neural net is very susceptible to noise (in the linear case). This means, the columns of $\boldsymbol{M}$ should be linearly independent to guarantee a full rank and, additionally, orthogonal to guarantee a small condition number. A second motivation for a high rank is given by Jaeger in Jaeger (2001b): He proved that the short term memory is maximum if $\boldsymbol{WM}$ is regular which implies necessarily that $\boldsymbol{M}$ is regular. This means $\boldsymbol{M}$ is directly related to the network's ability to remember past inputs.

Note that $\boldsymbol{M} = (\boldsymbol{w}^{in}|\boldsymbol{W}\boldsymbol{w}^{in}|\dots|\boldsymbol{W}^{N-1}\boldsymbol{w}^{in}) = (\boldsymbol{w}^{in}|\lambda\boldsymbol{W}_0\boldsymbol{w}^{in}|\dots|\lambda^{N-1}\boldsymbol{W}_0^{N-1}\boldsymbol{w}^{in})$ and, thus, $\boldsymbol{M}$ can never be orthogonal for $\lambda \neq 1$ and orthogonal $\boldsymbol{W}_0$: Although the columns are mutually orthogonal, they do not have unit length. Therefore, we aim to construct a set of matrices $\boldsymbol{W}_0$ whose elements at least generate an orthogonal matrix $\boldsymbol{M}_0 := (\boldsymbol{w}^{in}|\boldsymbol{W}_0\boldsymbol{w}^{in}|\dots|\boldsymbol{W}_0^{N-1}\boldsymbol{w}^{in})$. The rank of $\boldsymbol{M}$ equals $\operatorname{rank}(\boldsymbol{M}_0)$ and the condition number of $\boldsymbol{M}_0$, $\operatorname{cond}(\boldsymbol{M}_0)$, is $\frac{1}{\lambda^{N-1}}\operatorname{cond}(\boldsymbol{M})$. Since $\boldsymbol{w}^{in}$ is part of $\boldsymbol{M}_0$, our investigations shall include also these input weights.

The construction of section 3.1 yields $\boldsymbol{W}_0 = \boldsymbol{UPV}^T$ (whereas $\boldsymbol{U} = \boldsymbol{Q}(h_{i_l}, k_{i_l}, \varphi_{i_l}) \cdot \dots \cdot \boldsymbol{Q}(h_{i_1}, k_{i_1}, \varphi_{i_1})$ and $\boldsymbol{V}^T = \boldsymbol{Q}(h_{j_1}, k_{j_1}, \varphi_{j_1}) \cdot \dots \cdot \boldsymbol{Q}(h_{j_r}, k_{j_r}, \varphi_{j_r})$). We denote the i-th column of the identity matrix by $\boldsymbol{e}_i$.

Lemma 3. If $\boldsymbol{U} = \boldsymbol{V}$, $\boldsymbol{w}^{in} = \boldsymbol{V}\boldsymbol{e}_1$, and the permutation related to $\boldsymbol{P}$ is of cycle length N, then $\boldsymbol{M}_0$ is orthogonal.

Proof. Assume $\boldsymbol{U} := \boldsymbol{V}$. The resulting matrix $\boldsymbol{W}_0 = \boldsymbol{VPV}^T$ is similar to the permutation matrix $\boldsymbol{P}$ (i.e. both matrices have the same eigenvalues). Now, $\boldsymbol{W}_0^k\boldsymbol{w}^{in}$ is orthogonal to $\boldsymbol{W}_0^j\boldsymbol{w}^{in}$ for $k \neq j \in \{0, \dots, N-1\}$ if and only if $(\boldsymbol{W}_0^k\boldsymbol{w}^{in})^T\boldsymbol{W}_0^j\boldsymbol{w}^{in} = 0$. Since $\boldsymbol{w}^{in}$ is the first column of the orthogonal matrix $\boldsymbol{V}$,

$$(\boldsymbol{W}_0^k\boldsymbol{w}^{in})^T\boldsymbol{W}_0^j\boldsymbol{w}^{in} = \boldsymbol{e}_1^T\boldsymbol{V}^T(\boldsymbol{VP}^T\boldsymbol{V}^T)^k(\boldsymbol{VPV}^T)^j\boldsymbol{V}\boldsymbol{e}_1 = \boldsymbol{e}_1^T(\boldsymbol{P}^k)^T\boldsymbol{P}^j\boldsymbol{e}_1 = \boldsymbol{e}_1^T\boldsymbol{P}^{j-k}\boldsymbol{e}_1. \quad (11)$$

Denote the permutation corresponding to $\boldsymbol{P}$ by π. Then $\boldsymbol{P}\boldsymbol{e}_1 = \boldsymbol{e}_{\pi(1)}$. Therefore, $\boldsymbol{e}_1^T\boldsymbol{P}^{j-k}\boldsymbol{e}_1 = 0$ is satisfied for any $k \neq j \in \{0, \dots, N-1\}$ iff π is a cycle of length N (i.e. $\boldsymbol{P}^i\boldsymbol{e}_1 \neq \boldsymbol{e}_1$ for any $i \in \{1, \dots, N-1\}$). Consequently, $\boldsymbol{M}_0$ is orthogonal for this special choice of $\boldsymbol{W}_0$ and $\boldsymbol{w}^{in}$ iff the cycle length of $\boldsymbol{P}$ is N. $\square$

The properties of $\boldsymbol{W}_0 = \boldsymbol{VPV}^T$ are well known. E.g. the weight matrix $\boldsymbol{W}_0$ is similar to $\boldsymbol{P}$, i.e. the characteristic polynomial, $p(x)$, equals that of $\boldsymbol{P}$, $p(x) = x^N - 1$. (To show this just use the Laplace expansion to calculate $\det(x\boldsymbol{I} - \boldsymbol{P})$.) This means the eigenvalues are uniformly distributed on the complex unit circle.

Afterwards, we again scale the spectral radius to the desired value. The above procedure yields sparse and orthogonal matrices like in subsection 3.1. Additionally, the columns of the above matrices go through the orthogonal subspaces in a cyclic manner if $\boldsymbol{w}^{in} = \boldsymbol{V}\boldsymbol{e}_1$. Under these conditions the reservoir saves the last N inputs $u(n)$ in disjoint orthogonal subspaces of $\mathbb{R}^N$. Only inputs older than $N-1$ time steps do not have an own subspace and superpose newer inputs. The matrices provide a maximum short term memory since they produce less superpositions of inputs than random matrices do. Mainly for the experimental section, we abbreviate the notation and call neural networks equipped with reservoirs based on the above procedure *CyclicSORMs*.

```
Algorithm CyclicSORM:
    1. Choose a permutation $\pi$ of cycle length $N$. The related matrix is denoted by $\boldsymbol{P}$.
    2. Choose a sparse and orthogonal transformation matrix $\boldsymbol{V}$ as in subsection 3.1.
    3. Set $\boldsymbol{W}_0 := \boldsymbol{V}\boldsymbol{P}\boldsymbol{V}^T$.
```

For all theoretical investigations, we assume a linear activation function f. We will show later that under this assumption, $\boldsymbol{w}^{in} = \boldsymbol{V}\boldsymbol{e}_1$ yields a well conditioned correlation matrix of the network states. Using a linear combination of different columns of $\boldsymbol{V}$ does not make sense because no additional dynamic is introduced. This linear combination can also be done within the readout layer leading to the same outcome. The only (unwanted) effect is a reduction of the memory capacity.

However, for nonlinear activation functions ($f = \tanh$), there it could be (there are, as we will see later) additional dynamics if $\boldsymbol{w}^{in}$ is such a linear combination of different columns of $\boldsymbol{V}$. Therefore, inspired by the above analysis for $\boldsymbol{w}^{in}$, we will practically use a weighted sum of different columns of $\boldsymbol{V}$ as input weights for the experiments.

3.3. RingOfNeurons and ChainOfNeurons

Taking a closer look at the CyclicSORMs, we discover an even simpler system which has the same properties in the linear case. Again, let us assume linear activation functions $f = \mathrm{id}$ and $f^{out} = \mathrm{id}$ for this section. Then the current activations can be calculated as

$$\boldsymbol{x}(n) = \boldsymbol{W}\boldsymbol{x}(n-1) + \boldsymbol{w}^{in}u(n), \tag{12}$$

$$\boldsymbol{y}(n) = \boldsymbol{w}^{out}\boldsymbol{x}(n). \tag{13}$$

For $\boldsymbol{W}$ and $\boldsymbol{w}^{in}$ generated by the construction of subsection 3.2, we obtain

$$\boldsymbol{x}(n) = (\lambda\boldsymbol{V}\boldsymbol{P}\boldsymbol{V}^T)\boldsymbol{x}(n-1) + \boldsymbol{V}\boldsymbol{e}_1u(n) \tag{14}$$

and

$$\boldsymbol{V}^T\boldsymbol{x}(n) = \lambda\boldsymbol{P}\boldsymbol{V}^T\boldsymbol{x}(n-1) + \boldsymbol{e}_1u(n). \tag{15}$$

Considering $\boldsymbol{x}(n)$ and $\widehat{\boldsymbol{x}}(n) = \boldsymbol{V}^T\boldsymbol{x}(n)$ as states of the same but rotated dynamical system, we can also work with the simpler version $\widehat{\boldsymbol{x}}$ generated by the network update

$$\widehat{\boldsymbol{x}}(n) = \lambda\boldsymbol{P}\widehat{\boldsymbol{x}}(n-1) + \boldsymbol{e}_1u(n) \tag{16}$$

i.e. with the very simple reservoir matrix $\boldsymbol{P}$ together with input weights $\boldsymbol{e}_1$. It produces the same outputs $y(n)$ as before if $y(n) = \boldsymbol{w}^{out}\boldsymbol{x}(n) = \boldsymbol{w}^{out}\boldsymbol{V}\widehat{\boldsymbol{x}}(n)$ i.e. with appropriately modified output weights $\widehat{\boldsymbol{w}}^{out} = \boldsymbol{w}^{out}\boldsymbol{V}$.

According to the permutation matrix $\boldsymbol{P}$, the internal units of the simplified network are connected in a cyclic way. The units can be relabeled such that without loss of generality the reservoir matrix is

$$\widehat{\boldsymbol{W}} := \lambda \begin{pmatrix} 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & 0 \\ 0 & \dots & 0 & 1 & 0 \end{pmatrix}. \tag{17}$$

The neural net with this matrix $\widehat{\boldsymbol{W}}$ is called *RingOfNeurons*.

Now, $x_1(n) = \sum_{i=0}^{\infty} \lambda^{iN} u(n - iN)$ while all other activations $x_j(n)$ are independent of $u(n - iN)$ for all $i \in \mathbb{N}$. This means, neither $u(n)$ nor former inputs can be reproduced without error except for the trivial case that $u(n) = \alpha u(n - N)$ for some α. So, the activation $x_1(n)$ is an uncontrolled superposition of the inputs $u(n - iN)$ $(i \in \mathbb{N})$ which we would like to avoid by forbidding the connection from the last internal neuron to the first one ($\widehat{w}_{1,N} = 0$). It remains

$$\widehat{\boldsymbol{W}} := \lambda \begin{pmatrix} 0 & 0 & \dots & 0 & 0 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & 0 \\ 0 & \dots & 0 & 1 & 0 \end{pmatrix}. \tag{18}$$

The above matrix $\widehat{\boldsymbol{W}}$ is nilpotent and has only zero eigenvalues. Such a dynamical reservoir acts like a FIFO-memory. After N steps the system forgets the input. We will refer to this reservoir as a *ChainOfNeurons*.

Because of the same motivation as at the end of section 3.2, for practical experiments, we will use linear combinations of the columns of the identity matrix $\boldsymbol{I}$ as input weights in section 5.

3.4. Related publications

The idea of using orthogonal matrices was already reported in Jaeger (2001b). However, the construction of the matrix was different and without control of the matrix density. It was shown there experimentally that the short term memory of reservoirs equipped with orthogonal matrices is almost maximum.

A theoretical analysis of the short term memory of related reservoir matrices in linear networks was done by White et al. in White et al. (2004). The authors investigated the memory of random orthogonal matrices and distributed shift register networks. The latter are strongly related to the CyclicSORM, the RingOfNeurons and the ChainOfNeurons. Actually, the ChainOfNeurons is a distributed shift register network. The difference to the other kinds of reservoir weights is that a distributed shift register depends only on the last N inputs. The authors give precise formulas to calculate the memory capacity of these networks.

Reservoirs similar to the RingOfNeurons and the ChainOfNeurons were also introduced in Čerňanský and Tiňo (2008) and Rodan and Tiňo (2011) where the major intention was a simplification of the network complexity and the authors concluded that for some tasks the above reservoirs work as well as random reservoirs. The first article investigates full input connections as well as $\boldsymbol{w}^{in} = \alpha \boldsymbol{e}_1$, The second one analyzes an even more simplified connection method where all input weights are constant and only the sign varies. In the latter article, the authors prove that, under certain conditions, the RingOfNeurons can achieve a memory capacity arbitrarily close to N.

4. Estimation of errors

The states of the dynamical reservoir are typically noisy. Sources of noise are for example perturbed input data, user forced noise to avoid overfitting or to regularize the training equation (ridge regression). But noisy activations lead to perturbations of the training, to incorrect output weights, and, finally, to a larger output error. One would expect that the error highly depends on the rank and the singular values of the matrix $\boldsymbol{M} := (\boldsymbol{w}^{in}|\boldsymbol{W}\boldsymbol{w}^{in}|\dots|\boldsymbol{W}^{N-1}\boldsymbol{w}^{in})$. This is exactly what we will show on the next pages. Furthermore, we will prove that for linear activation functions, the ChainOfNeurons' matrix $\boldsymbol{M}$ has optimal singular values. Randomly created weight matrices usually yield ill-conditioned or even singular $\boldsymbol{M}$ and, thus, are more susceptible to noise. This also motivates to prefer the ChainOfNeurons architecture compared to random matrices.

We store all reservoir states row-by-row in the matrix $\widehat{\boldsymbol{S}}$. Analogously, the targets are saved in $\boldsymbol{t}$. Assuming the linear case, Jaeger (2001b) proved that the memory capacity is bounded by N and that MC_k (which is a measure for the memory of the input k time steps before) is decreasing monotonically. This was shown experimentally to be valid also in the nonlinear case (see e.g. section 5.1). Consequently, if one wants to reproduce the input $u(n-N)$ entered N time steps before, this implicates two facts: Firstly, since MC is bounded by N, $u(n-k)$ $(k = 0, \dots, N-1)$ is regenerated with errors. Secondly, since MC_k is monotonically decreasing, the ESN also cannot reproduce $u(n-N)$ without error. Therefore, we assume that the exact state $\boldsymbol{x}(n)$ depends only on the last N inputs to guarantee an optimal - noiseless linear dependence of $\boldsymbol{y}(n)$ on the previous N inputs. The exact matrix of the reservoir activations is denoted by $\boldsymbol{S}$, and $\boldsymbol{S}^+$ is its pseudoinverse. We denote the perturbation by $\boldsymbol{\Delta S}$, i.e. $\widehat{\boldsymbol{S}} = \boldsymbol{S} + \boldsymbol{\Delta S}$. Analogously, let $\widehat{\boldsymbol{w}}^{out}$ be the solution of

$$\|\widehat{\boldsymbol{S}}(\widehat{\boldsymbol{w}}^{out})^T - \boldsymbol{t}\| \to \min\,, \tag{19}$$

whereas $\boldsymbol{w}^{out} = \boldsymbol{t}^T(\boldsymbol{S}^+)^T$ is the exact output weight vector and $\boldsymbol{\Delta w}^{out}$ denotes the difference $\widehat{\boldsymbol{w}}^{out} - \boldsymbol{w}^{out}$. Let $\boldsymbol{I}_p$ be the $p \times p$ identity matrix and assume for this section that $\|\cdot\|$ is the spectral norm. The proofs of the following theorems and lemmas are given in the appendix.

Theorem 4 (First order approximation of $\boldsymbol{\Delta w}^{out}$)**.** For sufficiently small $\|\boldsymbol{\Delta S}\|$,

$$\begin{aligned}\boldsymbol{\Delta w}^{out} \approx & \boldsymbol{t}^T\Big[(\boldsymbol{S}+\boldsymbol{\Delta S})\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*\left(\boldsymbol{V}\boldsymbol{J}_l\boldsymbol{V}^*\boldsymbol{\Delta S}^T\boldsymbol{S}\right)^+ + \boldsymbol{\Delta S}\boldsymbol{V}\boldsymbol{J}_l\boldsymbol{V}^*\left(\boldsymbol{S}^T\boldsymbol{\Delta S}\boldsymbol{V}\boldsymbol{J}_l\boldsymbol{V}^*\right)^+\Big]+ \\ & + \boldsymbol{r}^T\boldsymbol{\Delta S}(\boldsymbol{S}^T\boldsymbol{S})^+ - \boldsymbol{w}^{out}\boldsymbol{\Delta S}^T(\boldsymbol{S}^+)^T \end{aligned} \tag{20}$$

is a first order approximation of the disturbance of $\boldsymbol{w}^{out}$, where $\boldsymbol{r} := \boldsymbol{t} - \boldsymbol{S}(\boldsymbol{w}^{out})^T$ is the residuum and $\boldsymbol{V}$ is the orthogonal matrix from the SVD $\boldsymbol{S} = \boldsymbol{U}\begin{pmatrix}\boldsymbol{\Sigma} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{pmatrix}\boldsymbol{V}^*$. Let $\boldsymbol{J}_l = \begin{pmatrix}\boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{I}_{\bar{s}}\end{pmatrix}$ and $\boldsymbol{J}_u = \begin{pmatrix}\boldsymbol{I}_s & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{pmatrix}$ be the diagonal matrices with $\bar{s} = N - \operatorname{rank}(\boldsymbol{S})$ one entries in the lower right or $s = \operatorname{rank}(\boldsymbol{S})$ entries in the upper left corner, respectively.

Lemma 5. Let $f = \mathrm{id}$. If $\operatorname{rank}(\boldsymbol{M}) = N$ and at least N nonzero inputs are entered, then

$$\boldsymbol{J}_l = 0. \tag{21}$$

On the following pages, we aim to estimate the ratio $\frac{\|\boldsymbol{\Delta w}^{out}\|}{\|\boldsymbol{w}^{out}\|}$ and decrease this approximation afterwards to guarantee a robust training. We estimate this ratio for small perturbations $\|\boldsymbol{\Delta S}\|$ using Theorem 4:

$$\begin{aligned}\frac{\|\boldsymbol{\Delta w}^{out}\|}{\|\boldsymbol{w}^{out}\|} \approx & \frac{1}{\|\boldsymbol{w}^{out}\|}\Big(\Big\|\boldsymbol{r}^T\boldsymbol{\Delta S}(\boldsymbol{S}^T\boldsymbol{S})^+ - \boldsymbol{w}^{out}\boldsymbol{\Delta S}^T(\boldsymbol{S}^+)^T + \\ & + \boldsymbol{t}^T\Big[(\boldsymbol{S}+\boldsymbol{\Delta S})\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*\left(\boldsymbol{V}\boldsymbol{J}_l\boldsymbol{V}^*\boldsymbol{\Delta S}^T\boldsymbol{S}\right)^+ + \boldsymbol{\Delta S}\boldsymbol{V}\boldsymbol{J}_l\boldsymbol{V}^*\left(\boldsymbol{S}^T\boldsymbol{\Delta S}\boldsymbol{V}\boldsymbol{J}_l\boldsymbol{V}^*\right)^+\Big]\Big\|\Big) \end{aligned} \tag{22}$$

$$\begin{aligned}\leq & \frac{\|\boldsymbol{r}\|}{\|\boldsymbol{w}^{out}\boldsymbol{S}^T\|}\frac{\|\boldsymbol{\Delta S}\|}{\|\boldsymbol{S}\|}\|(\boldsymbol{S}^T\boldsymbol{S})^+\|\|\boldsymbol{S}\|^2 + \frac{\|\boldsymbol{\Delta S}\|}{\|\boldsymbol{S}\|}\|\boldsymbol{S}^+\|\|\boldsymbol{S}\| + \\ & + \frac{\Big\|\boldsymbol{t}^T\Big[(\boldsymbol{S}+\boldsymbol{\Delta S})\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*\left(\boldsymbol{V}\boldsymbol{J}_l\boldsymbol{V}^*\boldsymbol{\Delta S}^T\boldsymbol{S}\right)^+ + \boldsymbol{\Delta S}\boldsymbol{V}\boldsymbol{J}_l\boldsymbol{V}^*\left(\boldsymbol{S}^T\boldsymbol{\Delta S}\boldsymbol{V}\boldsymbol{J}_l\boldsymbol{V}^*\right)^+\Big]\Big\|}{\|\boldsymbol{w}^{out}\|} \end{aligned} \tag{23}$$

If $\operatorname{rank}(\boldsymbol{M}) = N$ and $f = \mathrm{id}$ the application of Lemma 5 yields

$$\frac{\|\boldsymbol{\Delta w}^{out}\|}{\|\boldsymbol{w}^{out}\|} \lesssim \frac{\|\boldsymbol{r}\|}{\|\boldsymbol{w}^{out}\boldsymbol{S}^T\|}\frac{\|\boldsymbol{\Delta S}\|}{\|\boldsymbol{S}\|}\operatorname{cond}(\boldsymbol{S})^2 + \frac{\|\boldsymbol{\Delta S}\|}{\|\boldsymbol{S}\|}\operatorname{cond}(\boldsymbol{S}). \tag{24}$$

To decrease this bound, we decrease $\operatorname{cond}(\boldsymbol{S})$ assuming that the targets can be well approximated by the network such that $\|\boldsymbol{r}\|/\|\boldsymbol{w}^{out}\boldsymbol{S}^T\|$ is small.

Lemma 6. Let $f = \mathrm{id}$, θ be the training size, and the inputs $u(i)$ be independent realizations of the random variable u with zero expectation and variance $\mathbb{V}(u)$. Then for

$\theta \to \infty$, the singular values of $\boldsymbol{S}$ asymptotically equal those of $\sqrt{\theta\, \mathbb{V}(u)}\boldsymbol{M}$. Consequently, the matrices $\boldsymbol{S}$ and $\boldsymbol{M}$ have asymptotically equal condition numbers.

Remark 7. A crucial assumption of the above Lemma 6 is the identical and independent distribution of the inputs $u(n)$. Typically, this is not satisfied in practical situations, but it is necessary to allow our linear algebraic investigation. However, in section 5.3 we also perform an experiment which does not satisfy this assumption but the results remain comparable to experiments fulfilling it. But of course there may be situations for which our assertions do not hold at all.

Due to the above lemma, we can minimize the condition number of $\boldsymbol{M}$ instead of $\boldsymbol{S}$ for sufficiently large training sizes. Obviously, $\text{cond}(\boldsymbol{M}) = \frac{\sigma_{max}(\boldsymbol{M})}{\sigma_{min}(\boldsymbol{M})} \geq 1$. Here, the minimum 1 can be achieved by matrices $\boldsymbol{M}$ for which holds:

$$\forall \boldsymbol{v} \in \mathbb{R}^N : \ \|\boldsymbol{M}\boldsymbol{v}\| = \rho(\boldsymbol{M})\|\boldsymbol{v}\|. \tag{25}$$

Below, we assume $\rho(\boldsymbol{M}) = 1$ keeping in mind that we also could scale $\boldsymbol{M}$ by any factor. Since the above equation holds for any $\boldsymbol{v} \in \mathbb{R}^N$, $\boldsymbol{M}$ is an orthogonal matrix. Multiplying $\boldsymbol{W}$ by $\boldsymbol{M}$ from left and from right yields a surprisingly simple matrix:

$$\begin{aligned} \boldsymbol{M}^T\boldsymbol{W}\boldsymbol{M} &= \boldsymbol{M}^T(\boldsymbol{W}\boldsymbol{w}^{in}|\boldsymbol{W}^2\boldsymbol{w}^{in}|\dots|\boldsymbol{W}^N\boldsymbol{w}^{in}) \\ &= \begin{pmatrix} 0 & 0 & \dots & 0 & * \\ 1 & 0 & \dots & 0 & * \\ 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & * \\ 0 & \dots & 0 & 1 & * \end{pmatrix} =: \boldsymbol{Q} \end{aligned} \tag{26}$$

For later use, note that $(\boldsymbol{W}\boldsymbol{w}^{in}|\boldsymbol{W}^2\boldsymbol{w}^{in}|\dots|\boldsymbol{W}^N\boldsymbol{w}^{in})$ contains $N-1$ columns of $\boldsymbol{M}$. The crucial consequence of (26) is that, in order to get a regular matrix $\boldsymbol{M}$ with minimum condition number, $\boldsymbol{W}$ has to be an orthogonal transformation of a matrix of type $\boldsymbol{Q}$. Along the same line, the first column of the transformation matrix contains the appropriate, optimal input weights $\boldsymbol{w}^{in} = \boldsymbol{M}\boldsymbol{e}_1$.

This means, for any orthogonal matrix $\boldsymbol{V}$, $\boldsymbol{W} = \boldsymbol{V}\boldsymbol{Q}\boldsymbol{V}^T$ as above, and $\boldsymbol{w}^{in} = \boldsymbol{V}\boldsymbol{e}_1$, the rank of $\boldsymbol{M}$ is N provided that also nonzero inputs are entered in the network for at least N time steps.

Finally, let us take a closer look at $\boldsymbol{\Delta S}$. We assumed that the exact states just depend on the previous N inputs whereas earlier influences are considered to be part of $\boldsymbol{\Delta S}$. But the calculated states $\widehat{\boldsymbol{x}}$ strongly depend on even earlier inputs for those $\boldsymbol{Q}$ with a big norm of the last column. To see this, let $\boldsymbol{\delta}(n) = \widehat{\boldsymbol{x}}(n) - \sum_{i=0}^{N-1} \boldsymbol{W}^i\boldsymbol{e}_1 u(n-i) + \boldsymbol{W}^N\boldsymbol{x}(n-N)$ be the external noise. Then $\|\boldsymbol{\Delta x}(n)\| \leq \|\boldsymbol{\delta}(n)\| + \|\boldsymbol{W}^N\|$. A first step to minimize $\|\boldsymbol{\Delta x}(n)\|$ is to decrease $\|\boldsymbol{W}^N\| = \|\boldsymbol{Q}^N\| \geq \|\boldsymbol{Q}\boldsymbol{e}_N\|$ since $\boldsymbol{Q}\boldsymbol{e}_N$ appears as the first column of $\boldsymbol{Q}^N$. A small norm is therefore preferable and choosing the last column to be $\boldsymbol{e}_1$ appears to be reasonable. Note that this construction then corresponds to CyclicSORM.

If even $\boldsymbol{Q}\boldsymbol{e}_N = \boldsymbol{0}$, then we finally arrived at the ChainOfNeurons model. It can provide robust weights which yield a small error, which has been derived and shown to be optimal even theoretically at least for a linear activation function and fairly well-suited taskes (e.g. with $\|\boldsymbol{r}\| \leq \|\boldsymbol{S}\boldsymbol{w}^{out}\|$).

In this linear case, it is easy to see that including further ChainOfNeurons with different weights does not provide any further improvement because the corresponding states are linearly dependent. However, this is different in the nonlinear case. If $\operatorname{rank}(\boldsymbol{S}) = N$, equation (24) holds also in the nonlinear case. Here, two or more separate chains (with different weights of course) may decrease $\boldsymbol{r}$. Due to the nonlinear activation function, the states are not linearly dependent and thus the rank of $\boldsymbol{S}$ does not decrease. Apparently, it has almost the same effect if, instead of injecting the input only to the first hidden neuron, we connect input and hidden layer sparsely with different weights, i.e. in the nonlinear case, we apply other input weights than those derived in the earlier sections 3.2 and 3.3. Although our experiments seem to confirm this proposition, corresponding theoretical investigations remain to be done for nonlinear activation functions.

At least for the for the ChainOfNeurons with $\boldsymbol{w}^{in} = \alpha \boldsymbol{e}_1$, similar results can be derived for the nonlinear activation function ($f = \tanh$). Since Theorem 4 is independent of the activation function, the first order approximation hold also in the nonlinear case for any reservoir type (only $f^{out} = id$ is crucial). Also the Lemma 5 is valid for nonlinear f using the ChainOfNeurons. Lemma 6 holds in a slightly modified version for nonlinear f and the ChainOfNeurons. Let $\rho(\boldsymbol{W})$ be the spectral radius of $\boldsymbol{W}$. Then Lemma 6 holds also under the current assumptions by substituting $\boldsymbol{M}$ by

$$\boldsymbol{M}_{nl} := \Big(\mathbb{E}[\phi_0(\alpha u)]\boldsymbol{e}_1 \Big| \mathbb{E}[(\phi_1(\alpha u)]\boldsymbol{e}_2 \Big| \dots \Big| \mathbb{E}[\phi_{N-1}(\alpha u)]\boldsymbol{e}_N \Big),$$

where $\phi_n \equiv (\tanh \circ \rho(\boldsymbol{W}))^n \circ \tanh$. This means, if $\mathbb{E}[\phi_0(\alpha u)]/\mathbb{E}[\phi_{N-1}(\alpha u)]$ is small, also the relative error $\|\boldsymbol{\Delta w}^{out}\|/\|\boldsymbol{w}^{out}\|$ will be small.

5. Experiments

In this section, we compare the error rates of ESNs equipped with one of

- StandardMat - the original randomly generated reservoirs
- SORM
- CyclicSORM
- RingOfNeurons
- ChainOfNeurons.

All results of the following experiments are reported in tables with the following captions:

- ρ - spectral radius, or constant connection weight along the ChainOfNeurons, respectively
- N - number of reservoir neurons

- $RDens$ - connection density of the reservoir
- $FScale$ - feedback scale: bound for weights of output-hidden connections
- $FDens$ - feedback density: portion of used output-hidden connections
- $IScale$ - input scale: bound for weights of input-hidden connections (see below)
- $IDens$ - connection density from the input neurons into the reservoir; $\times$ - means only first input unit/vector will be used

All experiments (except for the first one in 5.1) were accomplished both with the Oger (OrGanic Environment for Reservoir computing) toolbox and with the ANN toolbox of PLANET intelligent systems GmbH. We optimize the parameter setup using the convenient gridsearch functionality of Oger. All these optimizations are organized in the same way. We provide a number of training sets, including input and target data. Afterwards, we take two samples from these data sets. The first is the training set and the second is used for validation. The plots are generated by fixing all but one parameter of the resulting error arrays.

We test different input strategies. The first strategy is the classical random connection of input and hidden layer. We set a predefined percentage of the connections to uniformly distributed random values from $[-IScale, IScale]$. The second way of connecting input and hidden neurons is to distribute the input connections equally on the neurons, e.g., if we use a density of $IDens = 0.25$, only those neurons with index $\equiv 1 \pmod 4$ are connected with the input. The input weights of the CyclicSORM additionally are rotated afterwards by the matrix $\boldsymbol{V}$ of the Algorithm CyclicSORM. Again, the weight strengths are uniformly distributed from the interval $[-IScale, IScale]$.

5.1. Delayline

The first experiment of this section is the well known delayline experiment. The neural net gets random inputs (uniformly distributed over $[-0.5, 0.5]$), and the task is to reproduce those inputs after a given time delay at the output of the neural net. We refer to Jaeger (2001a,b, 2002) for details. We only used PLANET's ANN toolbox for this experiment.

We have tested all of the considered matrices and improved the parameters roughly for every class of matrices separately by hand. The most important values are given in table 1. The experiment was processed 10.000 steps with a washout of 200 steps, we have measured the mean squared error (MSE) over the last 100 steps of the experiment. The output activation function is the identity function and the activation of the hidden units is tanh. We use ($N =$) 50 internal neurons. For each system, the results are averaged over 50 different runs (50 different randomly generated systems and workflows). CyclicSORM, SORM and StandardMat have a connection density of $RDens = 0.1$.

In Figure 2 the error is plotted against the delay interval. The slope of the pink curve, which represents the results of SORM, is smoother than the one of the light blue curve achieved by StandardMat. The random reservoir produces already at a delay interval of 30 an output signal which is almost uncorrelated to the target signal. At this point

	ρ	$IScale$	$IDens$
ChainOfNeurons	0.95	1.0	$\times$
RingOfNeurons	0.95	1.0	$\times$
CyclicSORM	0.95	1.0	$\times$
SORM	0.95	1.0	0.1
StandardMat	0.99	1.0	0.1

Table 1: Best parameters for the delayline experiment optimized by hand.

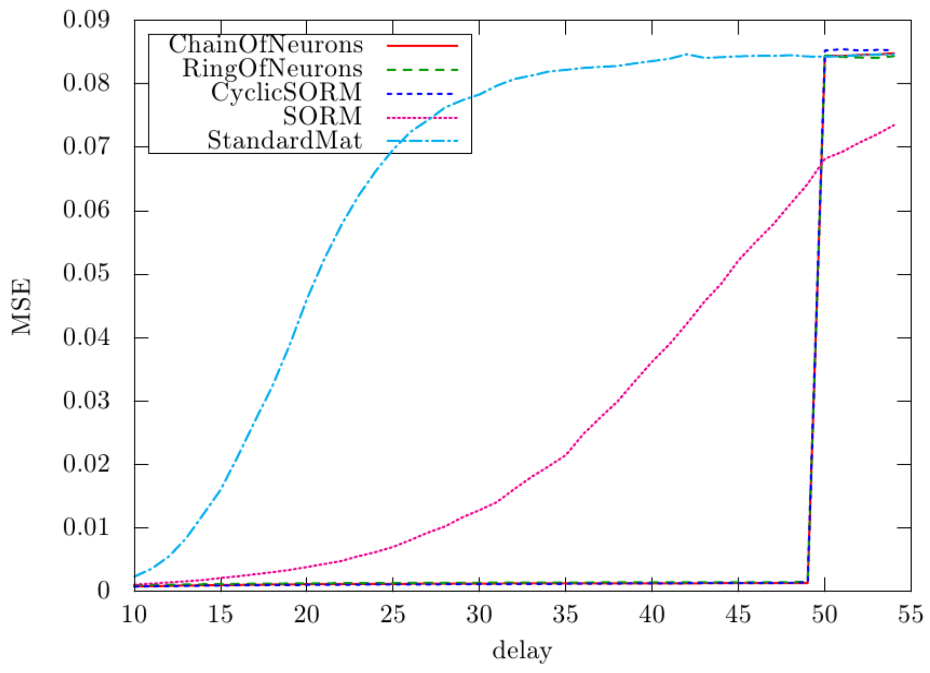

Figure 2: Delayline experiment executed with different reservoir types and averaged over 50 different initializations

the SORM reservoir still yields good results. The curves of the CyclicSORM and the RingOfNeurons show a good short term memory until a delay interval of $N-1=49$.

In section 4, we proved the relation of $\boldsymbol{M}$ to perturbations. The greater the rank and the smaller the condition number of the related matrix, the smaller is the impact of disturbances on the system predictions in the linear case. In fact, the more precise bound substitutes cond($\boldsymbol{M}$) by $\|\boldsymbol{M}\|\,\|\boldsymbol{M}^+\|$. Instead of rank($\boldsymbol{M}$) and $\|\boldsymbol{M}\|\,\|\boldsymbol{M}^+\|$, we can use rank($\boldsymbol{S}^T\boldsymbol{S}$) and $\|\boldsymbol{S}^T\boldsymbol{S}\|\,\|(\boldsymbol{S}^T\boldsymbol{S})^+\|$ as an approximation if the training size θ is large. Table 2 shows these values for the matrices used in this experiment. As expected, the ChainOfNeurons, the RingOfNeurons, and the CyclicSORM yield the best values. This coincides with the measured errors of the experiment where these three matrices also yield a good performance.

The delayline experiment shows that the newly introduced weight matrices have a good

	rank($\boldsymbol{S}^T\boldsymbol{S}$)	$\|\boldsymbol{S}^T\boldsymbol{S}\| \, \|(\boldsymbol{S}^T\boldsymbol{S})^+\|$
ChainOfNeurons	50.0	12.34648
RingOfNeurons	50.0	12.34648
CyclicSORM	50.0	12.34648
SORM	49.86	1584.37109
StandardMat	48.68	$2.53285 \cdot 10^{13}$

Table 2: Averaged rank of $\boldsymbol{S}^T\boldsymbol{S}$ and $\|\boldsymbol{S}^T\boldsymbol{S}\| \, \|(\boldsymbol{S}^T\boldsymbol{S})^+\|$ on average of 50 runs

short term memory. The CyclicSORM, the RingOfNeurons, and the ChainOfNeurons provide the best memory and work almost flawless until a delay of $N - 1$ time steps. Although the error of the SORM is soon greater than the error of these matrices, it is significantly lower than the error obtained by the randomly created matrices. This means for tasks requiring a high short term memory one should choose any of the CyclicSORM, the RingOfNeurons, or the ChainOfNeurons. Interestingly, it seems that networks equipped with SORMs have an (blurred) memory of what happened earlier than $N - 1$ time steps before.

5.2. Pattern detection

The delayline experiment is a simple recognition task to test the short term memory of the neural net. As seen above, the introduced matrices provide a better short term memory. The next experiment shows that the newly introduced matrices are also capable of distinguishing (with a higher accuracy) between a known pattern (six consecutive values $p_1, \ldots, p_6 \in [-0.5, 0.5]$) and a random sequence. The input is a random sequence merged with this pattern. The intervals between two pattern are of random length. The task is to identify this pattern within the random sequence (for details see Jaeger (2001b)). This is obviously not a short term memory test since the important inputs are entered just within the last few steps.

The data sets contain 10000 elements plus 200 additional washout data points. We use one data set for training and another for validation. The neural net contains one input, one output and 50 hidden neurons. The grid of the search is four-dimensional containing the bias scaling value of the reservoir, the input connection density, the scaling value for the input and reservoir weights. The best values found by this search are contained in Table 3. We say the network has detected a pattern if the output neuron has an activation greater than a certain threshold. This threshold is set optimally and separately for each matrix by minimizing the sum of false positives and false negatives *after the experiment*. For any parameter setup, the experiments are executed with 10 different reservoirs and 6 different data sets each.

Using a linear activation function $f = \text{id}$ increases the error rates because of the following important feature of the nonlinearity: Assume q is part of the pattern (neither 0 nor ± 0.5), and assume further two hidden neurons which are not connected. Their activation in each step is defined as $\tanh(w_1^{in} x)$ and $\tanh(w_2^{in} x)$ if x is the current input.

The output activation y (a linear combination of the activations of the hidden neurons $y = \alpha \tanh(w_1 x) + \beta \tanh(w_2 x)$) should increase if $x = q$ since q is part of the pattern and decrease else. Figure 3 shows the activation of the output neuron for fixed $\boldsymbol{w}^{out}$, w_1^{in} and w_2^{in}. There is a maximum within the input interval which can be adjusted by changing w_1^{in} and w_2^{in} and learning $\boldsymbol{w}^{out}$ such that the maximum is near q. I.e. saving the input within the reservoir several times with different input scalings improves the detection of pattern. This is the reason why the ChainOfNeurons can detect values far away from the boundary. To support this feature, we also use the second input strategy (distribute the input weights equally) already mentioned above.

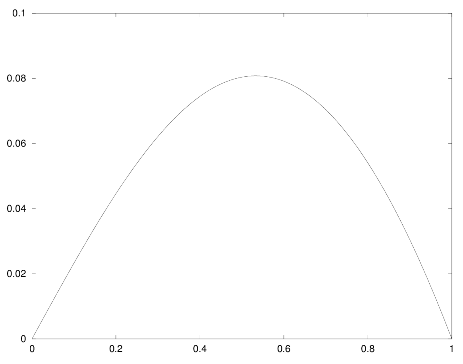

Figure 3: Output $y = \alpha \cdot \tanh(w_1^{in} \cdot x) - \beta \cdot \tanh(w_2^{in} \cdot x)$ for $x \in [0, 1]$ whereas $\alpha = 1$, $\beta = \tanh(1)/\tanh(0.1)$, $w_1^{in} = 1$ and $w_2^{in} = 0.1$. The maximum is between 0.5 and 0.6.

	bias	ρ	$IScale$	$IDens$
ChainOfNeurons	0.0	0.1	0.8	1/7
RingOfNeurons	0.0	0.1	1.0	1/9
CyclicSORM	0.0	0.15	0.9	1/9
SORM	0.0	0.1	1.0	1.0
StandardMat	0.0	0.05	1.0	1.0

Table 3: Best parameter setup for the pattern detection experiment found by gridsearch.

Figure 5 shows the number of false detections against the spectral radius. We take the average over 100 different initializations. The minimal number of false detections using StandardMat (random reservoirs) is clearly larger than the minimal number of those neural nets equipped with SORMs. Again, we obtain the best results by using RingOfNeurons or CyclicSORM. They provide a similar behavior which is not surprising since, in the linear case, the mappings are identical: We just rotate the coordinate system. Changing the input strategy does not change the error of the networks with randomly generated DR weights (6.7 or 8.8, respectively, false detections minimum using random or equally distributed input weights) and SORM (8.0 or 7.3, respectively, false detections minimum using random or equally distributed input weights) much. On the

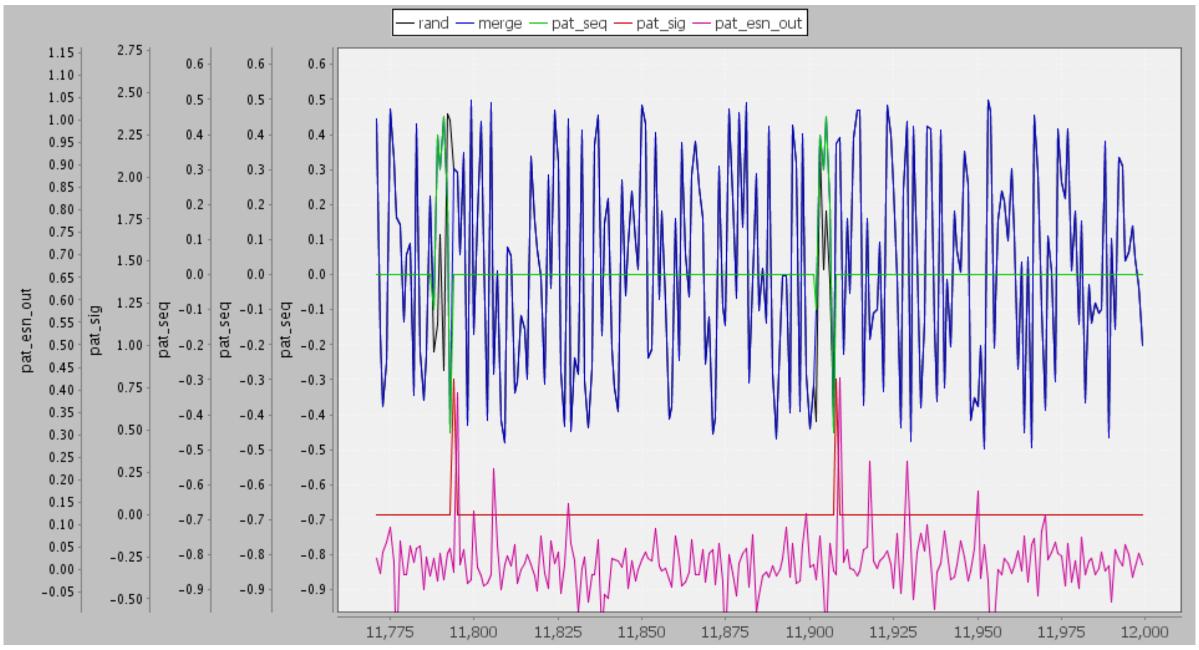

Figure 4: Pattern detection: time series of the random sequence (black), the merged sequence (blue), the pattern (green), the training signal (red) and the network output (pink)

other side, the RingOfNeurons and the ChainOfNeurons produce similar error rates in the case of random inputs. But if the inputs are equally distributed, the error rates decrease to only one false detection on average for the best setup. In this case, also the variance (shown as errorbars in Figure 5) is small in relation to the other weight matrices.

At this point, we want to note some observations we made during this experiment. We tested with different bias scales and interpreted a bias as input (expectation value) shift which possibly increases cond($\boldsymbol{S}^T\boldsymbol{S}$). We obtained the best results with no bias. This supports one assumption for the estimates of sections 4, namely $\mathbb{E}(u) = 0$ which is (almost) satisfied with no bias.

5.3. Mackey-Glass

The Mackey-Glass experiment is a prediction task, where the system has to do a 1-step ahead prediction of the Mackey-Glass time series (a nonlinear chaotic time series, see Jaeger (2001a) for details and parameters). In the “free run” mode, the system will perform several of these prediction steps and therefore produce a given number of steps ahead into the future, in other words: it will “generate” the time series. For our experiments, the Mackey-Glass time series is generated with $\tau = 17$, the traditional value for cautiously chaotic behavior.

The system is designed with 400 reservoir neurons connected as before by one of the corresponding connection matrices and one output neuron aiming to predict the next

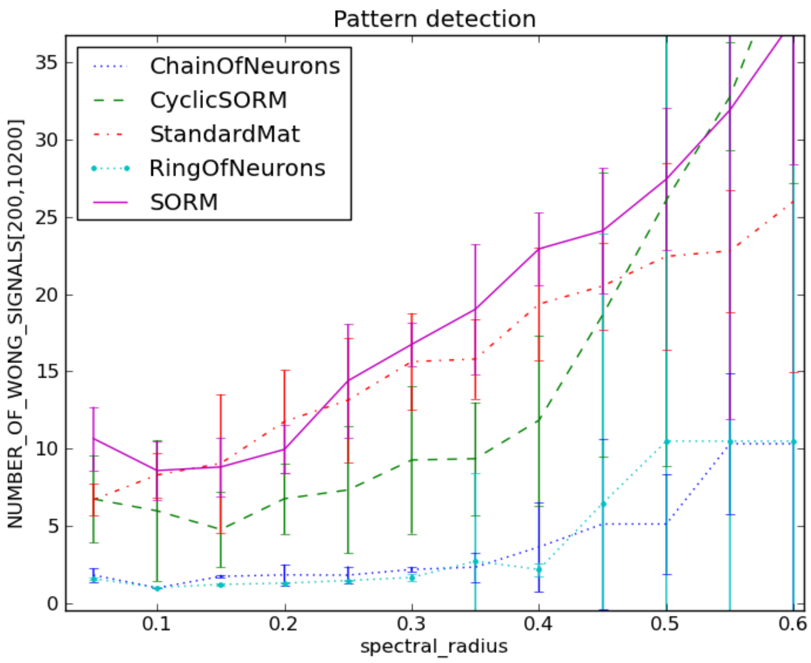

Figure 5: Results of pattern detection experiments averaged over 60 different initializations. The errorbars denote the variance.

step of the time series. The system input is organized by feeding back the output of the system: The single output neuron is connected back into the reservoir by feedback connections. These feedback connections are treated just like input connections.

Except for the “free run” mode, the output value of the system (activation of the output neuron) is set to the sequence value (teacher forcing mode for washout and training). We do not touch the system in the free “run mode”. The size of the training and test data sets is 5000 including 500 steps washout. The “free run” mode lasts 200 steps. We measure the NRMSE over 12 different matrix initializations with 12 different runs for each parameter setup. The search includes the bias scaling factor, the number of input connections, the input and reservoir scaling factor. Both the random and the equidistant input connection modes are tested. For this experiment, we use leaky integrator neurons within the reservoir which after the update according to equation (4) additionally perform the following second step:

$$\boldsymbol{x}'(n+1) = (1-\delta)\boldsymbol{x}'(n+1) + \delta\boldsymbol{x}(n+1).$$

The value $\boldsymbol{x}'(n)$ is used as neuron activation instead of $\boldsymbol{x}(n)$ at time n. We use constant $\delta = 0.4$ as the Oger samples suggest. Table 4 shows other parameters for obtaining a minimum error rate.

Figure 7 shows the experimental results. Since the variances are very small (at least

	bias	ρ	$FScale$	$FDens$
ChainOfNeurons	0.15	0.85	1.1	1.0
RingOfNeurons	0.05	0.95	1.1	1.0
CyclicSORM	0.2	1.0	1.2	1.0
SORM	0.2	1.05	1.2	1.0
StandardMat	0.2	1.1	0.9	1.0

Table 4: Best parameters for the Mackey-Glass experiment (found by Oger gridseach).

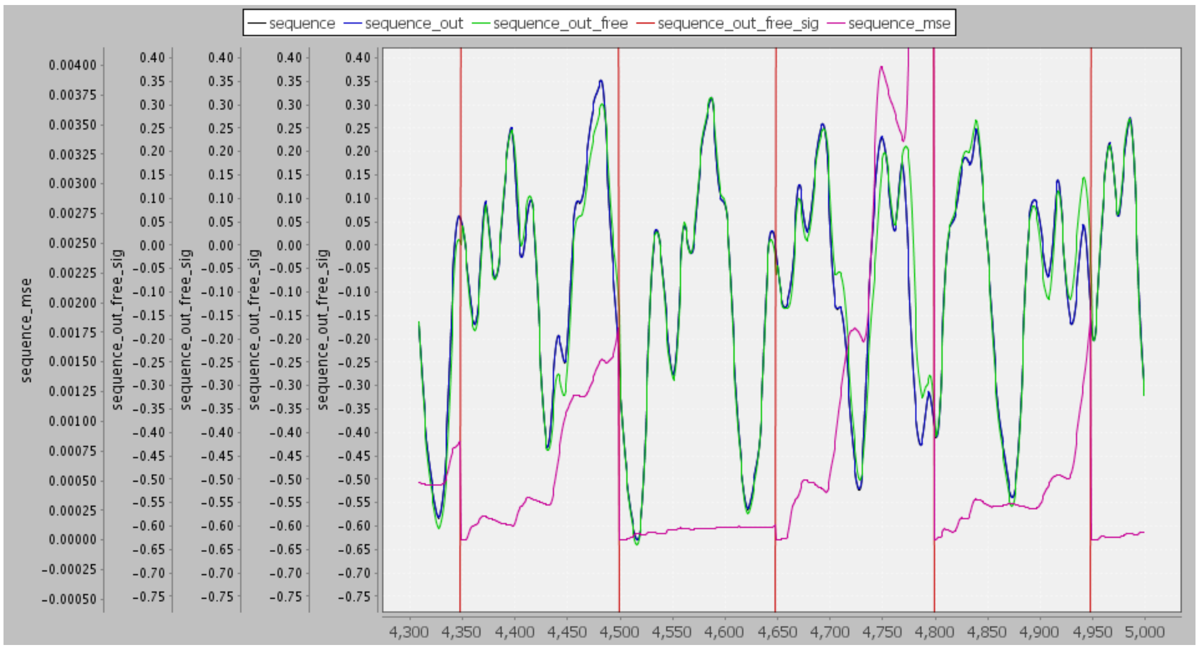

Figure 6: Mackey-Glass experiment: time series of the random sequence (black), the merged sequence (blue), the pattern (green), the training signal (red) and the network output (pink) sequence.

until the spectral radius reaches the minimum), we omit them in the plot. Again the standard ESNs do not perform as well as the other matrices. Interestingly, a complete connection of output and hidden layer yields the best performance for all reservoir types. A sparse but equal distribution did not lead to an improvement.

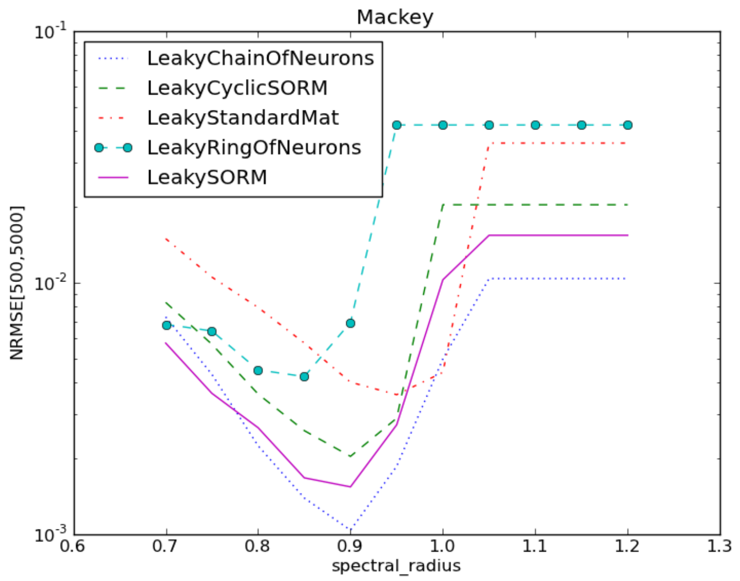

Figure 7: Results of the Mackey-Glass experiment executed with different reservoir types and averaged over 144 different initializations.

5.4. NARMA

The target signal in this section will be the 10th order nonlinear autoregressive moving average (NARMA).

$$t(n+1) = 0.3t(n) + 0.05t(n)\left(\sum_{i=0}^{9} t(n-i)\right) + 1.5u(n-9)u(n) + 0.1$$

The task is to predict this sequence for one future step. The parameters of this sequence were taken from Jaeger (2003) where we refer to for a detailed description of this experiment.

Each neural net has 200 hidden neurons, one input and one output neuron. The data sets contain 2200 elements each, whereby the first 200 steps are for washout. Again, one data set is used to train the neural net and we validate on a second one. We measure the NMSE error. The grid of the parameter search consists of the input scaling, the weight scaling value of the reservoir and the number of input connections. For every parameter set, 5 different reservoir initializations each with 10 different data sets are tested. Again, we connect input and hidden layer in the two already mentioned ways. Table 5 provides the optimal parameters resulting from the gridsearch.

Figure 8 plots the average NMSE and the minimum NMSE against the spectral radius. The CyclicSORM, the RingOfNeurons, and the ChainOfNeurons generate the smallest error rates. In contrast to the above experiments, the average error rates of all SORMs

	ρ	$IScale$	$IDens$
ChainOfNeurons	0.75	0.95	1/9
RingOfNeurons	0.75	0.95	1/9
CyclicSORM	0.8	0.75	1/9
SORM	0.65	0.45	1/5
StandardMat	0.75	0.2	1/8

Table 5: Best parameters for the NARMA experiment (found by Oger gridsearch).

this time are worse than the random matrices. The variances of the errors are all small (of magnitude 10^{-6}) except for the ESNs with small spectral radius (0.7, 0.75). So we omit the errorbars.

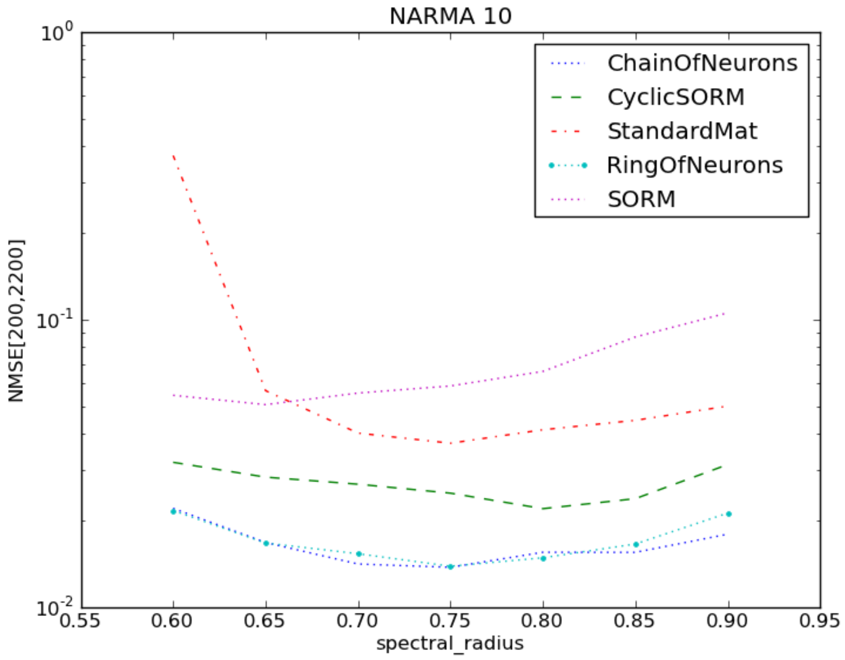

Figure 8: Results of the NARMA experiment executed with different reservoir types and averaged over 50 different initializations.

6. Conclusions and outlook

We have introduced design strategies for reservoir weight matrices and analyzed their linear algebraic properties. In contrast to many state of the art articles, we consider superpositions as a source of errors instead of "interesting echos". In this paper, we compare the different approaches at 4 academic test scenarios:

- short term memory test
- pattern detection
- discretization of the Mackey-Glass differential equation
- NARMA

Finally, we come up with the surprisingly simple ChainOfNeurons, which is shown to be the most robust reservoir in the linear case and performed very well in our experiments. We like to point out that ChainOfNeurons is not even a recurrent system anymore. It seems that the reservoir does not need internal dynamics for many tasks. In comparision to the recent Echo State approach, the ChainOfNeurons has the following advantages:

- extremely sparse (1 input connection per unit) and computationally very fast
- maximum robustness against noise on the training data
- no random variants of different reservoirs
- clearly defined and maximum short term memory
- much easier for applying additional approaches as, e.g., leaky or bandpass filters (which make the ChainOfNeurons slower from the beginning of the chain up to its end)
- much more intuitive and therefore easier to handle "echos", e.g. the historical inputs

As a small note, we would like to add the following remarks: Firstly, in case of multi-dimensional inputs we would recommend a separat ChainOfNeurons for each dimension or a maximum distance between the input vectors on the ChainOfNeurons because the idea is to reduce superposition as much as possible. Secondly, remembering previous research, we tested special reservoir topologies, as for example sparse versus densely connected matrices, or small world and fractal connection topologies. There we discovered that it does not matter what topology we use for certain tasks. We think now that the results, presented in this paper, fit with this experience. It seems that linear algebraic properties are more essential than the topology of the connections.

From our perspectives there are several interesting open points for further research:

Training algorithms It has turned out that Echo State Networks deal not very well with gradient descent learning techniques. Roughly spoken, we think that this is related to the spectrum of eigenvalues of these reservoirs. Different inputs from different time scales will be represented by the echos of the reservoir with extremely different magnitudes. The ChainOfNeurons, as well as SORMs and CyclicSORMs have different properties due to representing previous inputs. So we think, it could be interesting to evaluate different gradient descent learning techniques for the ChainOfNeurons approach. This is also related to the next point.

Time Delay Neural Networks (TDNN) The ChainOfNeurons and the TDNN approaches are quite similar. Indeed an ChainOfNeurons could be considered as a certain TDNN. So we think, it will be interesting to evaluate TDNNs and its recent learning techniques in comparison with the ChainOfNeurons approach.

Implications for reservoir computing technologies From our perspective it is an open question, how the presented results will affect the research of other reservoir computing technologies as for example the liquid state machines (LSMs). It would be interesting to evaluate and if possible to adapt or to generalize the presented ideas for related research areas.

7. Acknowledgment

This research was entirely funded by the research grant no. V220-630-08-TFMV-S/F-059 (Verbundvorhaben, Technologieförderung Land Mecklenburg-Vorpommern) in European Social / Regional Development Funds.

References

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79:2554–2558.

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

Jaeger, H. (2001b). Short term memory in echo state networks. Technical Report GMD Report 152, German National Research Center for Information Technology.

Jaeger, H. (2002). A tutorial on training recurrent neuronal networks, covering BPPT,RTRL, EKF and the "echo state network" approach. Technical Report GMD Report 159, German National Research Center for Information Technology. Third revision.

Jaeger, H. (2003). Adaptive nonlinear system identification with echo state networks. In Becker, S. T. S. and Obermayer, K., editors, *Advances in Neural Information Processing Systems*, pages 593–600. MIT Press Cambridge.

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

Ozturk, M. C., Xu, D., and Príncipe, J. C. (2007). Analysis and Design of Echo State Networks. *Neural Computation*, 19:111–138.

Rodan, A. and Tiňo, P. (2011). Minimum complexity echo state network. *Neural Networks, IEEE Transactions on*, 22(1):131 –144.

Čerňanský, M. and Tiňo, P. (2008). Predictive modeling with echo state networks. In *Proceedings of the 18th international conference on Artificial Neural Networks, Part I*, ICANN '08, pages 778–787, Berlin, Heidelberg. Springer-Verlag.

White, O. L., Lee, D. D., and Sompolinsky, H. (2004). Short-term memory in orthogonal neural networks. *Physical Review Letters*, 92:148102.

A. Appendix

In this section, we need the generalized or pseudoinverse of a matrix $\boldsymbol{S}$. Let

$$\boldsymbol{S} = \boldsymbol{U}\begin{pmatrix}\boldsymbol{\Sigma} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{pmatrix}\boldsymbol{V}^*, \quad \boldsymbol{\Sigma} = \begin{pmatrix}\sigma_1 & 0 & \dots & 0\\ 0 & \sigma_2 & \ddots & \vdots\\ \vdots & \ddots & \ddots & 0\\ 0 & \dots & 0 & \sigma_r\end{pmatrix} \tag{27}$$

be the singular value decomposition (SVD) of $\boldsymbol{S}$ with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$, where $\boldsymbol{U}$ and $\boldsymbol{V}$ are unitary, and $\boldsymbol{\Sigma} \in \mathbb{R}^{r\times r}$ (i.e. $r = \operatorname{rank}(\boldsymbol{S})$). Then the pseudoinverse $\boldsymbol{S}^+$ of $\boldsymbol{S}$ is defined as

$$\boldsymbol{S}^+ = \boldsymbol{V}\begin{pmatrix}\boldsymbol{\Sigma}^{-1} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{pmatrix}\boldsymbol{U}^*. \tag{28}$$

$\boldsymbol{S}^+$ is uniquely determined by the Moore–Penrose criteria: $\boldsymbol{S}\boldsymbol{S}^+\boldsymbol{S} = \boldsymbol{S}$, $\boldsymbol{S}^+\boldsymbol{S}\boldsymbol{S}^+ = \boldsymbol{S}^+$, $(\boldsymbol{S}\boldsymbol{S}^+)^* = \boldsymbol{S}\boldsymbol{S}^+$ and $(\boldsymbol{S}^+\boldsymbol{S})^* = \boldsymbol{S}^+\boldsymbol{S}$.

Lemma 8. Let $\boldsymbol{V}\begin{pmatrix}\boldsymbol{\Sigma} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{pmatrix}\boldsymbol{V}^*$ be the SVD of $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ where $\boldsymbol{\Sigma} \in \mathbb{R}^{r\times r}$ is a regular $r \times r$ diagonal matrix and $\boldsymbol{J}_u = \begin{pmatrix}\boldsymbol{I}_r & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{pmatrix}$. Let $\boldsymbol{B} \in \mathbb{R}^{n\times m}$ for any $m \in \mathbb{N}$ and $\|\boldsymbol{A}^+\boldsymbol{B}\| < 1$ for some appropriate, sub-multiplicative matrix norm $\|\cdot\|$. Then

$$\left[\boldsymbol{A}(\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^* + \boldsymbol{A}^+\boldsymbol{B}\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*)\right]^+ = \left(\sum_{i=0}^{\infty}(-1)^i(\boldsymbol{A}^+\boldsymbol{B})^i\right)\boldsymbol{A}^+. \tag{29}$$

Proof. We denote $\boldsymbol{C} := \boldsymbol{A}(\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^* + \boldsymbol{A}^+\boldsymbol{B}\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*)$ and $\boldsymbol{C}^- := \left(\sum_{i=0}^{\infty}(-1)^i(\boldsymbol{A}^+\boldsymbol{B})^i\right)\boldsymbol{A}^+$. Furthermore, we define $(\boldsymbol{A}^+\boldsymbol{B})^0 = \boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*$.

$$\boldsymbol{C}\boldsymbol{C}^- = \boldsymbol{A}\left(\sum_{i=0}^{\infty}(-1)^i(\boldsymbol{A}^+\boldsymbol{B})^i + (-1)^i(\boldsymbol{A}^+\boldsymbol{B})^{i+1}\right)\boldsymbol{A}^+ \tag{30}$$

$$= \boldsymbol{A}\left(\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*\right)\boldsymbol{A}^+ = \boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^* \tag{31}$$

In the same way we get

$$\boldsymbol{C}^-\boldsymbol{C} = \left(\sum_{i=0}^{\infty}(-1)^i(\boldsymbol{A}^+\boldsymbol{B})^i\right)\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*\left[\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^* + \boldsymbol{A}^+\boldsymbol{B}\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*\right] \tag{32}$$

$$= \left(\sum_{i=0}^{\infty}(-1)^i(\boldsymbol{A}^+\boldsymbol{B})^i\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^* + (-1)^i(\boldsymbol{A}^+\boldsymbol{B})^{i+1}\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*\right) = \boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*. \tag{33}$$

Therefore, $(\boldsymbol{C}^-\boldsymbol{C})^* = \boldsymbol{C}^-\boldsymbol{C}$ and $(\boldsymbol{C}\boldsymbol{C}^-)^* = \boldsymbol{C}\boldsymbol{C}^-$. Furthermore, $\boldsymbol{C}\boldsymbol{C}^-\boldsymbol{C} = \boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*\boldsymbol{A}(\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^* + \boldsymbol{A}^+\boldsymbol{B}\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*) = \boldsymbol{C}$ and $\boldsymbol{C}^-\boldsymbol{C}\boldsymbol{C}^- = \left(\sum_{i=0}^{\infty}(-1)^i(\boldsymbol{A}^+\boldsymbol{B})^i\right)\boldsymbol{A}^+\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^* = \boldsymbol{C}^-$. Thus, the Moore–Penrose criteria are satisfied and $\boldsymbol{C}^-$ is the pseudoinverse of $\boldsymbol{C}$.

Let $\|\boldsymbol{M}\|_{max} := \max_{1\le i,j\le n}|m_{i,j}|$. The sum $\sum_{i=0}^{\infty}(-1)^i(\boldsymbol{A}^+\boldsymbol{B})^i$ converges since for any $\epsilon > 0$, there is some $C \in \mathbb{N}$, such that for any $k \ge l > C$

$$\left\|\sum_{i=0}^{k}(-1)^i(\boldsymbol{A}^+\boldsymbol{B})^i - \sum_{i=0}^{l}(-1)^i(\boldsymbol{A}^+\boldsymbol{B})^i\right\|_{max} = \left\|\sum_{i=l+1}^{k}(-1)^i(\boldsymbol{A}^+\boldsymbol{B})^i\right\|_{max} \tag{34}$$

Because of the equivalence of all norms, there is an α such that

$$\le \alpha\left\|\sum_{i=l+1}^{k}(-1)^i(\boldsymbol{A}^+\boldsymbol{B})^i\right\| \tag{35}$$

$$\le \alpha\sum_{i=l+1}^{k}\|\boldsymbol{A}^+\boldsymbol{B}\|^i \le \epsilon. \tag{36}$$

□

Proof of Theorem 4. Remember that $\widehat{\boldsymbol{w}}^{out} = \boldsymbol{w}^{out} + \boldsymbol{\Delta}\boldsymbol{w}^{out}$ is the solution of $(\boldsymbol{S} + \boldsymbol{\Delta}\boldsymbol{S})\widehat{\boldsymbol{w}}^{out} = \boldsymbol{t}$, whereas $\boldsymbol{w}^{out} := \boldsymbol{t}^T\boldsymbol{S}^+$. Below, we will neglect terms containing $\boldsymbol{\Delta}\boldsymbol{S}$ of higher order.

$$(\boldsymbol{S} + \boldsymbol{\Delta}\boldsymbol{S})(\widehat{\boldsymbol{w}}^{out})^T = \boldsymbol{t} \tag{37}$$

$$(\boldsymbol{S} + \boldsymbol{\Delta}\boldsymbol{S})^T(\boldsymbol{S} + \boldsymbol{\Delta}\boldsymbol{S})(\widehat{\boldsymbol{w}}^{out})^T = (\boldsymbol{S} + \boldsymbol{\Delta}\boldsymbol{S})^T\boldsymbol{t} \tag{38}$$

Let

$$\boldsymbol{S} = \boldsymbol{U}\begin{pmatrix}\boldsymbol{\Sigma} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{pmatrix}\boldsymbol{V}^* \tag{39}$$

the SVD of $\boldsymbol{S}$. We multiply equation (38) by $\boldsymbol{V}\left(\boldsymbol{J}_u+\boldsymbol{J}_l\right)\boldsymbol{V}^*=\boldsymbol{I}_N$ whereas $\boldsymbol{J}_u=\begin{pmatrix}\boldsymbol{I}_r & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{pmatrix}$ and $\boldsymbol{J}_l=\begin{pmatrix}\boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{I}_{N-r}\end{pmatrix}$ are the upper and lower part of the identity matrix. Obviously, $\boldsymbol{J}_l$ is a diagonal matrix with $N-r$ one entries and zeros else.

$$\boldsymbol{V}\left(\boldsymbol{J}_u+\boldsymbol{J}_l\right)\boldsymbol{V}^*(\boldsymbol{S}+\boldsymbol{\Delta S})^T(\boldsymbol{S}+\boldsymbol{\Delta S})\boldsymbol{V}\left(\boldsymbol{J}_u+\boldsymbol{J}_l\right)\boldsymbol{V}^*(\widehat{\boldsymbol{w}}^{out})^T=\boldsymbol{V}\left(\boldsymbol{J}_u+\boldsymbol{J}_l\right)\boldsymbol{V}^*(\boldsymbol{S}+\boldsymbol{\Delta S})^T\boldsymbol{t} \tag{40}$$

We separate this into four equations

$$\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*(\boldsymbol{S}+\boldsymbol{\Delta S})^T(\boldsymbol{S}+\boldsymbol{\Delta S})\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*(\widehat{\boldsymbol{w}}_1^{out})^T=\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*(\boldsymbol{S}+\boldsymbol{\Delta S})^T\boldsymbol{t} \tag{41}$$

$$\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*(\boldsymbol{S}+\boldsymbol{\Delta S})^T(\boldsymbol{S}+\boldsymbol{\Delta S})\boldsymbol{V}\boldsymbol{J}_l\boldsymbol{V}^*(\widehat{\boldsymbol{w}}_2^{out})^T=\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*(\boldsymbol{S}+\boldsymbol{\Delta S})^T\boldsymbol{t} \tag{42}$$

$$\boldsymbol{V}\boldsymbol{J}_l\boldsymbol{V}^*(\boldsymbol{S}+\boldsymbol{\Delta S})^T(\boldsymbol{S}+\boldsymbol{\Delta S})\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*(\widehat{\boldsymbol{w}}_3^{out})^T=\boldsymbol{V}\boldsymbol{J}_l\boldsymbol{V}^*(\boldsymbol{S}+\boldsymbol{\Delta S})^T\boldsymbol{t} \tag{43}$$

and

$$\boldsymbol{V}\boldsymbol{J}_l\boldsymbol{V}^*(\boldsymbol{S}+\boldsymbol{\Delta S})^T(\boldsymbol{S}+\boldsymbol{\Delta S})\boldsymbol{V}\boldsymbol{J}_l\boldsymbol{V}^*(\widehat{\boldsymbol{w}}_4^{out})^T=\boldsymbol{V}\boldsymbol{J}_l\boldsymbol{V}^*(\boldsymbol{S}+\boldsymbol{\Delta S})^T\boldsymbol{t}. \tag{44}$$

For small $\|\boldsymbol{\Delta S}\|$, we approximate

$$(\boldsymbol{S}+\boldsymbol{\Delta S})^T(\boldsymbol{S}+\boldsymbol{\Delta S})\approx(\boldsymbol{S}^T\boldsymbol{S}+\boldsymbol{\Delta S}^T\boldsymbol{S}+\boldsymbol{S}^T\boldsymbol{\Delta S}). \tag{45}$$

We analyze (41) first.

$$(\widehat{\boldsymbol{w}}_1^{out})^T\approx[\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*(\boldsymbol{S}^T\boldsymbol{S}+\boldsymbol{S}^T\boldsymbol{\Delta S}+\boldsymbol{\Delta S}^T\boldsymbol{S})\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*]^+\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*(\boldsymbol{S}+\boldsymbol{\Delta S})^T\boldsymbol{t} \tag{46}$$

$$=\left[\boldsymbol{S}^T\boldsymbol{S}(\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*+(\boldsymbol{S}^T\boldsymbol{S})^+(\boldsymbol{\Delta S}^T\boldsymbol{S}+\boldsymbol{S}^T\boldsymbol{\Delta S}))\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*\right]^+\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*(\boldsymbol{S}+\boldsymbol{\Delta S})^T\boldsymbol{t} \tag{47}$$

Due to Lemma 8 the generalized inverse of $\boldsymbol{S}^T\boldsymbol{S}(\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*+(\boldsymbol{S}^T\boldsymbol{S})^+(\boldsymbol{\Delta S}^T\boldsymbol{S}+\boldsymbol{S}^T\boldsymbol{\Delta S}))\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*$ is
$\left(\sum_{k=0}^{\infty}(-1)^k((\boldsymbol{S}^T\boldsymbol{S})^+(\boldsymbol{\Delta S}^T\boldsymbol{S}+\boldsymbol{S}^T\boldsymbol{\Delta S}))^k\right)(\boldsymbol{S}^T\boldsymbol{S})^+$ if $\|\boldsymbol{\Delta S}\|<\frac{1}{2}\sigma_{max}(\boldsymbol{S})^{-1}\sigma_{min}(\boldsymbol{S})^2$. We approximate the pseudoinverse by the first two terms.

$$\approx\left[\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*-(\boldsymbol{S}^T\boldsymbol{S})^+(\boldsymbol{S}^T\boldsymbol{\Delta S}+\boldsymbol{\Delta S}^T\boldsymbol{S})\right](\boldsymbol{S}^T\boldsymbol{S})^+(\boldsymbol{S}+\boldsymbol{\Delta S})^T\boldsymbol{t} \tag{48}$$

$$\approx\underbrace{(\boldsymbol{S}^T\boldsymbol{S})^+\boldsymbol{S}^T\boldsymbol{t}}_{=(\boldsymbol{w}^{out})^T}+(\boldsymbol{S}^T\boldsymbol{S})^+\boldsymbol{\Delta S}^T\underbrace{(\boldsymbol{t}-\boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S})^+\boldsymbol{S}^T\boldsymbol{t})}_{=\boldsymbol{t}-\boldsymbol{S}(\boldsymbol{w}^{out})^T=:\boldsymbol{r}}-(\boldsymbol{S}^T\boldsymbol{S})^+\boldsymbol{S}^T\boldsymbol{\Delta S}\underbrace{(\boldsymbol{S}^T\boldsymbol{S})^+\boldsymbol{S}^T\boldsymbol{t}}_{=(\boldsymbol{w}^{out})^T} \tag{49}$$

The term $\boldsymbol{r}$ is called residuum and describes the approximation error. We obtain a first-order approximation of $\Delta \boldsymbol{w}_1^{out} := \widehat{\boldsymbol{w}}_1^{out} - \boldsymbol{w}^{out}$.

$$(\Delta \boldsymbol{w}_1^{out})^T \approx (\boldsymbol{S}^T\boldsymbol{S})^+ \Delta \boldsymbol{S}^T \boldsymbol{r} - (\boldsymbol{S}^T\boldsymbol{S})^+ \boldsymbol{S}^T \Delta \boldsymbol{S} (\boldsymbol{w}^{out})^T \tag{50}$$

$$= (\boldsymbol{S}^T\boldsymbol{S})^+ \Delta \boldsymbol{S}^T \boldsymbol{r} - \boldsymbol{S}^+ \Delta \boldsymbol{S} (\boldsymbol{w}^{out})^T \tag{51}$$

For $i \in \{2,3,4\}$, $\widehat{\boldsymbol{w}}_i^{out}$ contains $\boldsymbol{J_l}$ such that we obtain (together with (45))

$$(\widehat{\boldsymbol{w}}_2^{out})^T \approx \left(\boldsymbol{S}^T \Delta \boldsymbol{S} \boldsymbol{V} \boldsymbol{J}_l \boldsymbol{V}^*\right)^+ \boldsymbol{V} \boldsymbol{J}_u \boldsymbol{V}^* (\boldsymbol{S} + \Delta \boldsymbol{S})^T \boldsymbol{t} \tag{52}$$

$$(\widehat{\boldsymbol{w}}_3^{out})^T \approx \left(\boldsymbol{V} \boldsymbol{J}_l \boldsymbol{V}^* \Delta \boldsymbol{S}^T \boldsymbol{S}\right)^+ \boldsymbol{V} \boldsymbol{J}_l \boldsymbol{V}^* \Delta \boldsymbol{S}^T \boldsymbol{t} \tag{53}$$

and

$$(\widehat{\boldsymbol{w}}_4^{out})^T \approx \boldsymbol{0}. \tag{54}$$

The last approximation results from $\boldsymbol{S}\boldsymbol{V}\boldsymbol{J}_l = \boldsymbol{0}$ and $\boldsymbol{J}_l \boldsymbol{V}^* \boldsymbol{S} = \boldsymbol{0}$. Finally, we obtain

$$\Delta \boldsymbol{w}^{out} = \widehat{\boldsymbol{w}}_1^{out} + \widehat{\boldsymbol{w}}_2^{out} + \widehat{\boldsymbol{w}}_3^{out} + \widehat{\boldsymbol{w}}_4^{out} - \boldsymbol{w}^{out} \tag{55}$$

$$\approx \boldsymbol{r}^T \Delta \boldsymbol{S} (\boldsymbol{S}^T\boldsymbol{S})^+ - \boldsymbol{w}^{out} \Delta \boldsymbol{S}^T (\boldsymbol{S}^+)^T + \boldsymbol{t}^T \Big[(\boldsymbol{S} + \Delta \boldsymbol{S}) \boldsymbol{V} \boldsymbol{J}_u \boldsymbol{V}^* \left(\boldsymbol{V} \boldsymbol{J}_l \boldsymbol{V}^* \Delta \boldsymbol{S}^T \boldsymbol{S}\right)^+ + + \Delta \boldsymbol{S} \boldsymbol{V} \boldsymbol{J}_l \boldsymbol{V}^* \left(\boldsymbol{S}^T \Delta \boldsymbol{S} \boldsymbol{V} \boldsymbol{J}_l \boldsymbol{V}^*\right)^+ \Big] \tag{56}$$

$\square$

Proof of Lemma 5. If $\operatorname{rank}(\boldsymbol{M}) = \operatorname{rank}(\boldsymbol{S})$, the proposition is obvious because then $\boldsymbol{J}_l$ just contains zeros by definition. So let us prove $\operatorname{rank}(\boldsymbol{M}) = \operatorname{rank}(\boldsymbol{S})$. Recall $\boldsymbol{S}^T = (\boldsymbol{x}(1), \boldsymbol{x}(2), \ldots, \boldsymbol{x}(\theta))$. We assumed that $f = \mathrm{id}$. Thus,

$$\begin{aligned} \operatorname{rank}(\boldsymbol{S}) &= \dim \left\{ \sum_{i=0}^{\theta} \alpha_i \boldsymbol{x}(i) \mid \alpha_i \in \mathbb{R} \right\} = \dim \left\{ \sum_{i=0}^{\theta} \sum_{j=0}^{N-1} \alpha_i u(i-j) \boldsymbol{W}^j \boldsymbol{w}^{in} \mid \alpha_i \in \mathbb{R} \right\} \\ &= \dim \left\{ \sum_{j=0}^{N-1} \boldsymbol{W}^j \boldsymbol{w}^{in} \sum_{i=0}^{\theta} \alpha_i u(i-j) \mid \alpha_i \in \mathbb{R},\ u(k) = 0 \text{ if } k < 0 \right\}. \end{aligned}$$

Due to the N nonzero inputs, for any $j \in \{0, 1, \ldots, N-1\}$ there exist real values $(\alpha_i)_{i=0,\ldots,\theta}$ such that for any $\beta_j \in \mathbb{R}$: $\beta_j = \sum_{i=0}^{\theta} \alpha_i u(i-j)$, such that

$$\operatorname{rank}(\boldsymbol{S}) = \dim(\operatorname{span}\{\boldsymbol{W}^i \boldsymbol{w}^{in} \mid i = 0, 1 \ldots, N-1\}) = \operatorname{rank}(\boldsymbol{M}).$$

$\square$

Proof of Lemma 6. Let $\boldsymbol{S} \in \mathbb{R}^{\theta \times N}$, i.e. θ is the training size.

$$\begin{aligned}
\frac{1}{\theta}\boldsymbol{S}^T\boldsymbol{S} =& \frac{1}{\theta}\sum_{t=0}^{\theta}\boldsymbol{x}(t)\boldsymbol{x}(t)^T && (57)\\
=& \frac{1}{\theta}\sum_{t=0}^{\theta}\left(\sum_{i=0}^{N-1}\boldsymbol{W}^i\boldsymbol{w}^{in}u(t-i)\right)\left(\sum_{k=0}^{N-1}\boldsymbol{W}^k\boldsymbol{w}^{in}u(t-k)\right)^T && (58)\\
=& \frac{1}{\theta}\sum_{i=0}^{N-1}\sum_{k=0}^{N-1}\boldsymbol{W}^i\boldsymbol{w}^{in}\left(\boldsymbol{W}^k\boldsymbol{w}^{in}\right)^T\sum_{t=0}^{\theta}u(t-k)u(t-i) && (59)\\
=& \boldsymbol{M}\boldsymbol{M}^T\sum_{t=0}^{\theta}\frac{(u(t))^2}{\theta} + \sum_{i=0}^{N-1}\sum_{\substack{k=0\\k\neq i}}^{N-1}\boldsymbol{W}^i\boldsymbol{w}^{in}\left(\boldsymbol{W}^k\boldsymbol{w}^{in}\right)^T\sum_{t=0}^{\theta}\frac{u(t-k)u(t-i)}{\theta} && (60)
\end{aligned}$$

For $k \neq i$

$$\lim_{\theta\to\infty}\frac{1}{\theta}\sum_{t=0}^{\theta}u(t-k)u(t-i) = \mathbb{E}(u)^2 = 0 \tag{61}$$

because $u(t-k)$ and $u(t-i)$ are independently distributed with zero expectation. In case of $k = i$

$$\lim_{\theta\to\infty}\frac{1}{\theta}\sum_{t=0}^{\theta}(u(t-k))^2 = \mathbb{E}(u^2) = \mathbb{V}(u) \tag{62}$$

Let $\sigma_i(\boldsymbol{S})$ and $\lambda_i(\boldsymbol{S})$ be the ith singular value of $\boldsymbol{S}$ and the ith eigenvalue of $\boldsymbol{S}$, respectively. For sufficiently large θ, we approximate then

$$\sigma_i\left(\frac{1}{\sqrt{\theta}}\boldsymbol{S}\right) = \sqrt{\frac{1}{\theta}\lambda_i(\boldsymbol{S}^T\boldsymbol{S})} \approx \sqrt{\mathbb{V}(u)\lambda_i(\boldsymbol{M}\boldsymbol{M}^T)} = \sqrt{\mathbb{V}(u)}\sigma_i(\boldsymbol{M}). \tag{63}$$

□