# Forecast of time series using Knowledge-based and Data-driven approaches

GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

Max Planck Institute for
Dynamics and Self-Organization

Report for the module
"Modul B.Phy.406: Einführung ins wissenschaftliche Arbeiten: Bio-physik/Physik komplexer Systeme"

**Luk Fleddermann**
*l.fleddermann@stud.uni-goettingen.de*
Supervisor: Prof. Dr. Ulrich Parlitz
Date of delivery: 09.02.22

# Contents

**1 Introduction** **1**

**2 Knowledge-based approach: Mathematical modeling of excitable media** **2**
2.1 Numerical solutions to spatiotemporal PDEs . . . . . 2
2.2 Models of excitable media . . . . . 4
2.2.1 FitzHugh-Nagumo . . . . . 5
2.2.2 Aliev-Panfilov model . . . . . 6

**3 Data-driven approach: Forecast of time series using Echo state networks** **8**
3.1 Echo state network . . . . . 9
3.1.1 Structure . . . . . 10
3.1.2 Use and reuse . . . . . 11
3.1.3 Initialisation . . . . . 11
3.1.4 Training . . . . . 12
3.2 Application of ESN to the Lorenz system . . . . . 13
3.2.1 Evaluation of a prediction . . . . . 14
3.2.2 Evaluation of a set of hyperparameters . . . . . 16

**4 Future work** **17**

# 1 Introduction

The prediction of the future of high dimensional dynamical systems is a frequent task in medicine, science and economical processes, f.i. cardiac dynamics or prediction of stock trends. Generally two approaches are used to predict future states of the systems.

The first is a knowledge-based approach, in which the dynamics of a system is modeled with a set of differential equations. Solving those differential equations is usually done numerically, since many considered problems lack analytical solutions. Here, it has to be noted that each model only models the reality and hence does not capture the full dynamical system, i.e. parameters used in the description are merely approximations of parameters in real world systems and further terms or variables (and hence complete differential equations) might even be missing completely.

This is the reason for the rapid spread of a second, data-driven approach. *Neural networks* (NN) are wide spread in science today and make use of a large pool of training data, without a deeper insight into the underlying dynamics. Unfortunately, NN predictions for chaotic systems are only valid for short time periods and also require large amounts of training data, which are often not available experimentally.

In this report the two different approaches on prediction of future dynamics are portrayed. This parallel consideration of both approaches urges a third approach: the unification of the other two, the so-called *hybrid approach*. Here, the foundation for the latter is laid such that it can be examined in subsequent work.

The underlying target is hereby the prediction of the future for (chaotic) *excitable media* in the application of cardiac dynamics.

This is directly visible in the treatment of the knowledge-based approach (see Chapter 2). Here, two basic models of excitable media - the *Fitzhugh-Nagumo model* and the *Aliev-Panfilov model* - are presented and numerical results related to phenomena observed in cardiac dynamics. At this point it has to be noted that only the Aliev-Panfilov model is specifically tailored to the application of cardiac dynamics and performs chaotic dynamics for wisely chosen parameters.

In contrast, the introduction of the data-driven approach (see Chapter 3) is kept in a more general form. Here, the initialization, training and use of a specific kind of a NN - an *Echo state network* (ESN) - is presented. Furthermore, in order to present the methodology, an exemplary evaluation on a basic chaotic system (the Lorenz attractor) is performed, which differs from the actual goal of predictions for spatially extended systems mainly in the (low) dimensionality of the system.

# 2 Knowledge-based approach: Mathematical modeling of excitable media

In this chapter two simple models of *excitable media* are presented, the numerical implementation is described and observations are related to cardiac dynamics. Although a detailed description of many partial differential equations (PDEs) is required for a full model of cardiac electrophysiology and hence the models considered here are far from complete, this approach is referred to as the knowledge-based approach [1], as the description of the system is given by a set of known partial differential equations.

Both considered models can be classified as spatiotemporally extended *reaction diffusion systems* of two variables, i.e. sets of differential equations where each equation consists of a (possibly vanishing) diffusion ($\Delta u$) and a reaction term[1] ($R_i(u, w)$), s.t.

$$\partial_t u = R_1(u, w) + d_1 \Delta u, \quad \partial_t w = R_2(u, w) + d_2 \Delta u \quad , \tag{1}$$

for spatially extended variables $u(\vec{x}, t)$ and $w(\vec{x}, t)$ and diffusion constants $d_1$ and $d_2$ [2].

In the following, the numerical solution of reaction diffusion equations, using finite differences, is described in general (see Chap. 2.1). Subsequently, numerical results of the FitzHugh-Nagumo model (Chap. 2.2.1) and the Aliev-Panfilov model (Chap. 2.2.2) are presented.

## 2.1 Numerical solutions to spatiotemporal PDEs

To solve a set of spatiotemporal PDEs as in Equation 1 on a two dimensional spacial domain $\Omega = [0, L]^2$ boundary conditions are required. In the following *no-flux* boundary conditions were chosen. This means, that for the differentiable variable $u$ (resp. $w$) the partial derivatives orthogonal to the boundary vanish at the boundary: $\partial_x u(0, y) = \partial_x u(L, y) = \partial_y u(x, 0) = \partial_y u(x, 0) = 0$ for all $x, y \in [0, L]$ [2].

Further, for the numerical treatment the set of PDEs is turned into a large set of coupled ordinary differential equations (ODEs). That means, instead of considering the continuous variable $u$ (or $w$ resp.) and its temporal evolution, $N^2$ coupled ODEs of the variables $\boldsymbol{u}_n^m(t) := u(mh, nh, t)$, with $m, n \in 0, 1, 2, .., N$ and $Nh = L$ are derived.

[1] Even though conventionally this reaction term might depend on first (spatial) derivatives of $u$ and $w$, finite differences approaches on first derivatives are omitted in Chapter 2.1, since they are not needed in the models considered in Chapter 2.2.

To obtain a time series of the variables $\boldsymbol{u}_n^m$, the ODEs can be integrated over the temporal component using any ODE solver. In the following, a forward Euler method is used for this purpose, which follows

$$\boldsymbol{u}_n^m(l+1) = \boldsymbol{u}_n^m(l) + \Delta t \cdot \frac{\partial \boldsymbol{u}_n^m}{\partial t}(l) + \mathcal{O}(\Delta t^2) \quad , \tag{2}$$

with $\boldsymbol{u}_n^m(l) := \boldsymbol{u}_n^m(l\Delta t + t_0)$, where $l$ indexes the discrete time steps [3, p.20f].

For the right-hand side of the ODEs, a discrete form of the Laplace-Beltrami operator is needed, which approximates the spatial derivatives of $\boldsymbol{u}_n^m = u(mh, nh)$ by values of $u$ only evaluating at adjacent grid points. The finite difference approach suggests the use of a so-called *stencil*.

A quick derivation of the *five-point-stencil* follows to derive an error estimate and illustrate the stencils origin. Therefore the symmetric approximation of derivatives $\frac{\partial u}{\partial x} = \frac{u(x+\Delta x/2) - u(x-\Delta x/2)}{\Delta x} + \mathcal{O}(\Delta x^2)$ is iteratively applied to derive an approximation of the second derivative:

$$\frac{\partial^2 u}{\partial^2 x} = \frac{u(x+\Delta x) + u(x-\Delta x) - 2u(x)}{\Delta x^2} + \mathcal{O}(\Delta x^2) \quad .$$

Consequently, the two dimensional laplacian of $u = \boldsymbol{u}_n^m$ at an arbitrary grid point $(mh, nh) \in [0, L]^2$ can be approximated up to second order by

$$\begin{aligned} \Delta \boldsymbol{u}_n^m &:= \partial_x^2 u(mh, nh) + \partial_y^2 u(mh, nh) \\ &\approx \frac{\boldsymbol{u}_n^{m+1} + \boldsymbol{u}_n^{m-1} + \boldsymbol{u}_{n+1}^m + \boldsymbol{u}_{n-1}^m - 4\boldsymbol{u}_n^m}{h^2} \quad , \end{aligned} \tag{3}$$

with $\Delta x = \Delta y = h$.

By similar reasoning, only using higher order of accuracy, the *nine-point-stencil* can be derived and applied[2] [4]:

$$\begin{aligned} \Delta \boldsymbol{u}_n^m &:= \partial_x^2 u(mh, nh) + \partial_y^2 u(mh, nh) \\ &\approx \frac{\left(\sum_{l,k\in\{-1,1\}} \boldsymbol{u}_{n+k}^{m+l}\right) + 4\left(\boldsymbol{u}_n^{m+1} + \boldsymbol{u}_n^{m-1} + \boldsymbol{u}_{n+1}^m + \boldsymbol{u}_{n-1}^m\right) - 20\boldsymbol{u}_n^m}{6h^2} \quad . \end{aligned} \tag{4}$$

A visualisation of the five- (*left*) and nine-point-stencil (*right*) is given in Table 1 and follows the Equations 3 and 4.

Finally, the no-flux boundary conditions can be enforced numerically by introducing a *ghost layer*, i.e. a layer which is added to the existing mesh by

[2] Even though both approaches have been numerically implemented and produce reasonable results, in the results shown in Chap. 2.2 only the nine-point-stencil is used.

Table 1: Graphical scheme for the five- and nine-point stencil.

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | -4 | 1 |
| 0 | 1 | 0 |

| | | |
|---|---|---|
| 1 | 4 | 1 |
| 4 | -20 | 4 |
| 1 | 4 | 1 |

setting $L$ to $L+2h$ and $N$ to $N+2$, where the outermost cells are artificially equated with the neighboring inner cells at each discrete time step:

$$\begin{aligned} \boldsymbol{u}_{N+1}^m(l) &= \boldsymbol{u}_N^m(l), \quad \boldsymbol{u}_{-1}^m(l) = \boldsymbol{u}_0^m(l) \quad \forall m \in [-1, L+1], \\ \boldsymbol{u}_n^{N+1}(l) &= \boldsymbol{u}_n^N(l), \quad \boldsymbol{u}_n^{-1}(l) = \boldsymbol{u}_n^0(l) \quad \forall n \in [-1, L+1]. \end{aligned} \tag{5}$$

This ends the discretization of the PDEs and the boundary conditions. In the next chapter the previously derived methods are applied to solve two specific sets of PDEs.

## 2.2 Models of excitable media

The models addressed in this chapter portray (as models of cardiac electrophysiology) *excitable media*. This means, that the dynamical systems are excitable, capable of letting waves proceed and a so-called *refractory time* must elapse before renewed excitation can take place [2].

The latter becomes particularly visible when considering waves that run into each other: Here the waves do not pass through but annihilate one another. Figure 1 visualizes the described characteristics, showing excitable (blue) and non-excitable (purple) regions and the annihilation of two colliding wave fronts (in green and yellow), for two concentric waves following the FitzHugh-Nagumo model (comp. Chap. 2.2.1).

As usual for excitable media, in the considered models there is a fast variable $u$ and a slow variable $w$. Generally speaking an excitation is a quick approach of an extremum for the fast variable $u$.

Relating the models to the application - cardiac electrophysiology - $u$ can be considered as cell membrane voltage and $w$ as ionic currents. The regular heart beat then results from planar waves passing through the heart muscle periodically stimulated by the *sinu-atrial node* [2]. Moreover, the models presented already give a qualitative understanding of further possible states of cardiac activities. Thus, in addition to the usual coordinated contractions of the heart, phenomena such as tachycardia and its transition into ventricular fibrillation can be qualitatively explained and

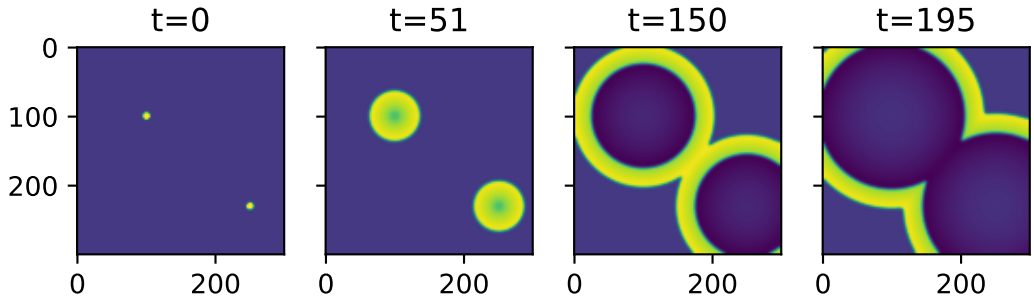


Figure 1: The figure shows the evolution of the variable $u$ of two point-stimuli following the FitzHugh-Nagumo model (comp. Eq. 6) with parameters chosen as in Table 2.

visualized using the models presented [5]. The code used in the numerical implementation, animations and further figures regarding the modeling of of excitable media can be found in a *Gitlab* repository under https://gitlab.gwdg.de/l.fleddermann/reaction_diffusion_modelin.

### 2.2.1 FitzHugh-Nagumo

The FitzHugh-Nagumo model is a simple model of excitable media consisting of two variables following the system of partial differential equations [2, p.7]:

$$\begin{aligned} \frac{\partial u}{\partial t} &= au(u-b)(1-u) - w + d\Delta u \quad , \\ \frac{\partial w}{\partial t} &= -\varepsilon(u-w) \quad , \end{aligned} \tag{6}$$

with model parameters $a$, $b$, $\varepsilon$ and the diffusion constant $d$. The model has been implemented using finite difference methods as depicted in Chapter 2.1. The chosen parameters are shown in Table 2. Hereby the parameters for the discretization of space and time (i.e. spatiotemporal parameters) are separated from the parameters only needed within this specific Model[3].

To this point only planar and concentric waves have been discussed. Figure 2 visualizes the dynamics of another wave pattern, which is relevant in cardiac dynamics. The shown *spiral wave*, was initiated with the conditions for the variables $u$ and $w$ as shown for $t = 0$ in the first pictures of each row, where the uniform steady state $u = w = 0$ is perturbed, s.t. $u = 1$ for $y \leq 120$ and $w = 0.2$ for $x \leq 150$ and $y \geq 120$.

[3]The diffusion constant $d$ is assigned to the spatiotemporal parameters, since a change in the diffusion constant is equivalent to rescaling the space.

Table 2: The table shows the parameters used in the numerical implementation of the FitzHugh-Nagumo.

| Spatiotemporal | | | | | FitzHugh-Nagumo | | |
|---|---|---|---|---|---|---|---|
| L | N | $\Delta x$ | $\Delta t$ | d | a | b | $\varepsilon$ |
| 300 | 300 | 1 | 0.003 | 1 | 3 | 0.2 | 0.01 |

Related to the cardiac dynamics, conceptually, such stable spiral waves generate periodic excitations of the trans-membrane voltage leading to contractions of the muscle with relatively high frequency and low amplitude [5]. This phenomenon is medically known as tachycardia.

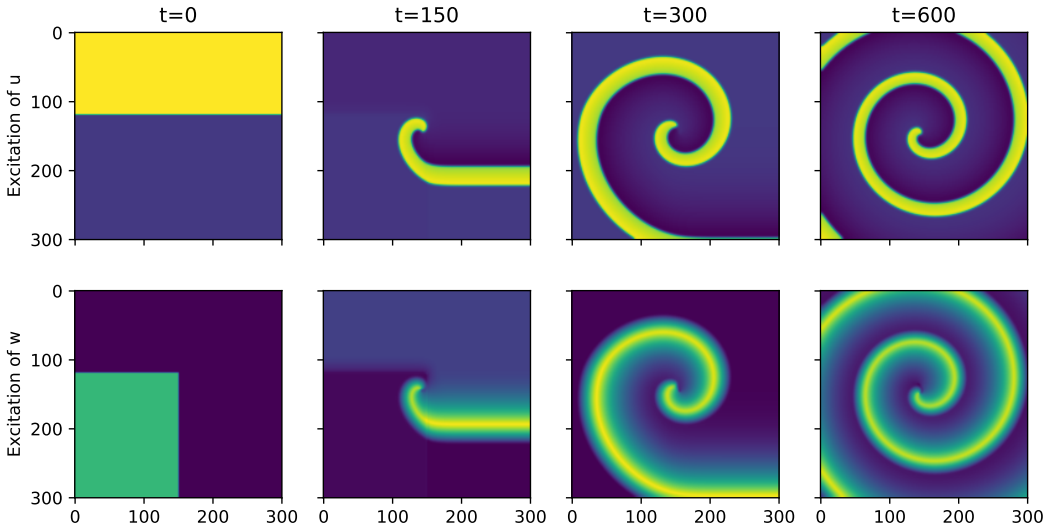


Figure 2: The figure illustrates the initialization of periodic spiral wave using the Fitzhugh-Nagumo model with in Tabel 2 portrayed parameters.

### 2.2.2 Aliev-Panfilov model

The Aliev-Panfilov model is a simple model of excitable media which is specifically tailored to model cardiac dynamics only using two variables [6]. It follows the differential equations[4]:

$$\begin{aligned} \frac{\partial u}{\partial t} &= ku(u-a)(1-u) - uw + d\Delta u \quad , \\ \frac{\partial w}{\partial t} &= \left(\epsilon + \frac{\mu_1 w}{\mu_2 + u}\right) \cdot (-w - ku(u-a-1)) \quad , \end{aligned} \tag{7}$$

[4]The more general form of the diffusion term $\nabla(\mathbf{D}\nabla u)$, with diffusion tensor $\mathbf{D}$ (as it is found in [6]) is simplified and set to a constant $d$.

where $a$, $\varepsilon$, $k$, $\mu_1$ and $\mu_2$ are model parameter and $d$ the diffusion constant.

In addition to the dynamics described so far, the Aliev-Panfilov model is also capable of inducing spatiotemporal chaotic dynamics.

Table 3: The table shows parameters of the Aliev-Panfilov model for stable and unstable spiral waves.

| Spatiotemporal | | | | | Aliev-Panfilov | | | stable | | unstable | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $L$ | $N$ | $\Delta x$ | $\Delta t$ | $d$ | $\mu_1$ | $\mu_2$ | $k$ | $a$ | $\varepsilon$ | $a$ | $\varepsilon$ |
| 300 | 300 | 1 | 0.003 | 0.2 | 0.2 | 0.3 | 8 | 0.09 | 0.01 | 0.05 | 0.02 |

The two sets of parameters used in the numerical implementation are given in Table 3. The first set admits stable spiral waves, leading to non-chaotic dynamics[5]. The second set of parameters induces non-stable spiral waves. This results in break ups of the spirals, which turn the whole dynamics chaotic. The described phenomenon is illustrated in Figure 3. The shown dynamics is initiated at $t = 0$ by setting $u = 1$ in the upper half of the domain and zero on the lower half and further setting $w = 2$ in the lower left quarter and one in the rest of the domain.

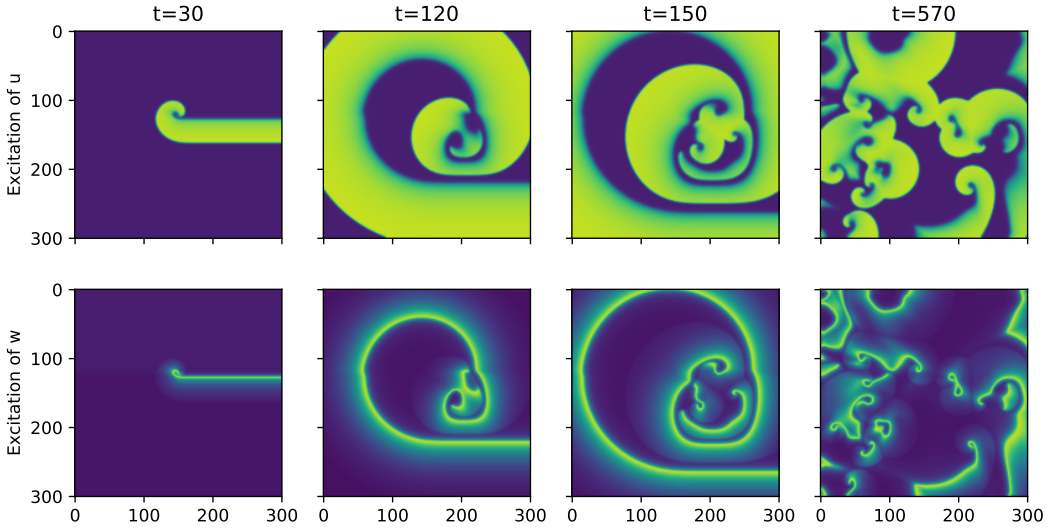


Figure 3: The figure shows how an unstable spiral wave results in chaotic dynamics using the Aliev-Panfilov model.

In fact, the chaotic behavior shown in Figure 3 might not last forever and hence might portray a form of *transient chaos*, as the dynamics could tend to the stable and trivial fixpoint of the system with $u = w = 0$.

[5]The dynamics of stable spiral waves is not shown visually, as it is qualitatively similar to the spiral waves of the Fitzhugh-Nagumo model.

The transition from stable spiral waves to (transient) chaos, which can be simulated by externally changing parameters of the Aliev-Panfilov model, conceptually visualizes the transition from tachycardia to ventricular fibrillation, where the heart fails to pump blood properly due to chaotic excitations of the transmembrane potential [5].

# 3 Data-driven approach: Forecast of time series using Echo state networks

Within the last decades, in addition to the knowledge-based approach a second, data-driven approach has become commonly used. Inspired by biological neural networks and enabled by major advances in computer technology that allow the processing of large amounts of data in a relatively short time, artificial neural networks are more and more widespread. Generally, neural networks approximate a functional relation $f : X \to Y$ between two sets $X \subset \mathbb{R}^n$ and $Y \subset \mathbb{R}^m$, by expanding the knowledge in given *training data*, i.e. elements in the graph $(x, f(x)) \in X \times Y$ of $f$ to the domain of interest $\Omega \subset X$ by assumption of sufficiently smooth functions $f$.

Neural networks can be divided into two categories: feed-forward networks (FFNN) and recurrent neural networks (RNN). Even though this report only deals with RNNs, both of these categories are briefly introduced for comparison.

A FFNN consists of several *layers*. The first and last layer represent the input and output data and are hence called *input* and *output layer*. The middle layers of the network, which are invisible to the user, are called *hidden layers*. A schematic representation of an FFNN is shown in Figure 4 (*left*). Each layer $i$ consists of excitable *nodes* $s_n^i$ (also called neurons), which are connected through *weights* $w_{nm}^i$ to the nodes of the next layer $s_m^{i+1}$. When applying the network on some input data, the values of the nodes are calculated iteratively through the layers by summing the products of the weights and nodes from the previous layer and using a non-linear, sigmoid *transfer function* $f_{tf} : \mathbb{R} \to [0, 1]$ to keep the activation between zero and one, hence $s_m^{i+1} = f_{tf}\left(\sum_n w_{nm}^i s_n^i\right)$. This structure leads to the fact that signals are transmitted only in one direction - forward - which gives the network its name. Furthermore, this is the basis for the most well-known training method of neural networks to work - the *back-propagation algorithm*. Here a cost function, which measures the distance[6] between the prediction

[6]Usually the mean square displacement, combined with a regularization term is considered. Compare with Chapter 3.1.4 and [7].

of the network and the wanted output in the training data is defined and (locally) - as a function of the weights of the network - minimized following a *steepest decent method*. For a closer description of the structure and training of FFNNs [7] may be consulted.

FFNNs are widely used but due to their structure they are particularly suitable for static input output mappings, such as classification problems like pattern recognition.

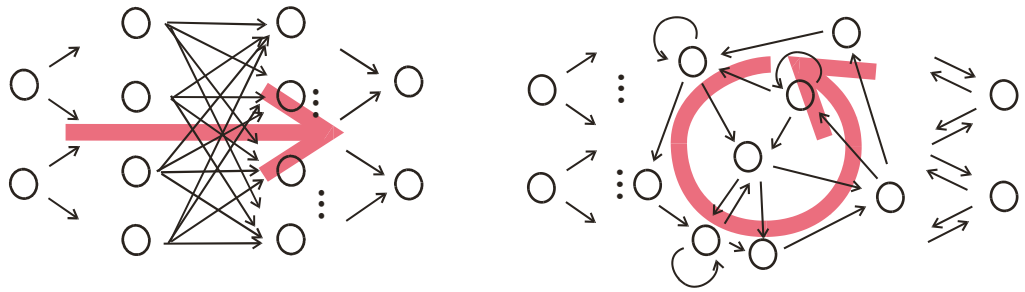

Figure 4: Schematic illustration of a FFNN (*left*) and a RNN (*right*) (Taken from [8]).

In difference to that, RNNs have recurrent connection of the interior nodes, i.e. each inner node is possibly connected to all the other inner nodes and at least one cyclic path of connections exists. A schematic illustration is given in Figure 4 (*right*). The cyclic connections between the nodes cause an influence of past uses of the network on the result of a current application. Thus the network has a memory, which can be beneficial if used correctly. Due to their memory, RNNs are well suited for temporally or spatially extended (discretized) training data that is to be continued over the boundary of a compact set (f.i. time series forecast) usually implemented by iterative applications of the RNN.

However, for the RNN other minimization methods than back propagation need to be used, which are only exemplary discussed in this thesis. For a more general treatment [8] may be consulted.

In the first part of this chapter (see Chap. 3.1) a specific type of RNN, an *Echo state network* (ESN), is introduced. Here, the structure follows the design of Jaeger [9]. In Chapter 3.2 the ESN is exemplary analysed on a basic chaotic system - the Lorenz attractor.

## 3.1 Echo state network

While in general a RNN is trained by changing arbitrary weights of the connections, in the training process of the ESN only the output weights, i.e. the

weights connecting inner nodes to the output vector, are trained. Thus, the inner nodes are considered as a so-called *reservoir* (a driven dynamical network), and only the echoes of the input vectors in the reservoir are learned to be interpreted. This is the key property differentiating the ESN from a general RNN which on one hand leads to a drastic reduction of calculation time within the training and on the other hand increases the numerical stability of the training procedure [10].

Subsequently, the recurrent structure of the ESN is used by an iterative application of the ESN, by using the current output signal of the ESN as the next input. Hence, quantities changing in each step are consistently labeled with the iteration step-number $n$, which can be thought of as the number of temporal steps of same size, i.e. the time at the moment $n$ is $t = n\Delta t$, or number of discrete spacial steps, i.e. the (one dimensional) location $x$ is given by $x = n\Delta x$.

### 3.1.1 Structure

As depicted in the introduction of the chapter, the ESN is subdivided in three sections: First, the input layer, which consists of $(N_u + 1)$ nodes including a input vector of the system $\vec{u}(n)$ and a constant *input bias* $b_{in}$. The second section - the inner layer - consists of $N$ inner nodes, which are arranged in the *state vector* $\vec{s}_n$ and a constant output bias $b_{out}$. The third and last section is the output layer, which depicts the prediction of the network in form of a $N_y$ dimensional vector $\vec{y}(n)$.

A schematic visualisation of the structure and an overview of the used notation is given in Figure 5.

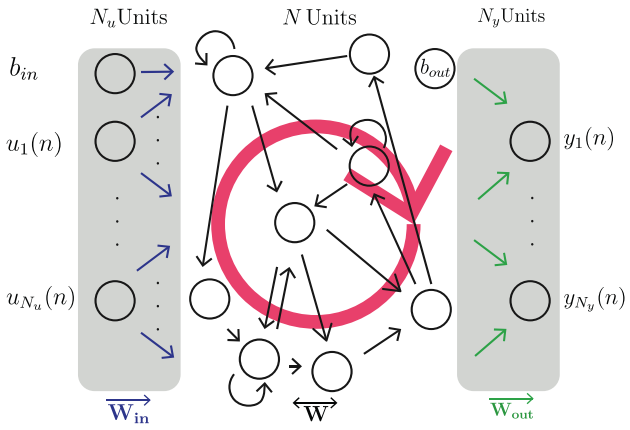


Figure 5: Schematic illustration of the structure of an ESN. Further, the notation used is shown (Taken from Zimmermann and Parlitz [11]).

The connections between the nodes are summarized into three matrices. The input layer is connected to the interior nodes through the *input matrix* $\boldsymbol{W}_{in} \in \mathbb{R}^{N \times (N_u+1)}$. Further, the interior nodes are coupled with each other by the *inner matrix* $\boldsymbol{W} \in \mathbb{R}^{N \times N}$. Lastly, the *output matrix* $\boldsymbol{W}_{out} \in \mathbb{R}^{N_y \times (1+N+N_u)}$ connects the interior nodes and the input vector with the output vector $\vec{y}(n)$.

### 3.1.2 Use and reuse

For the use of a trained ESN, the previous inner states of the ESN $\vec{s}_{n-1}$ and an input vector $\vec{u}(n)$ need to be given in addition to the constant weight matrices. Following Zimmerman and Parlitz [11], in application of the network the inner nodes are updated by

$$\vec{s}_n = (1-\alpha)\vec{s}_{n-1} + \alpha f_{tf}\left(\boldsymbol{W}_{in}[b_{in}, \vec{u}(n)] + \boldsymbol{W}\vec{s}_{n-1}\right) \quad , \tag{8}$$

where $[.,.]$ denotes the concatenation of the two inputs.

Further, two settings to be made externally have been introduced in Equation 8: First, $f_{tf} : \mathbb{R} \to [0,1]$ denotes the transfer function, which is chosen as either $f_{tr} = \tanh$ or $f_{tr}(x) = \frac{1}{1+e^{-x}}$. Second, the *leaking rate* $\alpha \in (0,1]$ is introduced which defines the stiffness of the inner nodes.

To compute the output of the network, an *extended interior state vector*

$$\vec{x}_n = [b_{out}, \vec{s}_n, \vec{u}(n)] \in \mathbb{R}^{1+N+N_u}$$

is defined. Using this definition, the output vector $\vec{y}(n)$ is calculated by $W_{out}\vec{x}_n = \vec{y}(n)$.

To use the trained network, it has to be run for a number of *transient steps* $T_0$ on the training data (by updating the inner state vector following Equation 8) to adjust the inner nodes to the system and the current state within the system.

After the inner nodes have been adjusted to the system state, a prediction can be made. The iterative application of the networks are hence performed by backfeeding the output signal as new input back to the network, i.e. setting $\vec{u}(n+1) := \vec{y}(n)$.

### 3.1.3 Initialisation

Following Zimmermann and Parlitz [11], the entries of the input and interior matrix are chosen to be uniformly distributed random numbers $\in [-0.5, 0.5]$. Further, to increase the number of possible echo signals, the inner matrix is transformed into a sparse matrix. In this way the reservoir decomposes

into many loosely coupled subsystems and thus possesses a large number of possible excitation states [12]. Numerically this is implemented by setting random, non-vanishing entries to zero until only a (small) fraction $\varepsilon \in (0, 1]$ of the entries is non-zero.

Furthermore, an ESN must have the so-called *Echo state property* (compare [13]), which loosely speaking states that the inner states adapt to the current input vectors and previous states and inputs become less important with increasing number of applications. The need for this property becomes particularly clear considering that the randomly chosen start conditions of the inner states $\vec{s}_0$ are supposed to have no influence on the networks output after a sufficiently great amount of steps $T_0$. A network has this property if

$$\rho(|\boldsymbol{W}|) < 1 \quad , \tag{9}$$

where $|.|$ takes the absolute value of each entry of $\boldsymbol{W}$ [13, p.15]. It is important to note that for certain input data ($\vec{u}(n) \neq 0$) this property is not necessarily needed and further well performing networks often even do have great spectral radii. Therefore, an investigation of networks, s.t. Equation 9 is not fulfilled, is also useful.

### 3.1.4 Training

For the training procedure two different methods of minimization are presented in the following. For either the aim of the training process is to find the weights of the output matrix $\boldsymbol{W}_{out}$ such that the predicted output $\vec{y}(n) = \boldsymbol{W}_{out}\vec{x}_n$ matches the true output $\vec{y}^{true}(n)$ as good as possible. Therefore, a set of training data is needed, i.e. tuples of vectors of the kind $(\vec{u}(n), \vec{y}^{\,true}(n))$. Here it has to be noted, that due to the memory of the ESN, two consecutive elements of the training data must be structured in such a way that the relation between $\vec{y}^{true}(n)$ and $\vec{u}(n+1)$ is the same for all $n$. In the following, only the case $\vec{u}(n+1) = \vec{y}^{true}(n)$ is considered, which is used in this report to predict time series.

For a number of transient steps $T_0$ the randomly initialised inner states of the ESN are updated following Equation 8 to adjust the reservoir to the system (-state). Subsequently, the ESN is run for $T$ steps[7] and the extended interior state vectors $\vec{x}_n$ with $n \in \{0, 1, ..., T\}$ are stored as columns in a matrix $\boldsymbol{X} \in \mathbb{R}^{(1+N_u+N)\times T}$. Similarly, the true output data $\vec{y}^{true}(n)$ is stored column wise in a Matrix $\boldsymbol{Y} \in \mathbb{R}^{N_y \times T}$. We now search for the (in general not

[7]The label '$T$' for a number of steps is rather uncommon, but is chosen to continue the imagination of steps as discrete temporal steps of same size.

unique) solution of the linear equation

$$\boldsymbol{Y} = \boldsymbol{W}_{out}\boldsymbol{X} \tag{10}$$

for $\boldsymbol{W}_{out}$. Different methods of solving the system are proposed through the literature.

The most straight forward attempt is the use of the *Moore-Penrose-Pseudoinverse* $\boldsymbol{X'}$ of $\boldsymbol{X}$, s.t.

$$\boldsymbol{W}_{out} = \boldsymbol{Y}\boldsymbol{X'} \quad . \tag{11}$$

The major advantage of this method is the high numerical stability, however it is expensive due to a need of large memory for great numbers of steps $T$. Further, the method has no *regularization term*, i.e. a term suppressing great weights. This restricts the training process to highly overdetermined systems (i.e. $T \gg 1 + N_u + N$), as otherwise overfitting might worsen the prediction of the network significantly [14].

To encounter the latter, a different method, the so-called *Tikhonov regularization* is presented, which introduces a regularization term to suppress overfitting. In the approach the cost function

$$\sum_n \|\vec{y}^{true}(n) - \boldsymbol{W}_{out}\vec{x}_n\|^2 + \beta \sum_i \|\vec{\boldsymbol{W}}_{out,i}\|^2 \quad ,$$

is minimized. Here $\vec{\boldsymbol{W}}_{out,i}$ is the i-th row of $\boldsymbol{W}_{out}$ and $\beta$ the *regularization constant*. The minimization can be executed analytically and returns

$$\boldsymbol{W}_{out} = \boldsymbol{Y}\boldsymbol{X}^T \left(\boldsymbol{X}\boldsymbol{X}^T + \beta \boldsymbol{I}\right)^{-1} \quad . \tag{12}$$

This method is more prone to numerical instability (due to matrix inversion), but produces better results since overfitting is avoided [15]. A more detailed analysis and discussion of the training procedures is beyond the scope of this report. Hence, for a more in-depth treatment of the topic it is referred to [14].

## 3.2 Application of ESN to the Lorenz system

In this chapter, ESNs are applied to the Lorenz system and exemplary methods of analyzing the quality of predictions are presented. The written code used for this purpose, including the implementation of an ESN, can be found in a *Gitlab* repository under https://gitlab.gwdg.de/l.fleddermann/esn. Further, the presented results have decisively influenced

the choice of the selected, so-called *hyperparameters*, i.e. the externally chosen parameters for the network, which are summarized in Table 4.

The Lorenz attractor is a chaotic system following the differential equations [16]:

$$\frac{\partial x}{\partial t} = a(y - x), \quad \frac{\partial y}{\partial t} = x(b - z) - y, \quad \frac{\partial z}{\partial t} = xy - cz \quad . \tag{13}$$

Three dimensional trajectories within the Lorenz system are computed by a numerical integration of the partial differential equations (Eq. 13) with (temporal) step size $\Delta t = 0.01$ and serve as training and evaluation data for the ESN. A single ESN is used to predict the whole Lorenz system, i.e. the input of the network is a state vector $\vec{u}(n) \in \mathbb{R}^3$ at time $n\Delta t$ and the output is the state vector $\vec{y}(n) \in \mathbb{R}^3$ at the time $(n+1)\Delta t$.

The initialization, training and use of the network hereby follows the description in the previous Chapter 3.1. In this case the transfer function $f_{tr} = \tanh$ is used and the choice of hyperparameters and number of training and transient steps portrayed in Table 4. In the training process the Tikhonov regularization (comp. Eq. 12) was used, as numerical tests suggested better performance.

Table 4: The table portrays possible ranges and the chosen values of hyperparameters used in the implementation. Additionally, the number of training and adjustment steps is depicted.

| | hyperparameter | | | | | | training | |
|---|---|---|---|---|---|---|---|---|
| label | $N$ | $\alpha$ | $\beta$ | $\varepsilon$ | $\beta_{in}$ | $\beta_{out}$ | $T$ | $T_0$ |
| range | $\mathbb{N}$ | $[0,1]$ | $\mathbb{R}_{\geq 0}$ | $[0,1]$ | $\mathbb{R}$ | $\mathbb{R}$ | $\mathbb{N}$ | $\mathbb{N}$ |
| chosen value | 800 | 0.2 | 0.0001 | 0.2 | 1 | 1 | 50000 | 100 |

### 3.2.1 Evaluation of a prediction

For the evaluation of the prediction of a trained ESN, a random initial condition $\vec{u}(0) \in [0,1]^3$ is numerically integrated over 100 seconds to derive a random state within the attractor. Afterwards a numerical integration is performed for further $T_{ev} = 2000$ evaluation steps for which the trajectory is saved in a matrix $\boldsymbol{u}^{true} \in \mathbb{R}^{3\times T_{ev}}$. The trained ESN is run on the data for a number of transient steps $T_0$ to adjust the inner nodes to the current state. Afterwards a prediction is performed, by iteratively feeding the output of the ESN back as new input. The predicted trajectory $\boldsymbol{u} \in \mathbb{R}^{3\times(T_{ev}-T_0)}$ is saved in a matrix and compared to the knowledge-based trajectory.

Following Pathak *et al.* [1], for an assessment basis of the quality of a prediction, a normalized prediction error

$$E(n) = \frac{\|\vec{u}(n) - \vec{u}^{true}(n)\|}{\langle \|\vec{u}^{true}\|^2 \rangle^{1/2}} \tag{14}$$

is computed at each time step $n$, where $\langle . \rangle$ denotes the mean over the whole trajectory and $\|.\|$ the euclidean norm. A prediction is considered valid as long as $E(n)$ is smaller than some threshold $e$. Furthermore the *valid time* - which serves as a measure of the quality of the prediction - is defined as $t_{valid} = n_{valid}\Delta t$, with $n_{valid}$ the smallest number s.t. $E(n_{valid} + 1) > e$. In the following $e = 0.1$ is used. Exemplary, the three coordinates of the true (in blue) and the predicted (in green) trajectory are portrayed in Figure 6 as functions of the time (in Lyapunov time) $\lambda_{max}t$, with $\lambda_{max}$ denoting the greatest Lyapunov exponent of the system.

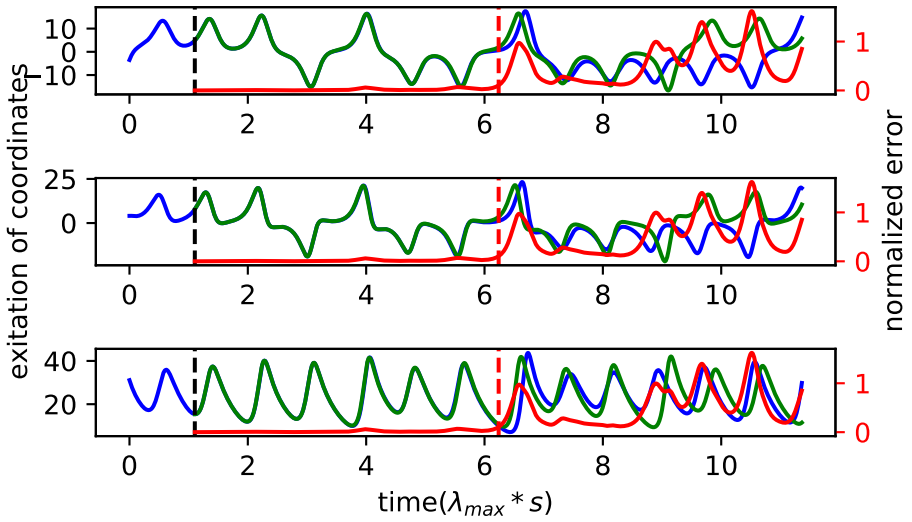


Figure 6: Visualisation of $x$-(*top*), $y$- (*middle*) and $z$-coordinates (*bottom*) of the Lorenz attractor: In blue the numerically integrated trajectory of the different coordinates is shown. The prediction of the network is depicted in green and the normalized error (comp. Eq. 14) in red on a differently scaled axis (right). The dotted black line marks the beginning of the prediction, while the dotted red line marks the end of the valid period.

Additionally, the normalized error is included in all three plots. The prediction starts at the black dashed line and the valid region of the prediction ends at the red dashed line. The valid time of the portrayed prediction is $\lambda_{max}t_{valid} = 5.13 \pm 0.01$ (in Lyapunov time).

### 3.2.2 Evaluation of a set of hyperparameters

For the evaluation of a set of hyperparameters, the valid time for $k$ different ESNs is computed for $l$ different random initial conditions. From this data set a mean and standard error is calculated. The mean valid time is used as a measure for the quality of the set of hyperparameters. To find high performing sets of hyperparameters, here a *grid-search* method was used, calculating the mean valid times on a mesh grid of different sets of hyperparameters. The hyperparameters in Table 4 are then chosen, because they seem to be the best trade off between computation time needed (mainly influences the choice of $N$[8]) and performance.

Figure 7 (*left*) exemplary portrays the dependence of (mean) valid times on the number of inner Nodes $N$ and the regularization constant $\beta$, for the other hyperparameters chosen as depicted in Table 4.

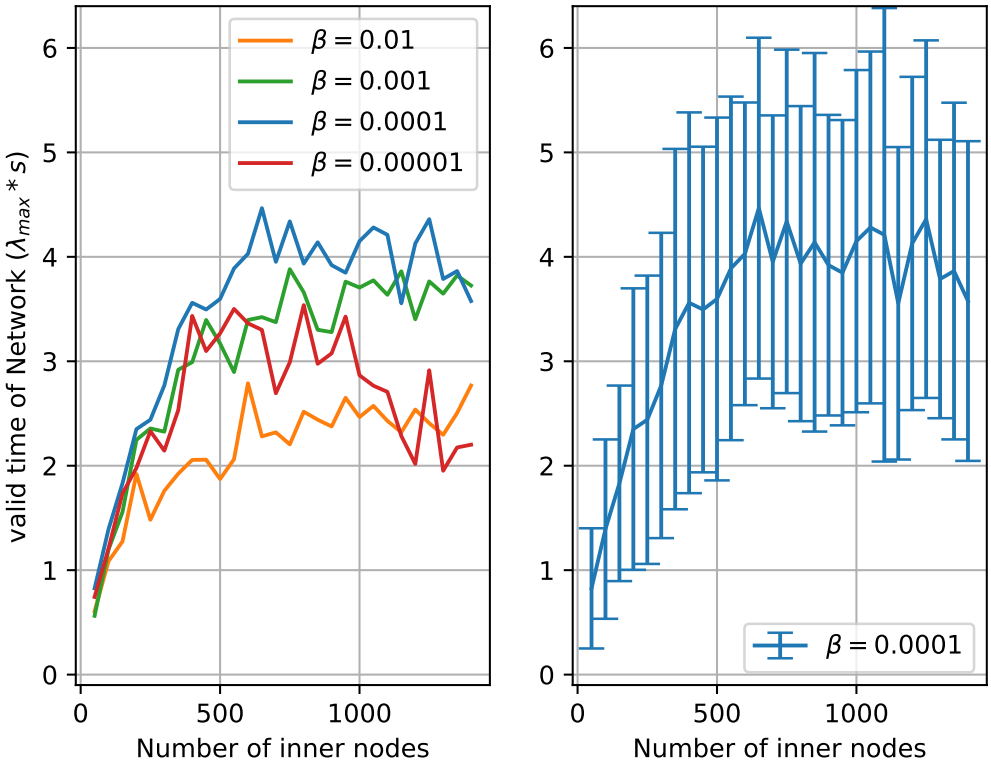


Figure 7: Evaluation of the dependence of average valid times on the number of inner nodes $N$ for different regularization constants $\beta$. The other hyperparameters are chosen according to Table 4.

[8]Obviously, the computation time is also directly influenced by the number of training steps $T$, which is not considered as a hyperparameter.

For each set of hyperparameters it is averaged over $k = 5$ ESNs, each tested on $l = 20$ initial conditions. To avoid the clutter, the standard error is omitted in the plot with multiple regularization constants and only shown in Figure 7 (*right*) for the chosen regularization constant $\beta = 0.0001$. Even though a great number of training steps $T = 50000$ is chosen here, the effects of overfitting are visible for small regularization constants and great numbers of inner nodes (comp. Figure 7 (*left*), $\beta = 0.00001$). In addition to that, also great regularization constants worsen the prediction especially for small numbers of inner nodes, as depicted for $\beta = 0.01$.

In difference to the chosen set of hyperparameters, also the choice of $\beta = 0.001$ is useful if training time is needed to be shortened, as it allows the use of smaller numbers of training steps (compare with visualizations in *Gitlab* folder).

# 4 Future work

Having presented both approaches individually for selected examples, attention can be directed to the overall goal of applying hybrid approaches models of cardiac dynamics to predict time series.

Therefore the data-driven approach is to be applied to spatially extended systems. However, a problem of using just one ESN for the prediction of the entire state space is that this method leads to huge amounts of memory and computation time needed, due to the high dimensionality of the spatially discretized systems (compare with Chapter 2). A possible solution is presented by Zimmermann and Parlitz in [11] and shall be used. Their proposal takes advantage of the local structure of the dynamics by using many ESNs, which are driven only by local input data to generate predictions for the next time step as output.

Further, the subsequent goal is to combine the two methods in such a way that they advantageously complement each other. An example of such is given by Pathak *et al.* in [1], showing that hybrid approaches can drastically reduce the amount of training data needed or even make longer valid times possible. One of the main issues to be addressed is the application of hybrid approaches to models of cardiac dynamics. Therefore the suggestions Pathak *et al.* present might be used or even better ways of combining knowledge and data based approaches can be derived. Another major point of interest is the extension of hybrid approaches to incomplete models with complete dynamical variables[9] missing.

[9] This is expressed through the lack of complete (sets of) differential equations.

# References


[1] Jaideep Pathak, Alexander Wikner, Rebeckah Fussell, Sarthak Chandra, Brian R Hunt, Michelle Girvan, and Edward Ott. Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28(4):041101, 2018.

[2] George Datseris and Ulrich Parlitz. *Nonlinear Dynamics: A Concise Introduction Interlaced with Code*. SPRINGER, 2022.

[3] K. Strehmel, R. Weiner, and H. Podhaisky. *Numerik gewöhnlicher Differentialgleichungen: nichtsteife, steife und differential-algebraische Gleichungen*. Springer Science & Business Media, 2012.

[4] J. Barkley Rosser. Nine-point difference solutions for poisson's equation. *Computers & Mathematics with Applications*, 1(3-4):351–360, 1975.

[5] Flavio H. Fenton Elizabeth M. Cherry. The virtual heart - introduction. http://dev1.thevirtualheart.org/Media/Movies.html; last visited 29.01.2021.

[6] Rubin R Aliev and Alexander V Panfilov. A simple two-variable model of cardiac excitation. *Chaos, Solitons & Fractals*, 7(3):293–301, 1996.

[7] Christopher M Bishop. Pattern recognition. *Machine learning*, 128(9), 2006.

[8] Herbert Jaeger. *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the" echo state network" approach*, volume 5. GMD-Forschungszentrum Informationstechnik Bonn, 2002.

[9] Herbert Jaeger. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148(34):13, 2001.

[10] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR, 2013.

[11] Roland S. Zimmermann and Ulrich Parlitz. Observing spatio-temporal dynamics of excitable media using reservoir computing. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28(4):043118, April 2018.

[12] Herbert Jaeger and Harald Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *science*, 304(5667):78–80, 2004.

[13] Izzet B Yildiz, Herbert Jaeger, and Stefan J Kiebel. Re-visiting the echo state property. *Neural networks*, 35:1–9, 2012.

[14] Mantas Lukoševičius. A practical guide to applying echo state networks. In *Neural networks: Tricks of the trade*, pages 659–686. Springer, 2012.

[15] Mantas Lukoševičius and Herbert Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

[16] Edward N Lorenz. Deterministic nonperiodic flow. *Journal of atmospheric sciences*, 20(2):130–141, 1963.