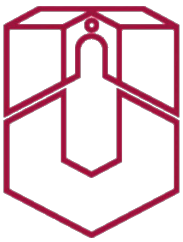

University of Osnabrück
Neuroinformatics Department

A Volterra Series Approach to Liquid State Machines by Application of the Lambert W Function

Bachelor Thesis
by

Rafael Reisenhofer

April 12, 2012 – July 9, 2012

Primary Referee:	Prof. Dr. Gordon Pipa
Secondary Referee:	Prof. Dr. Peter König
Supervisor:	Prof. Dr. Gordon Pipa

Proclamation

Hereby I confirm that I wrote this thesis independently and that I have not made use of any other resources or means than those indicated.

Berlin, July 9, 2012

_____

Rafael Reisenhofer

Contents

List of Figures **5**

1 Introduction **7**

2 Liquid State Machines **9**
2.1 Introduction 9
2.1.1 Discrete and continuous Time 9
2.1.2 Recurrence 10
2.1.3 Stereotypy, Adaptivity and Real-Time Computation 12
2.2 A conceptual and formal Description of Liquid State Machines 12
2.3 The computational Power of a Liquid State Machine realized by recurrent Circuits 15
2.3.1 Time invariant fading Memory Operators and recurrent Circuits . . . 15
2.3.2 Two mild Conditions 16
2.3.3 A Liquid State Machine can simulate any Time invariant fading Memory Operator 17

3 Liquid State Machines and Delay Differential Equations **23**
3.1 Optical computing and Liquid State Machines 23
3.1.1 Circuits of Semiconductor Optical Amplifiers 25
3.2 Realizing a recurrent Liquid with a single Node 26
3.3 Delay Differential Equations and the Method of Steps 30

4 Towards a Volterra Series Approach to Liquid State Machines **33**
4.1 Obtaining Volterra-like Series Representations from linear Ordinary Differential Equations 33
4.2 Obtaining Volterra-like Series Representations from linear Delay Differential Equations 37
4.3 Discussion 40

References **43**

List of Figures

2.1 Schematic depiction of a feedforward network and a recurrent neural network 11
2.2 Example of a liquid state machine realized by neurons. 13

3.1 The steady-state transfer function of a semiconductor optical amplifier compared to the *tanh* function. 25
3.2 The swirl topology for photonics-based liquid state machines. 26
3.3 Sketch of a single node liquid state machine. 27
3.4 The discretization operator J 29
3.5 Solution of a delay differential obtained by the method of steps 31

4.1 Plots of the Lambert W function. 38

Chapter 1

Introduction

The aim of this thesis is to find a method for deriving explicit input-output representations in the form of Volterra series expansions for dynamical systems implicitly defined by delay differential equations[1]. The motivation behind this task is to gain a better understanding of a specific realization of the liquid state machine framework [1], which is currently being explored by a large international scientific endeavor called PHOCUS[2]. The defining feature of this realization is that it requires only a single dynamical node equipped with delayed feedback to represent a complex recurrent network topology, a concept that was only recently proposed by Appeltant et al. [16]. In the remainder of this short introduction I will give a brief overview of the contents of the upcoming chapters.

In chapter 2, the framework of liquid state machines is introduced and motivated. Furthermore, section 2.3 is devoted to a detailed examination of the fundamental result concerning the computational power of liquid state machines, originally published by Maass and Markram in 2004 [5].

Chapter 3 discusses how the potential computational power of liquid state machines might be exploited by artificial implementations, particularly within the paradigm of optical computing. This will lead to a description of how a liquid state machine can be realized using only a single optical node whose dynamics are governed by a delay differential equation in section 3.2.

In chapter 4, I finally try to establish the desired connection between the implicit description of a dynamical system given by delay differential equations and the explicit description provided by a Volterra series expansion. My efforts are essentially based on the work of Asl et al. who in 2003 proposed a novel method for analytically solving linear delay differential equations [31]. We will see in section 4.2 that this method can in fact be utilized to derive an explicit input-output representation based on integral convolutions from a linear delay differential equations. While said representation shares some similarities with a proper Volterra series expansion, it unfortunately also possesses some undesirable properties that will briefly be discussed at the very end of this thesis.

[1] For a profound introduction to the Volterra series approach to dynamical systems and delay differential equations see [28] and [26].

[2] see `http://ifisc.uib-csic.es/phocus/`

Chapter 2

Liquid State Machines

2.1 Introduction

A liquid state machine (LSM) is a framework used for the computational modelling of the dynamics of cortical microcircuits, originally proposed by Maass, Natschläger and Markram in 2002 [1]. By applying this framework, it is possible to combine well established features of biological neural circuits that have proven to be surprisingly incompatible within computational models while maintaining a biologically realistic neuron model. These properties include continuity with respect to time, adaptivity, recurrence, stereotypy and the ability of real-time computing. The upcoming paragraphs will be devoted to a short analysis of these properties. I will discuss the motivating biological findings and the respective implications for computational models of neural microcircuits.

2.1.1 Discrete and continuous Time

It goes without saying that information is received and processed by human brains over continuous time. In particular, biological neurons are not synchronized by an internal clock as they are necessarily in discrete-time models. Hence, any model claiming to be a reasonable candidate for an accurate description of natural neural mechanisms has to allow for continuous transitions between states and incorporate continuous state spaces. Nonetheless, the fundamental building block of classical neural models, the so called McChulloch-Pitts Neuron [2] is defined under the assumption of discrete time.

Definition 2.1.1 (McCulloch Pitts Neuron)**.** *Let $n \in \mathbb{N}$ and $(N_i)_{i=1}^n$ be a set of neurons with $N_i \colon \mathbb{N} \to \{0,1\}$ for all $i \leq n$ and $(w_{ji})_{i,j=1}^n \subset \mathbb{R}$ a familiy of weights (synapses), then the value of the neuron N_j at time $t \in \mathbb{N}$ is given by:*

$$N_j(t) = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_{ji} N_i(t-1) > \phi_j \\ 0 & \text{else} \end{cases}$$

where $(\phi)_{i=1}^n \subset \mathbb{R}$ is a familiy of thresholds.

Treating neural networks as time-discrete systems may be a substantial simplification of what's really going on. Still, there are some notable advantages. The eminent success of feedforward

networks in the field of pattern recognition is mostly owed to learning algorithms based on backpropagation [3], and thereby reliant on both non-recurrence and discrete time. Furthermore, a neural network modeled over discrete time can be interpreted as a kind of finite state automaton. This observation inspired researchers to tackle the question of the potential computational power of neural networks by applying concepts of general automata theory, a well studied field in computer science. In fact, it has been shown that every possible Turing machine and thereby any digital computer can be simulated by a suitable artificial neural network [4]. Taken together, a time-discrete point of view seems to simplify learning mechanisms and facilitates a rigorous mathematical analysis of the properties of neural networks within the framework of automata theory. Also, it is in accordance with the still popular notion that the mechanisms of our brain are in principle comparable to those of a digital computer.

If we aim for a higher degree of biological realism and decide to explore the dynamics of neural circuits in the continuous time domain, both the description and the analysis of a developed model change significantly. Instead of simply defining a mapping from one state to another (see for example 2.1.1), a description of infinitesimal changes has to be provided by differential equations or related mathematical expressions like integral equations. Accordingly, inputs and outputs are no longer modeled as sequences of state descriptions but as possibly continuous functions over time. The computational power of a model is then determined by the set of mappings between continuous function spaces it can simulate. These considerations are of great importance to the understanding of this thesis, as its major goal is to develop a mathematical description of liquid state machines by using Volterra series expansions (see chapter 4). Also, I will state and derive the fundamental result on the computational power of liquid state machines, published by Maass and Markram in 2004 [5] in section 2.3.

2.1.2 Recurrence

The networks formed by mammalian brains are dense and highly connected. The cerebral cortex of a mouse is about $180\,\mathrm{mm}^3$ in size and consists of a total of 20 million neurons with an average of 8000 synapses per neuron. It was proposed that the probability of two pyramidal cells being connected within the mouse cerebral cortex is no less than $10\,\%$ if they have a distance of $0.1\,\mathrm{mm}$ [6]. While it is given that an anatomical connection can be established between any two parts of the brain, the above figures suggest something much more fundamental, namely that a high level of recurrence can already be found within cortical microcircuits.

This being said, we should realize that the presence of recurrent structures is of great consequence to the behavior and properties of the dynamical system induced by a neural network. Most notably, positive feedback loops are a simple and effective way to provide a system with the capability of short-term memory. Loosely speaking, the information passed to a recurrent neural network via the input layer does not necessarily traverse it in a strictly straightforward fashion as it is the case in classical feedforward networks and can hence be stored[1](see fig. 2.1). Another feature of recurrent networks is their disposition to exhibit chaotic behavior. This means that similar inputs can be mapped to very distant points in the state space along deterministic but seemingly chaotic trajectories. Such a behaviour results in a high sensitivity to noise but can also be desirable in terms of feature extraction.

[1]Please note that the capability of short-term memory is not restricted to recurrent neural networks. It can also be modeled in feedforward networks by using time-delay neurons. See chapter 13 in [7].

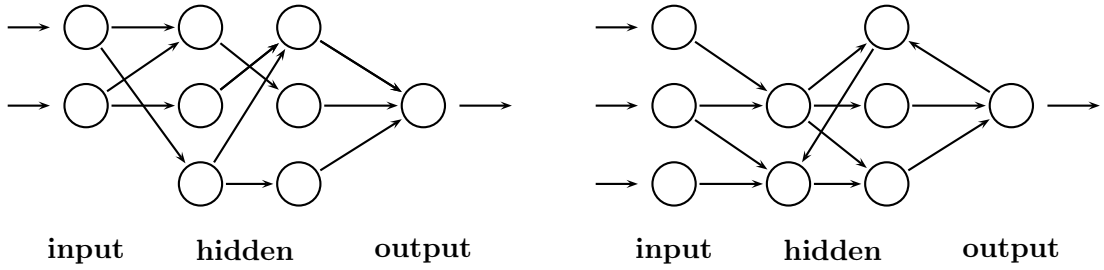

Figure 2.1: Schematic depiction of a typical feedforward network with two hidden layers (left) and a recurrent neural network (right).

So how can the complex dynamics of recurrent neural networks be computationally utilized? Since the state of a recurrent neural network may vary significantly over time, simply considering the values of output neurons at a specific point in time might be insufficient. The most common approach to this issue is to exploit the fact that despite their highly complex dynamics, recurrent networks can have equilibria, stable states and attractors. A state vector $\tilde{\mathbf{x}} \in S$ within an arbitrary state space S is said to be an equilibrium if it does not change with respect to time, that is:

$$\frac{d\tilde{\mathbf{x}}(t)}{dt} = \mathbf{0} \tag{2.1}$$

If additionally vectors in the vicinity of $\tilde{\mathbf{x}}$ are always on trajectories contained in a neighborhood of $\tilde{\mathbf{x}}$, $\tilde{\mathbf{x}}$ is called a stable state. A formal mathematical description resembles the famous epsilon-delta criterion for continuity:

$$\forall \epsilon > 0 \colon \exists \delta > 0 \text{ such that } \|\mathbf{x}(0) - \tilde{\mathbf{x}}\| < \delta \Rightarrow \|\mathbf{x}(t) - \tilde{\mathbf{x}}\| < \epsilon \quad \forall t > 0 \tag{2.2}$$

This notion of stability can be tightened further if we only consider stable states around which a region of initial vectors exists all of which converge to the stable state as time increases. Such a state is called a point attractor. The concept of attractors generalizes naturally to arbitrary manifolds embedded in a state space and can for instance be used to model associative memories: By training an artificial neural network in such a way that a specific vector becomes an attractor, similar initial states will converge over time and are thereby associatively linked to the attractor. One of the most famous and successful models applying these concepts is the Hopfield network, proposed by John Hopfield in 1982 [8].

In conclusion, the canonical way of dealing with the dynamics of a recurrent network is to control its stability behavior. This task is manageable but theoretically and computationally demanding, especially if a network is based on biologically more realistic neuron models like the integrate-and-fire model (a Hopfield network consists of McChulloch-Pitts neurons). Furthermore, the time attractor neural networks need to converge makes them generally unsuited for real-time computations. Hence, the ability of utilizing the complex dynamics of recurrent networks without relying on stable states or attractors can undoubtedly be regarded one of the major merits of liquid state machines. How and why this works will be explained in detail in sections 2.2 and 2.3.

2.1.3 Stereotypy, Adaptivity and Real-Time Computation

Even in the early days of anatomical neuroscience people like Rafael Lerente de Nó (1902 - 1990) believed that the synaptic organization of neural microcircuits in the neocortex was governed by a stereotypical basic pattern [9, p. 522]. Though it seems unlikely that such a universal structure really exists, recent research indicates a surprisingly high degree of stereotypy. Silberberg et al. suggested in 2002 that the neocortical microcircuitry is in fact "strictly stereotypical provided that the same species, region, area and age are considered" [10]. This would imply that biological neural microcircuits have not evolved to fulfill specific purposes but that there is a class of microcircuits in our brains capable of either performing or at the very least supporting the computation of arbitrary functions. It is again a key property of the concept of liquid state machines that it gives a realistic idea of how even small recurrent neural circuits can be equipped with such a universal computational power.

The two defining and by far most important functional properties of human brains are their capability of real-time computation and their adaptivity. We can tell in the fraction of a second if the girl standing next to us in a queue is someone familiar or just a nameless stranger. After having our tongue burnt by drinking too hot coffee, we are likely to just cautiously sip when having the next one. If we weren't able to process huge amounts of sensory information in practically no time and adapt our behaviour to fit the specificities of our surroundings, surviving would be difficult. Though there has been a strong focus on adaptivity, artificial neural networks have been designed to posses both these features. But very rarely have they accounted for both of them at the same time while maintaining biologically reasonable network structures and neuron models, as it is the case with liquid state machines.

2.2 A conceptual and formal Description of Liquid State Machines

On a strictly conceptual level, a liquid state machine consists of two functionally distinct components. One is the so called liquid (sometimes also referred to as the reservoir), an entity that receives inputs (we could also regard them as perturbations) over continuous time and thereby generates an internal state at each point in time that is subject to continuous change and hence designated the liquid state. It is important to note that this internal state cannot be computed as a function of the current input but may also depend on all former perturbations. The second component of a liquid state machine can be described as a set of readout devices that map the liquid state to specific values at each point in time. These readout devices are allowed to consider and combine all elements of a possibly high dimensional liquid state with the one restriction that they're not endowed with any kind of memory. It might already be apparent that liquid state machines aim to implement a kind of "division of labor" approach: While the liquid is confronted with raw perturbations and capable of memory to handle inputs with dependencies extended in the time domain, the readout maps are supposed to make sense of whatever information the liquid state contains at a single point in time and are kept deliberately simple. In the following paragraph, I will try to give an idea of how modeling neural networks as liquid state machines might lead to a system equipped with all the desirable features mentioned in section 2.1.

The liquid of a liquid state machine can be implemented by a recurrent network of integrate-

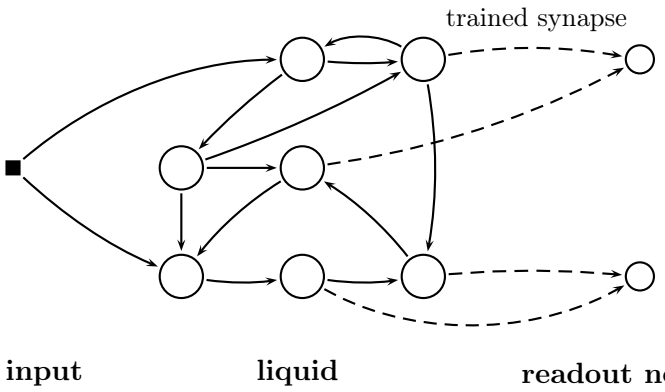

Figure 2.2: Example of a liquid state machine realized by neurons. The liquid is a recurrent neural network and subject to continuous perturbations by an external input. Readout neurons are connected to selected nodes within the liquid. These connections are drawn as dashed lines and represent the only synapses being optimized during a learning routine. Note that this implies that the structure of the recurrent network and the weights of its synapses can essentially be chosen randomly.

and-fire neurons where some of these neurons are exposed to time varying inputs, also represented by spike trains. The readout devices can simply be realized by neurons that have static connections to some elements within the liquid. The basic idea is then to restrict learning algorithms to finding the best weights for the synapses connecting the liquid with the output neurons, which turns out to be a rather trivial task because the thereby induced subnetwork is only two-layered, static and without recurrence. As we will see in section 2.3, this restriction doesn't diminish the potential computational power of a liquid state machine. Hence, this stunt renders all efforts of controlling or adapting the highly nonlinear dynamics of the recurrent network obsolete. In fact, it might even be desirable to choose a liquid that's reacting almost chaotically to perturbations in order to make even small differences in the input well distinguishable from the point of view of a readout neuron. Additionally, in such a setting there is no need to wait until the liquid has reached a stable state, hence real-time computations seem perfectly possible. Furthermore, as the liquid can be chosen independently of a specific task, considering neural networks as liquid state machines also accounts for stereotypy. This accumulation of desirable properties suggests that liquid state machines are nicely suited for real-time computations and hence raises the question of their potential computational power. It was proved by Maass and Markram that a spiking neural network modeled according to the concept of liquid state machines can approximate any time invariant mapping from spike trains to the space of real-valued functions on $\mathbb{R}$, when some mild conditions on the behaviour of the liquid and the readout neurons are imposed and the considered mappings have a so called fading memory property [5]. This is an extension of the result given in section 2.3.

The term "liquid state machine" emphasizes both the structural similarity to finite state machines and the fact that liquid states are subject to continuous change. It also suggests a way of vividly realizing the concept of liquid state machines in an everyday environment. Let's imagine a group of people standing on a lakeside. Someone is swimming in a certain distance of the group. It will be almost impossible for a member of the group to recognize certain features of the swimmer like the size of his hands, the frequency of his foot stroke or his precise speed by just looking at him. On the other hand, every move made by the swimmer

causes perturbations in the water, resulting in a complex spectrum of waves traveling towards the lakefront. It is a reasonable assumption that all information about the features mentioned earlier is contained somewhere in the shapes, magnitudes and the composition of these waves. If we imagine that for each feature, one member of the group is experienced enough to give a sensible estimate by observing the clearly visible perturbations in the lake, it is easy to see that this setting actually realizes a liquid state machine. The swimmer can be regarded as an input, causing continuous perturbations in the liquid that consists of the lake's water while every expert in the group represents a readout device.

By now, I have established liquid state machines conceptually and given two examples of concrete realizations. In order to perform a rigorous analysis of the computational properties and the computational power of liquid state machines, a mathematical description is required. As it was already pointed out in section 2.1, the fact that liquid state machines operate in the continuous time domain is of great consequence to this description. If we consider the input of a liquid state machine and the development of its internal state as functions of time and recognize that the liquid state can depend on all past and present inputs and the properties of the liquid, it is obvious that the liquid can be described as a mapping between two function spaces. The readout devices on the other hand are memory-less and map a current liquid state to single values. Hence, the complete readout map essentially defines a mapping between two real vector spaces. This leads to the following formal definition of a liquid state machine.

Definition 2.2.1 (Liquid state machine)**.** *Let $U \subset \mathbb{R}^{\mathbb{R}}$ be a subset of the space of all real-valued functions of time, $n, k \in \mathbb{N}$ and the operator F be an arbitrary mapping between U^n and $(\mathbb{R}^{\mathbb{R}})^k$, that is $F\colon U^n \to (\mathbb{R}^{\mathbb{R}})^k$. Let $l \in \mathbb{N}$ and $f\colon \mathbb{R}^k \to \mathbb{R}^l$ be a function between two finite dimensional real vector spaces, then the pair*

$$M := \langle F, f \rangle \tag{2.3}$$

is called a liquid state machine.

Let $u \in U^n$ be an input function, then the liquid state $x_M(t) \in \mathbb{R}^k$ and the value $y_M(t) \in \mathbb{R}^l$ of M at time $t \in \mathbb{R}$ are given by:

$$x_M(t) = (Fu)(t) \tag{2.4}$$
$$y_M(t) = f(x_M(t)) \tag{2.5}$$

I will conclude this section with two short remarks. The operator F is referred to as a filter in the original publications [1, 5]. While keeping the designation F, I will refrain from using the term filter to avoid any confusion concerning its long-established uses in the fields of signal processing and mathematics. Furthermore, it should be noted that most of the ideas underlying the framework of liquid state machines were independently developed and published under the name of echo state networks by Herbert Jaeger in 2001 [11] with the main difference being that echo state networks are set in the discrete time domain. Due to the substantial similarities, both frameworks have been subsumed under the concept of reservoir computing[2].

[2] see `http://reservoir-computing.org/`

2.3 The computational Power of a Liquid State Machine realized by recurrent Circuits

This section aims to show that a liquid state machine implemented by a neural network can simulate a large class of operators between function spaces. In the final step of our considerations, we will use the famous Stone-Weierstrass theorem to prove that the set of all operators computable by a liquid state machine is dense in the set of all time invariant operators satisfying the so-called fading memory property, given that certain conditions are fulfilled by the liquid and the readout map. However, before we are able to do this, there is some preliminary work ahead of us. First, we will specify what it means for an operator to exhibit the fading memory property and to be time invariant. The next step will be to show that the computations performed by a recurrent circuit can in fact be described by such an operator. Finally, we will define two mild conditions for liquid state machines and show that these conditions are sufficient to create a setting satisfying the prerequisites of the Stone-Weierstrass theorem.

Please note that in contrast to the realization proposed in section 2.2, the input will be assumed as a continuous function and not a spike train. In fact, we will require input functions to be uniformly bounded and to be Lipschitz continuous with a common Lipschitz constant. Also, we will consider the liquid to be an arbitrary recurrent circuit instead of a network of spiking neurons. These simplifications ease the mathematical analysis without altering the overall statement. The extension to networks of spiking neurons can be found in [5, p. 14].

2.3.1 Time invariant fading Memory Operators and recurrent Circuits

I have already stated that the results in this section will be restricted to operators that are both time invariant and equipped with a fading memory. An operator is called time invariant if a temporal shift in the input simply results in the exact same temporal shift in the output. We could also describe the class of time invariant operators as the class of operators that are independent of an internal clock.

Definition 2.3.1 (Time invariance)**.** *Let E be an arbitrary space of functions of time, $f \in E$ and $\tau \in \mathbb{R}$. The translation operator T_τ is then given by:*

$$(T_\tau f)(t) = f(t - \tau) \tag{2.6}$$

for all $t \in \mathbb{R}$. Let $n, k \in \mathbb{N}$ and $F\colon (\mathbb{R}^{\mathbb{R}})^n \to (\mathbb{R}^{\mathbb{R}})^k$ be a mapping between vectors composed of real-valued functions of time. F is said to be time invariant if it holds that:

$$F(T_\tau u) = T_\tau(Fu) \tag{2.7}$$

for all $u \in (\mathbb{R}^{\mathbb{R}})^n$ and $\tau \in \mathbb{R}$.

Furthermore, an operator is said to posses a fading memory if the value of an output function at a certain point in time can be approximated arbitrarily well by images of functions that have not significantly deviated from the original input function in the recent past. This essentially means that the value of an input function at time t_1 does not influence the value of its image at time t_2 if t_2 is sufficiently greater than t_1. A formal definition of a fading memory is given below.

Definition 2.3.2 (Fading memory property)**.** *Let $n, k \in \mathbb{N}$ and $F\colon (\mathbb{R}^{\mathbb{R}})^n \to (\mathbb{R}^{\mathbb{R}})^k$ be a mapping between vectors composed of real-valued functions of time. F is said to have the fading memory property if for each $u \in (\mathbb{R}^{\mathbb{R}})^n$ and each $\epsilon > 0$ there exist $\delta > 0$ and $T > 0$ such that:*

$$\Big(\forall v \in (\mathbb{R}^{\mathbb{R}})^k, \forall t \in [-T, 0]\colon \|u(t) - v(t)\| < \delta\Big) \Rightarrow \|(F(u))(0) - (F(v))(0)\| < \epsilon \tag{2.8}$$

In section 2.2, I have introduced a mathematical specification of the liquid state machine framework as well as a realization based on recurrent circuits of spiking neurons. Both are valid descriptions of liquid state machines, however, it's not a priori given that the dynamics of a recurrent circuit can be expressed in terms of definition 2.2.1.

Recurrent circuits of neurons and the operators used in definition 2.2.1 are alike in describing mappings from a space of input functions into a space of output functions. Still, there is one notable difference. While an operator $F\colon (\mathbb{R}^{\mathbb{R}})^n \to (\mathbb{R}^{\mathbb{R}})^k$ maps values of an input function $u \in (\mathbb{R}^{\mathbb{R}})^n$ to values of an output function $v \in (\mathbb{R}^{\mathbb{R}})^k$ at each point in time, a circuit C is generally assumed to start doing so at a specific point in time. Hence, the mapping defined by a recurrent circuit is not only dependent on the structure of the circuit, but also on its initial state at an initial point in time t_0. The following theorem illustrates that this difference vanishes, if we only consider operators and circuits that have a fading memory.

Theorem 2.3.3. *Let $x_0 \in \mathbb{R}^k$ with $k \in \mathbb{N}$ be the initial value of a circuit C at time $t_0 \in \mathbb{R}$. Let $C(x_0, u|_{[t_0,t]}, t) \in \mathbb{R}^k$ denote the output of C at time $t \in [t_0, \infty)$ regarding an input function $u \in (\mathbb{R}^{\mathbb{R}})^n$ with $n \in \mathbb{N}$ where $u|_{[t_0,t]}$ is the restriction of u to $[t_0, t]$.*

If C has the fading memory property (see definition 2.3.2) then $F_C\colon (\mathbb{R}^{\mathbb{R}})^n \to (\mathbb{R}^{\mathbb{R}})^k$ given by

$$(F_C(u))(t) = \lim_{t_0 \to -\infty} C(x_0, u|_{[t_0,t]}, t) \tag{2.9}$$

is a well defined fading memory operator.

Proof. It follows directly from the fading memory property of C that $\lim_{t_0 \to -\infty} C(x_0, u|_{[t_0,t]}, t)$ converges uniformly. This implies, by some technicalities, that F_C also has a fading memory. For a detailed proof see [5, p. 10]. □

2.3.2 Two mild Conditions

Theorem 2.3.3 tells us that if the liquid of a liquid state machine consists of a recurrent circuit possessing the fading memory property, it uniquely defines a corresponding fading memory operator. This can now be exploited to perform a rigorous mathematical analysis of the computational power of such a liquid state machine. First, I'd like to point out that according to definition 2.2.1, the operator defined by the liquid of a liquid state machine maps between vectors of real-valued functions of time. Hence, it could also be considered a set of basis operators that map vectors of real-valued functions of time to a single real-valued function of time. Formally, this can be expressed by the following definition.

Definition 2.3.4. *Let M be a liquid state machine (see definition 2.2.1) and its liquid F_M be a mapping from $(\mathbb{R}^{\mathbb{R}})^n$ to $(\mathbb{R}^{\mathbb{R}})^k$ with $k, n \in \mathbb{N}$. Let $(F_M(g))_i$ denote i-th component of $F_M(g) \in (\mathbb{R}^{\mathbb{R}})^k$, then the corresponding basis operator B_i is given by*

$$B_i\colon (\mathbb{R}^{\mathbb{R}})^n \to \mathbb{R}^{\mathbb{R}}\colon g \mapsto (F_M(g))_i \tag{2.10}$$

for all $i \leq k$ and we call $B_M = \{B_1, \ldots, B_i\}$ the set of basis operators of M.

From this point of view, it would be interesting if there exists a set of basis functions such that arbitrary operators mapping into the space of real-valued functions of time can be simulated by simply choosing a fitting readout map. Note that this would indeed justify the earlier made claim that restricting learning algorithms to the synapses of readout neurons would not diminish the potential computational power of a liquid state machine. In fact, such sets of basis operators not only exist, we can also name a condition that suffices to guarantee the universal computational power of a liquid state machine regarding time invariant fading memory operators. I have already stated that the decisive step in this proof will be to apply the Stone-Weierstrass theorem (see 2.3.7). Hence, it won't come as a surprise that one part of said condition is for the set of basis operators of a liquid state machine to have the pointwise separation property.

Definition 2.3.5 (Pointwise separation property)**.** *Let M be a liquid state machine, B_M the set of its basis operators (see definition 2.3.4), $U \subset (\mathbb{R}^{\mathbb{R}})$ be a set of input functions and $n \in \mathbb{N}$. B_M is said to have the pointwise separation property with respect to U^n if for all $u, v \in U^n$ there exists $B \in B_M$ such that:*

$$(\exists s \leq 0 \colon u(s) \neq v(s)) \Rightarrow (B(u))(0) \neq (B(v))(0) \tag{2.11}$$

If the set of basis operators of a liquid state machine satisfies the pointwise separation property, the Stone-Weierstrass theorem can be used to show that the set of all polynomials in the basis operators is dense in a larger space of operators. However, this result is only meaningful if these polynomials can also be computed by the respective liquid state machine. We will see that this is in fact the case, if we are free to choose readout maps from a class of functions satisfying a certain approximation property.

Definition 2.3.6 (Universal approximation property)**.** *A class of real-valued functions A is said to have the universal approximation property if it holds that for every $n \in \mathbb{N}$, every compact set $X \subset \mathbb{R}^n$, every function $f \in C(X, \mathbb{R})$ and every $\epsilon > 0$ exists a function $g \in A$ such that:*

$$|g(x) - f(x)| < \epsilon \tag{2.12}$$

for all $x \in X$, where $C(X, \mathbb{R})$ denotes the space of all continuous functions from X to $\mathbb{R}$.

Note that the universal approximation property just defined is slightly related to but not to be mistaken with the approximation property for Banach spaces which requires that every compact operator that maps into a specific Banach space is the limit of a sequence of linear operators with finite rank [12, p. 88]. I'd also like to emphasize that the universal approximation property is not a condition for the readout map of a liquid state machine but for the class of functions it can be constructed from. Hence, it is a far less restrictive requirement than one might think.

2.3.3 A Liquid State Machine can simulate any Time invariant fading Memory Operator

As I have already mentioned, we will assume input functions to be uniformly bounded and to be Lipschitz continuous with a common Lipschitz constant. Formally, we define the space

$U_{K,L}$ of input functions for constants $K, L > 0$ as follows:

$$U_{K,L} = \{u \in \mathbb{R}^{\mathbb{R}} \colon |u(s)| \leq K \text{ and } |u(s) - u(t)| \leq L\,|s - t| \text{ for all } s, t \in \mathbb{R}\} \tag{2.13}$$

We will now show that a liquid state machine can approximate any time invariant fading memory operator defined on the space of input functions $(U_{K,L})^n$ if it satisfies the conditions given in section 2.3.2, namely that the liquid has the pointwise separation property and the readout maps can be chosen arbitrarily from a class equipped with the universal approximation property. Let's first have a look at a formulation of the Stone-Weierstrass theorem.

Lemma 2.3.7 (Stone-Weierstrass)**.** *Let E be a compact metric space and $C(E, \mathbb{R})$ the space of all continuous functions from E to $\mathbb{R}$. Let $B \subset C(E, \mathbb{R})$ be a set that separates points in E, that is:*

$$a \neq b \Rightarrow \exists f \in B \colon f(a) \neq f(b) \tag{2.14}$$

for all $a, b \in E$. Then for every $f \in C(E, \mathbb{R})$ there is a sequence $(g_n)_{n\in\mathbb{N}} \subset C(E, \mathbb{R})$ of polynomials in the functions of B with real coefficients converging uniformly to f.

Proof. See for example [12, p. 427] or [13, p. 137], where this somewhat uncommon formulation of the Stone-Weierstrass theorem can also be found. □

First, we will show that any time invariant fading memory operator mapping from the input space $(U_{K,L})^n$ to $\mathbb{R}^{\mathbb{R}}$ can in fact be identified by a functional mapping from a certain space $(U^-_{K,L})^n$ to $\mathbb{R}$, where $(U^-_{K,L})^n$ is a compact metric space. Without loss of generality, we assume, from now on, $n = 1$.

Lemma 2.3.8. *Let $U^-_{K,L}$ be the set of all functions in $U_{K,L}$ restricted to the domain $(-\infty, 0]$, that is:*

$$U^-_{K,L} = \left\{u|_{(-\infty,0]} \colon u \in U_{K,L}\right\}. \tag{2.15}$$

Then

$$d(\cdot,\cdot) \colon U^-_{K,L} \times U^-_{K,L} \to \mathbb{R} \colon (u_1, u_2) \mapsto \sup_{t\leq 0} \frac{|u_1(t) - u_2(t)|}{1 + |t|} \tag{2.16}$$

defines a metric on $U^-_{K,L}$ and $(U^-_{K,L}, d)$ is a compact metric space.

Proof. We choose arbitrary $u_1, u_2, u_3 \in U^-_{K,L}$ and verify the metric axioms:

1. $d(u_1, u_2) = 0 \Leftrightarrow u_1 = u_2$

$$\begin{aligned}
\sup_{t\leq 0} \frac{|u_1(t) - u_2(t)|}{1 + |t|} = 0 \quad &\Leftrightarrow \\
\sup_{t\leq 0} |u_1(t) - u_2(t)| = 0 \quad &\Leftrightarrow \\
u_1(t) = u_2(t) \qquad \forall t \leq 0 \quad &\Leftrightarrow \\
u_1 = u_2 &
\end{aligned}$$

2. $d(u_1, u_2) = d(u_2, u_1)$

   This simply follows from $|u_1(t) - u_2(t)| = |u_2(t) - u_1(t)|$ for all $t \leq 0$.

3. $d(u_1, u_3) \leq d(u_1, u_2) + d(u_2, u_3)$

   We know that $|\cdot|$ satisfies the triangle inequality, hence:

$$\begin{aligned} d(u_1, u_3) &= \sup_{t \leq 0} \frac{|u_1(t) - u_3(t)|}{1 + |t|} \\ &= \sup_{t \leq 0} \frac{|u_1(t) - u_2(t) + u_2(t) - u_3(t)|}{1 + |t|} \\ &\leq \sup_{t \leq 0} \frac{|u_1(t) - u_2(t)| + |u_2(t) - u_3(t)|}{1 + |t|} \\ &= \sup_{t \leq 0} \frac{|u_1(t) - u_2(t)|}{1 + |t|} + \sup_{t \leq 0} \frac{|u_2(t) - u_3(t)|}{t + |t|} \\ &= d(u_1, u_2) + d(u_2, u_3) \end{aligned}$$

For compactness, we will show that any sequence in $U^-_{K,L}$ has a convergent subsequence.

Let $(u_i)_{i \in \mathbb{N}} \subset U^-_{K,L}$ be an arbitrary sequence. We first consider the restriction of $U^-_{K,L}$ to a compact set $[-n, 0]$ with $n \in \mathbb{N}$, defined by:

$$U^-_{K,L}|_{[-n,0]} = \{u|_{[-n,0]} \colon u \in U^-_{K,L}\} \tag{2.17}$$

By definition, $U^-_{K,L}|_{[-n,0]}$ is uniformly bounded and equicontinuous. Furthermore, $[n, 0] \subset \mathbb{R}$ is compact. Hence, by the Arzela-Ascoli theorem (see e.g. [13, p. 142]), $U^-_{K,L}|_{[-n,0]}$ is compact in the space of all continuous functions from $[-n, 0]$ to $\mathbb{R}$ denoted $C([-n, 0], \mathbb{R})$. In particular, $U^-_{K,L}|_{[-1,0]}$ is compact. Thus, there exists an index set $\mathbb{N}_1 \subset \mathbb{N}$ and a function $\tilde{u}_1 \in U^-_{K,L}|_{[-1,0]}$ such that $\tilde{u}_1$ is the limit of the sequence $(u_i)_{i \in \mathbb{N}_1} \subset (u_i)_{i \in \mathbb{N}}$ on $[-1, 0]$, that is:

$$\lim_{\substack{i \in \mathbb{N}_1 \\ i \to \infty}} \left(\sup_{t \in [-1,0]} |u_i(t) - \tilde{u}_1(t)| \right) = 0 \tag{2.18}$$

By the same principle, there exists a function $\tilde{u}_2 \in U^-_{K,L}|_{[-2,0]}$ and an index set $\mathbb{N}_2 \subset \mathbb{N}_1 \subset \mathbb{N}$ such that $\tilde{u}_2$ is the limit of the sequence $(u_i)_{i \in \mathbb{N}_2} \subset (u_i)_{i \in \mathbb{N}_1} \subset (u_i)_{i \in \mathbb{N}}$ and $\tilde{u}_2$ is an extension of $\tilde{u}_1$, that is $\tilde{u}_2(t) = \tilde{u}_1(t)$ for all $t \in [-1, 0]$. By recursively applying this construction, we get a decreasing sequence of index sets $(\mathbb{N}_n)_{n \in \mathbb{N}}$. Furthermore, this can be seen as a recursive algorithm defining a function $\tilde{u} \in U^-_{K,L}$ that satisfies:

$$\lim_{\substack{i \in \mathbb{N}_n \\ i \to \infty}} \left(\sup_{t \in [-n,0]} |u_i(t) - \tilde{u}(t)| \right) = 0 \tag{2.19}$$

for all $n \in \mathbb{N}$.

Now, we consider a sequence of indexes $(i_n)_{n \in \mathbb{N}} \subset \mathbb{N}$ with:

$$(i)_n \leq (i)_{n+1} \text{ and } (i)_n \in \mathbb{N}_n$$

for all $n \in \mathbb{N}$.

That is, the sequence $(i_n)_{n\in\mathbb{N}}$ is increasing and its n-th element always corresponds to a function $u_n \in U^-_{K,L}$ of which the restriction to the domain $[-n, 0]$ is part of a subsequence converging to $\tilde{u}_n \in U^-_{K,L}|_{[-n,0]}$.

We get:

$$\lim_{n\to\infty} d(u_{i_n}, \tilde{u}) = \lim_{n\to\infty} \sup_{t\leq 0} \frac{|u_{i_n}(t) - \tilde{u}(t)|}{1 + |t|} \tag{2.20}$$

$$= \lim_{n\to\infty} \max\left(\sup_{t\in(-\infty,-n]} \frac{|u_{i_n}(t) - \tilde{u}(t)|}{1 + |t|}, \sup_{t\in[-n,0]} \frac{|u_{i_n}(t) - \tilde{u}(t)|}{1 + |t|} \right) \tag{2.21}$$

$$\leq \lim_{n\to\infty} \max\left(\frac{2K}{1+n}, \sup_{t\in[-n,0]} |u_{i_n}(t) - \tilde{u}(t)| \right) \tag{2.22}$$

$$= 0 \tag{2.23}$$

In (2.22), we used that $U^-_{K,L}$ is uniformly bounded by K. Also note that $\lim_{n\to\infty} \left(\frac{2K}{1+n}\right) = 0$ is implied by the convergence of the harmonic progression and $\lim_{n\to\infty} \sup_{t\in[-n,0]} |u_{i_n}(t) - \tilde{u}(t)| = 0$ is an immediate consequence of (2.19). □

Remark 2.3.9. *The publication this section is based on states that it was an immediate consequence of the Arzela-Ascoli theorem that every sequence in $U^-_{K,L}$ has a convergent subsequence, which would imply the compactness of $U^-_{K,L}$ [5, p. 29]. However, this is not the case as $(-\infty, 0]$ is not a compact space. By the separability of $(-\infty, 0]$, the Arzela-Ascoli theorem only implies that every sequence in $U^-_{K,L}$ has a compactly converging subsequence, which is by no means sufficient for the compactness of $U^-_{K,L}$. The proof above can be found in its original form in the appendix of [14].*

Eventually, we will show that the set of operators computable by a liquid state machine satisfying the conditions given in section 2.3.2 can be identified with a set of functions that is dense in the space of all continuous functions from $U^-_{K,L}$ to $\mathbb{R}$. However, this result will only serve our purpose if every time invariant fading memory operator is uniquely determined by a function in $C(U^-_{K,L}, \mathbb{R})$. The following lemma will show that this is indeed the case.

Lemma 2.3.10. *Let $F\colon U_{K,L} \to \mathbb{R}^{\mathbb{R}}$ be an arbitrary time invariant fading memory operator, then F is uniquely determined by a function $f_F \in C(U^-_{K,L}, \mathbb{R})$.*

Proof. Let f_F be given by:

$$f_F\colon U_{K,L} \to \mathbb{R}\colon u \mapsto (Fu)(0) \tag{2.24}$$

Let $u \in U_{K,L}$ be an arbitrary input function. Due to the time invariance (see definition 2.3.1) of F, we get:

$$(Fu)(t) = (Fu)(0 - (-t)) \tag{2.25}$$

$$= (F(T_{-t}u))(0) \tag{2.26}$$

$$= f_F(T_{-t}u) \tag{2.27}$$

for all $t \in \mathbb{R}$, where T_t is the translation operator. Note that the last step is possible because the space $U_{K,L}$ is closed under translation, that is $u \in U_{K,L} \Rightarrow T_\tau u \in U_{K,L}$ for all $\tau \in \mathbb{R}$. We can conclude that F is uniquely defined the function f_F.

Furthermore, F has a fading memory (see definition 2.3.2) and is therefore causal. This means that an output $(Fu)(t)$ at a specific point in time $t \in \mathbb{R}$ only depends on past and present but not on future inputs. In particular, causality implies that the value $(Fu)(0)$ only depends on values $u(t)$ with $t \in (-\infty, 0]$. Hence, f_F can be restricted to the domain $(-\infty, 0]$, that is:

$$f_F \colon U^-_{K,L} \to \mathbb{R} \colon u \mapsto (Fu)(0) \tag{2.28}$$

It remains to be shown that f_F is continuous. Let $\epsilon > 0$ and $u, v \in U^-_{K,L}$. F has a fading memory, hence there exist $T > 0$ and $\delta > 0$ such that:

$$(\forall t \in [-T, 0] \colon |u(t) - v(t)| < \delta) \Rightarrow |(Fu)(0) - (Fv)(0)| < \epsilon \qquad \Leftrightarrow \tag{2.29}$$

$$\left(\forall t \in [-T, 0] \colon \frac{|u(t) - v(t)|}{1 + |t|} < \frac{\delta}{1 + |t|}\right) \Rightarrow |f_F(u) - f_F(v)| < \epsilon \qquad \Rightarrow \tag{2.30}$$

$$\sup_{t \in [-T,0]} \left(\frac{|u(t) - v(t)|}{1 + |t|}\right) < \frac{\delta}{1 + T} \Rightarrow |f_F(u) - f_F(v)| < \epsilon \qquad \Rightarrow \tag{2.31}$$

$$\sup_{t \in (-\infty,0]} \left(\frac{|u(t) - v(t)|}{1 + |t|}\right) < \frac{\delta}{1 + T} \Rightarrow |f_F(u) - f_F(v)| < \epsilon \qquad \Leftrightarrow \tag{2.32}$$

$$d(u, v) < \delta' \Rightarrow |f_F(u) - f_F(v)| < \epsilon \tag{2.33}$$

with $\delta' = \frac{\delta}{1+T}$. Hence, $f_F \in C(U^-_{K,L}, \mathbb{R})$. □

We are now prepared to state to promised result on the computational power of liquid state machines realized by recurrent circuits, which will also conclude this chapter.

Theorem 2.3.11. *Let $M = \langle B_M, f \rangle$ be a liquid state machine where the set of basis operators $B_M = \{B_1, \ldots, B_m\}$ with $m \in \mathbb{N}$ is equipped with the pointwise separation property with respect to $U_{K,L}$. Let A be a class of functions satisfying the universal approximation property. If the readout function f can be chosen arbitrarily from A, then any time invariant fading memory operator $F \colon U_{K,L} \to \mathbb{R}^{\mathbb{R}}$ can be approximated arbitrarily well by M.*

Proof. By lemma 2.3.10, the operator F is uniquely determined by the continuous function

$$f_F \colon U^-_{K,L} \to \mathbb{R} \colon u \mapsto (Fu)(0) \tag{2.34}$$

Furthermore, all basis operators are time invariant and equipped with a fading memory, hence they can also be uniquely determined by continuous functions

$$b_i \colon U^-_{K,L} \to \mathbb{R} \colon u \mapsto (B_i u)(0) \tag{2.35}$$

with $i \in \{1, \ldots, m\}$.

Let $\epsilon > 0$. By lemma 2.3.8, $U^-_{K,L}$ is compact and by definition, $\{b_1, \cdots, b_m\}$ has the pointwise separation property with respect to $U^-_{K,L}$. Hence, by the Stone-Weierstrass theorem, there is a polynomial $P \in \mathbb{R}[X_1, \ldots, X_m]$ such that:

$$|f_F(u) - P(b_1(u), \ldots, b_m(u))| < \frac{\epsilon}{2} \tag{2.36}$$

for all $u \in U^-_{K,L}$.

The image of a continuous function defined on a compact set is again compact. Hence, $b_i(U^-_{K,L})$ is compact for all $i \in \{1, \ldots, m\}$ and the polynomial $P\colon \mathbb{R}^m \to \mathbb{R}$ can be restricted to a compact domain $S \subset \mathbb{R}^m$. Thus, by the universal approximation property of A, we can choose $f \in A$ such that:

$$|f(b_1(u), \ldots, b_m(u)) - P(b_1(u), \ldots, b_m(u))| < \frac{\epsilon}{2} \tag{2.37}$$

for all $u \in U^-_{K,L}$.

By combining (2.36) and (2.37), we arrive at:

$$|f_F(u) - f(b_1(u), \ldots, b_m(u))| < \epsilon \tag{2.38}$$

for all $u \in U^-_{K,L}$. □

Chapter 3

Liquid State Machines and Delay Differential Equations

We have seen in chapter 2 that the liquid state machine framework can in theory be used to perform a wide variety of computational tasks. Furthermore, our considerations suggest that liquid state machines are very well suited for tasks requiring real-time computations and a high level of adaptivity.

While I have given examples of how liquid state machines could occur naturally as networks of spiking neurons or even lakes (see section 2.2), we have not yet discussed how to artificially implement a liquid state machine for the sole purpose of exploiting its computational power. The first approach one would think of is of course to develop a computer simulation, as it is typically done in machine learning approaches based on models of neural networks. In fact, it was shown by Maass, Markram and Natschläger in 2002 that such a simulation performs promisingly well on a speech recognition task commonly used for benchmarking [15].[1]

In this chapter, however, I will discuss how the liquid state machine paradigm can be applied within a quite different realm of computing, namely the field of optical computing. While offering some interesting advantages, this approach also poses significant technological issues. These issues mainly stem from the difficulty of constructing a large recurrent network in which each node is realized by an optical device. We will see that this problem can be overcome by shifting the representation of single nodes from the spatial domain into the time domain. That is, we will consider the construction of a network that is fully realized by single optical device representing different nodes at different points in time. This idea was only recently published by Appeltant et al. [16] and will lead us to a mathematical description of the dynamics of a liquid state machine in terms of delay differential equations which will eventually bring forward the original motivation behind this thesis.

3.1 Optical computing and Liquid State Machines

For someone like me, who was born in the late 1980's, the notion of computation is strictly tied to digital electronic devices based on the Von-Neumann architecture. In spite of the

[1] A simulator and a learning tool for neural networks based on the liquid state machine paradigm can be found here: `http://www.lsm.tugraz.at/`

apparent success of these devices, situations exist where it can be beneficial to apply means of representing and manipulating information that fundamentally deviate from anything realizable with electrons and transistors. This is for instance the case when the sheer amount of required computations makes the use of digital electronic computers unfeasible (consider for example the simulation of fluid dynamics). One way of overcoming such deficits on a most basic level is, along with the increasingly popular idea of quantum computing, the concept of optical computing.

In optical computing, the carriers of information are light beams rather than electrons and the processing of data is hence achieved by manipulating said beams. If one considers any device manipulating light waves to be an optical computer, the class of optical computers naturally becomes large and heterogeneous. Indeed, the ideas and concepts generally subsumed under the term optical computing range from the application of lenses for image processing purposes to the construction of binary logic gates using optical resonators [17, p. 170]. To give an illustration of the variety of possible realizations of optical computers: If the distance between a transparent input plane and a convex lens equals the focal length of the lens, then by illuminating this setup with a coherent beam of light, the Fourier transform of the two dimensional image represented on the input plane can be made visible on an output plane, placed again in a focal length distance on the opposite side of the lens [17, p. 20]. In short: A convex lens can be used to compute the two dimensional Fourier transform in real-time, an ability that is only little known to the mathematical community.

Despite the evident heterogeneity of approaches, there are some features generally considered advantageous in optical computing implementations. Most significantly, light beams can transfer information from one point to another at the speed of light, while allowing for a vast number of parallel transmissions without generating effects of mutual interference. This implies that optical computing could resolve the bandwidth limitations from which conventional realizations of the parallel computing paradigm typically suffer. Furthermore, optical computing has always been about the promise of not only conveying but also processing information both in parallel and at the highest speed there is. Though this promise remains unfulfilled, as most optical computers are still depending on electronic inputs and outputs, the impressive ability of performing complex computations like the Fourier transform in real-time indicates a potential that is yet to be fully exploited. Finally, optical devices are in general superior to their electronic equivalents with respect to energy efficiency and heat generation, a property whose significance is not to be underestimated given the environmental implications the tremendous energy consumption caused by microelectronic devices already has.

When implementing a liquid state machine with optical computing devices, most of the advantages mentioned above will take effect. As a software implementation typically needs 1 ms to simulate one timestep [18], this could be especially interesting regarding speed. Still, there are two other, quite specific reasons why combining the paradigms of optical computing and liquid state machines might be beneficial. Firstly, it is a defining feature of a liquid state machine that its state space resides in the continuous time domain and that it can be implemented by applying biologically realistic neuron models in which the value of a neuron is analog rather than discrete (see section 2.1). Both these properties necessarily vanish to a certain extent when simulating a liquid state machine on a digital computer, but can be maintained in the realm of optical computing. Secondly, as we have seen in section 2.3, the computational power of a liquid state machine highly depends on the capability of its liquid to sufficiently separate different input functions. In other words: To make different inputs

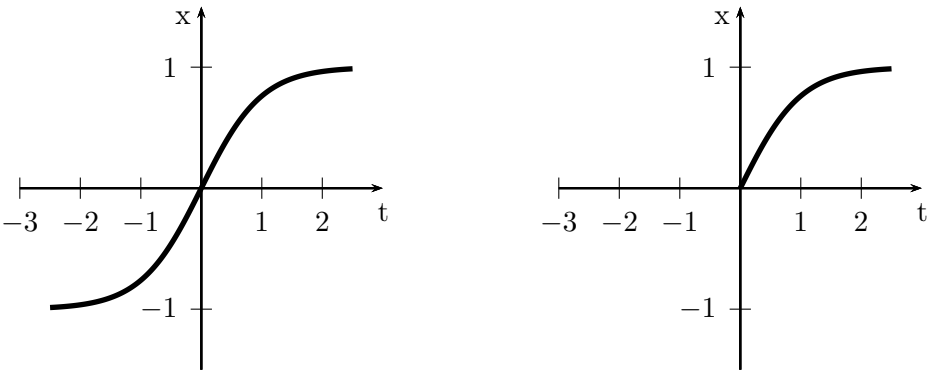

Figure 3.1: The steady-state transfer function of a semiconductor optical amplifier (right) compared to the *tanh* function (left). Note that the light level in a SOA cannot be negative, hence its transfer function is restricted to the positive domain.

well distinguishable for a linear readout function, the liquid has to map input functions into high dimensional feature spaces. It was suggested both in theory and by experiments that the rich internal dynamics of nonlinear optical elements strengthen this ability [18].

3.1.1 Circuits of Semiconductor Optical Amplifiers

In 2008, Kristof Vandoorne et al. proposed a concrete photonics-based realization of the reservoir of a liquid state machine and published first numerical results, obtained from a computer simulation [18]. Their general approach was to construct a (possibly recurrent) circuit of nodes in which each node consists of a certain optical device. Hence, the main technological issue was to find an optical device and a realizable network topology such that the dynamics of the resulting circuit were rich enough to provide good separation properties with respect to input functions.

In most software implementations of liquid state machines, the reservoir is formed by a recurrent network of neurons where the behavior of each neuron is governed by a hyperbolic tangent (*tanh*) transfer function. Inspired by the promising experimental results of such implementations, Vandoorne et al. decided to use semiconductor optical amplifiers (SOA) as the building blocks of their network, because the steady-state power transfer curve of such an amplifier is similar to the graph of a *tanh* function restricted to the positive domain (see figure 3.1). Furthermore, semiconductor optical amplifiers are well suited for the construction of photonic integrated circuits, the optical counterparts of conventional electronic microchips.

As the experiments carried out by Vandoorne et al. in 2008 weren't based on real-world implementations but on computer simulations, the topology of the circuit formed by interconnected SOAs could in theory have been chosen arbitrarily. Practically, however, one has to keep in mind that photonic chips are typically implemented on a single planar surface. Hence, a topology should avoid crossings wherever possible if to be technically realizable by optical devices.

In their first experiments, Vandoorne et al. restricted themselves to simple feed-forward structures with only a few additional feedback connections. They assumed that the nonlinear dynamics of the semiconducter optical amplifiers would suffice to compensate for the lack

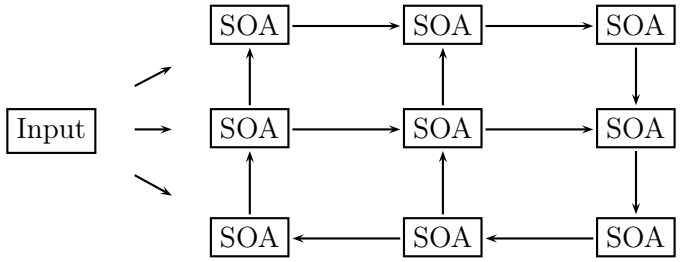

Figure 3.2: The so-called swirl topology was used by Vandoorne et al. in 2011 in simulations of photonics-based liquid state machines [19]. It is defined for a rectangular grid of nodes by the following rule: All nodes are connected to their nearest neighbors with all vertical connections on the center or left to the center of the grid directed upwards and all horizontal connections above or on the center of the grid oriented to the right.

of complexity regarding the network topology. A claim that was in fact supported by their experimental findings. In a later publication [19], a topology was proposed which exhibits a significantly higher degree of recurrence while maintaining the restriction that interconnections are only allowed between neighboring nodes. This so-called swirl topology is depicted in figure 3.2.

In the numerical simulations carried out by Vandoorne et al., the performance of a photonics-based liquid state machine was in general comparable to that of a traditional software implementation regarding benchmarking tasks such as pattern recognition [18] and speech recognition [19]. This is surprising insofar as that the reservoir of a simulated liquid state machine typically consists of thousands of nonlinear nodes while the network used by Vandoorne et al. in [19] contains no more than 100 SOAs. But even with this reduced number of nodes and significant restrictions regarding the topology of the reservoir, the technological difficulties underlying the construction of the proposed integrated circuit seem to be hard to come by: The promising results given by computer simulations are yet to be reproduced by a real implementation of a corresponding photonics-based circuit.

One way of circumventing such difficulties would be to categorically refrain from the use of interconnected nodes. Though it seems unlikely at first glance that a single node could provide rich enough dynamics to serve as an applicable reservoir, this approach is already pursued with some preliminary success by a large international research project called PHOCUS[2]. The pivotal idea behind this venture will be presented and explained in the following section.

3.2 Realizing a recurrent Liquid with a single Node

In this section, I will explain how the dynamics of a single optical node can be utilized to implement the reservoir of a liquid state machine. The concept described here was first published by Appeltant et al. in 2011 [16] and exploits the intriguing dynamics of a certain class of dynamical systems called ‘delay systems’.

[2]PHOCUS stands for: towards a PHOtonic liquid state machine based on delay-CoUpled Systems. See `http://ifisc.uib-csic.es/phocus/`.

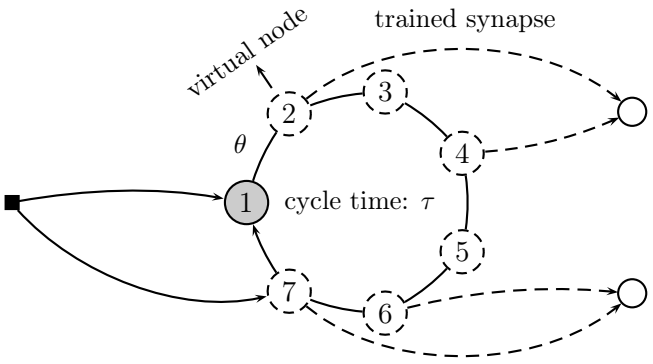

Figure 3.3: This is a sketch of a single node liquid state machine as proposed by Appeltant et al. [16]. One single optical device (node 1) that is subject to delayed feedback with delay time τ realizes a total of 7 virtual nodes. This is achieved by identifying each virtual node with intervals of time on which the optical device realizes the corresponding virtual node. For the 5th node, these intervals are for instance given by $[k\tau + 4\theta, (k+1)\tau + 5\theta) \subset \mathbb{R}$ with $k \in \mathbb{N}$ and $7 = \tau/\theta$. Note that this construction causes the output of a liquid state machine to be discretized, that is an input function of time is mapped to a sequence of time steps. This also demands that an input function $u\colon \mathbb{R} \to \mathbb{R}$ is discretized by an operator J before fed to the optical device (see definition 3.2.2).

The class of delay systems consists of all dynamical systems with delayed feedback. Note that systems with regular, that is undelayed, feedback can be considered delay systems with delay time 0. In 1987, Ikeda et al. mathematically modeled the dynamics of a nonlinear optical resonator with delayed feedback and studied the bifurcation behavior of the resulting equations [20]. Amongst other things, they showed that the increase of a certain control parameter caused the dynamics of their model to exhibit high dimensional chaos. This theoretical observation was confirmed experimentally in 1994 by Fischer et al. [21], who realized high dimensional chaos in a semiconductor laser whose output was fed back by an external mirror. These results suggest that the dynamics of a single optical device equipped with delayed feedback might be rich enough to realize a computationally applicable reservoir. In 2011, Appeltant et al. proposed a concrete implementation of such a single node liquid and published promising experimental results [16]. Their approach will be presented in the upcoming paragraphs.

As I have already mentioned, the basic idea of Appeltant et al. was to use one optical device to represent different nodes of a possibly large recurrent circuit at different points in time. That is, the realization of the topology of the circuit is shifted from the spatial domain to the time domain.

Let's consider a single nonlinear device D that is subject to delayed feedback with delay time $\tau > 0$. Let $n \in \mathbb{N}$ denote the number of nodes to be realized by D and $\theta = \tau/N$, then the values of the i-th node are given by the values of D at times $t_i = k\tau + i\theta$ with $k \in \mathbb{Z}$. Note that this construction results in a discretized state space: The state of the network realized by D cannot be determined at single points in time but only at intervals of time

given by $[k\tau, (k+1)\tau) \subset \mathbb{R}$ with $k \in \mathbb{Z}$. Considering that in this approach, the topology of a whole network is fully contained in the time domain, this loss of continuity doesn't come as a surprise. Still, it begs the question of how continuous inputs can be fed to an essentially time-discrete system.

The answer given to this issue by Appeltant et al. is again discretization. An input function $u \in \mathbb{R}^{\mathbb{R}}$ is sampled at each integer multiple of τ and thereby mapped to a piecewise constant function $u_{\mathrm{pc}} \colon \mathbb{R} \to \mathbb{R}$ with $u_{\mathrm{pc}}(k\tau) = u(k\tau)$ and $u_{\mathrm{pc}}(t) = u_{\mathrm{pc}}(k\tau)$ for all $t \in [k\tau, (k+1)\tau) \subset \mathbb{R}$ with $k \in \mathbb{Z}$. The weight of a synapse connecting the input layer with the i-th node of the liquid can than be realized by simply multiplying u_{pc} with the corresponding weight on intervals $[k\tau + (i-1)\theta, (k+1)\tau + i\theta) \subset \mathbb{R}$ and feeding the resulting value to the optical device. This procedure can formally be described by the discretization operator $J \colon \mathbb{R}^{\mathbb{R}} \to \mathbb{R}^{\mathbb{R}}$ (see definition 3.2.2 and figure 3.4). Also note that dependencies between nodes in the reservoir can be realized by choosing $\theta < T$, where T is the intrinsic time scale of the nonlinear node D (see supplementary discussion in [16]). This leads to the following formal definition of a single node liquid state machine (SNLSM).

Definition 3.2.1 (Single node liquid state machine)**.** *Let $D_\tau \colon \mathbb{R}^{\mathbb{R}} \to \mathbb{R}^{\mathbb{R}}$ be a single optical device that is subject to delayed feedback with delay time $\tau > 0$. Let $n \in \mathbb{N}$ be the number of nodes to be realized by D_τ and $\theta > 0$ such that $n = \tau/\theta$ and $f \colon \mathbb{R}^n \to \mathbb{R}^l$ be a readout map with $l \in \mathbb{N}$. Then*

$$M := \langle D_\tau, f \rangle_n \tag{3.1}$$

defines a single node liquid state machine (SNLSM), mapping from $\mathbb{R}^{\mathbb{R}}$ to $(\mathbb{R}^{\mathbb{Z}})^l$.

Let $u \in \mathbb{R}^{\mathbb{R}}$ be an arbitrary input function of time, then the liquid state $x_M(t) \in \mathbb{R}^n$ and the value $y_M(t) \in \mathbb{R}^l$ of M at time step $t \in \mathbb{Z}$ are given by:

$$\begin{aligned} x_M(t) &= \begin{pmatrix} (D_\tau(Ju))(t\tau) \\ (D_\tau(Ju))(t\tau + \theta) \\ \vdots \\ (D_\tau(Ju))(t\tau + (n-1)\theta) \end{pmatrix} \\ y_M(t) &= f(x_M(t)) \end{aligned}$$

where $J \colon \mathbb{R}^{\mathbb{R}} \to \mathbb{R}^{\mathbb{R}}$ is a discretization operator (see definition 3.2.2).

We now define the discretization parameter J (see also figure 3.4:

Definition 3.2.2. *Let $\tau, \theta > 0$ and $n \in \mathbb{N}$ such that $n = \tau/\theta$. Let $(w_i)_{i=1}^n \subset \mathbb{R}$ be a sequence of weights. Then the discretization operator $J \colon \mathbb{R}^{\mathbb{R}} \to \mathbb{R}^{\mathbb{R}}$ is given by:*

$$(Ju)(t) = w_i u(k\tau) \tag{3.2}$$

for $t \in [k\tau + (i-1)\theta, k\tau + i\theta)$ with $k \in \mathbb{Z}, i \leq n$ and $u \in \mathbb{R}^{\mathbb{R}}$.

To test the general applicability of their approach, Appeltant et al. conducted several numerical and physical experiments in which a SNLSM realized by an electronic implementation of a Mackey-Glass oscillator [22] was used to perform a spoken digit recognition task [23] and the NARMA system modelling task, proposed by Herbert Jaeger [24]. Note that it was not the aim of these experiments to confirm the assumed computational superiority of optical devices. Hence, using an electronic device instead of an optical node was perfectly valid.

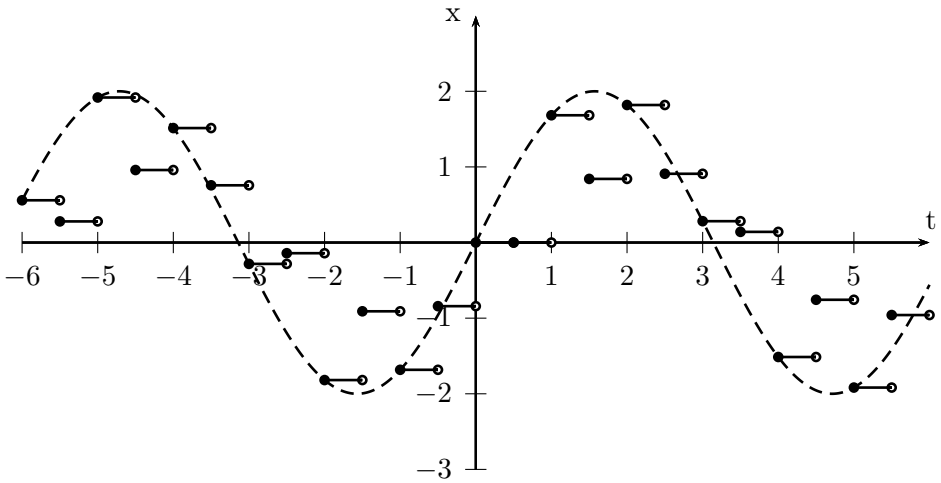

Figure 3.4: The discretization operator J (see definition 3.2.2) samples input functions at equidistant points $k\tau \in \mathbb{R}$ and applies weights on intervals $[j\theta, (j+1)\theta)$ with $j, k \in \mathbb{Z}$. This plot shows what happens when J is applied to a continuous function $f\colon \mathbb{R} \to \mathbb{R}\colon t \mapsto 2\sin(t)$ with $\tau = 1$, $\theta = 0.5$ and weights $w_1 = 1$, $w_2 = 0.5$.

In both tasks, the performance of the implemented SNLSM was comparable to and sometimes even better than that of conventional realizations of liquid state machines, given that certain parameters governing the behavior of the single dynamical node were chosen appropriately. However, it was also apparent that even slight modifications of these parameters would result in significant performance changes. This confirms the intuition that a computationally feasible liquid state machine can be constructed with a single dynamical node. It also shows that the performance of such a realization is highly dependent on the specific dynamics exhibited by this node, which doesn't come as a surprise as we have already demonstrated the strong connection between the configuration of the liquid and the computational power of the overall machine in section 2.3. Needless to say, it would be interesting if predictions about the computational behavior regarding a specific task can be made by solely analyzing the mathematical model of a single dynamical node rather than performing huge numbers of experiments. In a first step towards this goal, let us now consider the mathematical description of a Mackey-Glass system, which is given by a nonlinear differential equation. Note that in contrast to the original Mackey-Glass equation [22], the following definition describes a non-homogeneous system. That is, the system also depends on input provided by a function $u \in \mathbb{R}^{\mathbb{R}}$.

Definition 3.2.3 (Mackey-Glass dynamical node)**.** *Let $\eta, \gamma, p > 0$, then the dynamics of a Mackey-Glass dynamical node $D_\tau\colon \mathbb{R}^{\mathbb{R}} \to \mathbb{R}^{\mathbb{R}}$ with delay time $\tau > 0$ are given by:*

$$\frac{dx(t)}{dt} = -x(t) + \eta \frac{x(t-\tau) + \gamma(Ju)(t)}{1 + (x(t-\tau) + \gamma(Ju)(t))^p} \tag{3.3}$$

where $u\colon \mathbb{R} \to \mathbb{R}$ is an arbitrary input function, $x = D_\tau u$, $t \in \mathbb{R}$ and $J\colon \mathbb{R}^{\mathbb{R}} \to \mathbb{R}^{\mathbb{R}}$ denotes the discretization operator (see definition 3.2.2).

Note that the parameters η and γ in (3.3) can be used to control the feedback strength and the influence of an input function $u \in \mathbb{R}^{\mathbb{R}}$ while p describes the degree of nonlinearity. Furthermore, the parameter η can be chosen such that the dynamical system defined by (3.3) operates in a stable fixed point in the absence of input.

A basic understanding the influence parameters η, τ and p have on a dynamical system de-

scribed by (3.3) would be to find a general solution of (3.3). Though this seems promising at first sight, we will see in the following section that the specific structure of delay differential equations will prevent us from gaining insight by simply trying to solve them.

3.3 Delay Differential Equations and the Method of Steps

The characteristic property of a delay differential equation is that the derivatives of a solution x at time t may not only depend on the value $x(t)$ but also on values $x(t-\tau)$ with $\tau > 0$. Such delay-relationships exist in many biological and physical systems. A simple example would be a tank containing salt water that is being refilled with pure water while leaking from the bottom. Typically, such a system is modeled by an ordinary differential equation under the assumption that new water is instantaneously distributed uniformly across the tank. By using delay differential equations, this somewhat unrealistic assumption can be dropped. A biological example is the reproduction of blood cells which can be modeled by the same type of equations we already used in definition 3.2.3 (see also [25] and [22]).

The relationship between a function and its derivatives given by a differential equations is usually satisfied by a huge class of different solutions. The equation describing these solutions up to certain parameters is then called the general solution of the according differential equation. When working with ordinary differential equations, we can specify which solutions are of interest by additionally requiring solutions to have a specific value at an initial point in time. Such a combination of a differential equation and an initial condition is usually referred to as the initial value problem or the Cauchy problem.

In the case of delay differential equations, however, the derivatives of a solution x on an interval $[k\tau, (k+1)\tau]$ can depend on the values of the function x on the whole interval $[(k-1)\tau, k\tau]$ with $k \in \mathbb{Z}$ and delay $\tau > 0$. This implies that an initial value $x(t_0)$ at time t_0 would not suffice as an initial condition. Instead, an initial function defined on the interval $[t_0 - \tau, t_0]$ has to be provided.[3] Furthermore, above observation leads to the fundamental (and to my knowledge: only) method of analytically solving arbitrary delay differential equations, the so-called method of steps.

Consider a Cauchy problem given by a delay differential equation with delay $\tau > 0$ and an initial function $f\colon [t_0 - \tau, t_0] \to \mathbb{R}^n$ with initial time $t_0 \in \mathbb{R}$ and $n \in \mathbb{N}$. If we restrict ourselves to the time interval $[t_0, t_0 + \tau] \subset \mathbb{R}$, all occurrences of $x(t-\tau)$ in the delay differential equation can simply be replaced by $f(t-\tau)$. This procedure reduces the delay differential equation on the interval $[t_0, t_0 + \tau] \subset \mathbb{R}$ to an ordinary one that can be solved by conventional methods like the separation of variables or the variation of constants. Having obtained a solution on the interval $[t_0, t_0 + \tau] \subset \mathbb{R}$, we can use it in the same fashion as we have used the initial function f to reduce the given delay differential equation to an ordinary differential equation on the interval $[t_0 + \tau, t_0 + 2\tau] \subset \mathbb{R}$. By applying this method recursively, we can find a solution for an initial value problem for every $t \in [t_0, \infty)$.

In the final part of this chapter, I will demonstrate the method of steps by solving the initial value problem for a linear Mackey-Glass system on the interval $[0, 2\tau] \subset \mathbb{R}$. Note that

[3] Note that it is not required for the initial function f to satisfy the corresponding delay differential equation. Also note that the partition into subintervals of length τ induced by the initial value problem of a delay differential equation resembles the construction used in section 3.2 to contain the topology of recurrent circuits in a single node. This is, of course, no coincidence.

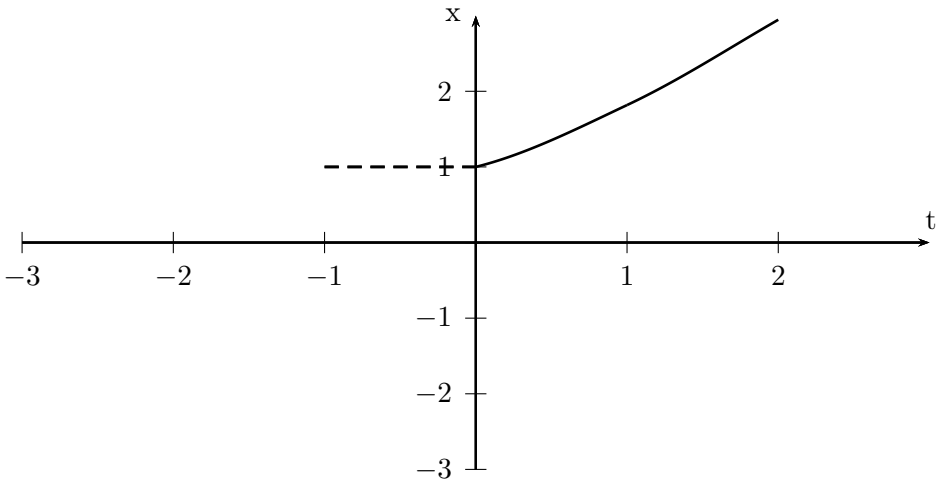

Figure 3.5: This figure shows the graph of the solution of the initial value problem given in example 3.3.1 on the interval $[0, 2]$. The dashed line corresponds to the initial function $f\colon [-1, 0] \to \mathbb{R}\colon x \mapsto 1$.

the results given in chapter 4 only concern linear systems, hence, the restriction to linear systems will be maintained throughout the remaining parts of this thesis. Furthermore, the application of the discretization operator J (see definition 3.2.2) will from now on be omitted while input functions will generally assumed to be continuous. This assumption also eases the mathematical analysis while having little effect on the validity of the following results regarding single node liquid state machines, as the operator J could easily be replaced by an operator mapping into the space of continuous functions in definition 3.2.1.

A comprehensive introduction to ordinary and delay differential equations can be found in [26].

Example 3.3.1. *Let $\tau = 1$, $\gamma = 1$, $\eta = 3$, $p = 0$, and an input function given by $u\colon \mathbb{R} \to \mathbb{R}\colon x \mapsto \sin(x)$. Furthermore, let $f\colon [-1, 0] \to \mathbb{R}\colon x \mapsto 1$ be an initial function, then the initial value problem of the according non-homogeneous Mackey-Glass system is defined as follows:*

$$\frac{dx(t)}{dt} = -x(t) + \frac{3}{2}(x(t-1) + \sin(t)) \qquad \forall t \in [0, \infty) \tag{3.4}$$

$$x(t) = 1 \qquad \forall t \in [-1, 0] \tag{3.5}$$

The solution of the Cauchy problem (3.4), (3.4) *on the interval* $[0, 2]$ *can be obtained by the method of steps and is given by:*

$$x(t) = \frac{1}{4}e^{-t} + \frac{3}{4}\left(\sin(t) - \cos(t) + 2\right) \qquad \forall t \in [0, 1] \tag{3.6}$$

$$x(t) = \frac{1}{4}e^{-t} + \frac{3}{8}\left(e^{1-t}t + 2\sin(t) - 3\cos(1-t) - 2\cos(t) + 6\right) \qquad \forall t \in [1, 2] \tag{3.7}$$

Proof. Note that for $t \in [0, 1]$, the initial value problem (3.4), (3.5) reduces to:

$$\frac{dx(t)}{dt} = -x(t) + \frac{3}{2}\left(1 + \sin(t)\right) \tag{3.8}$$

$$x(0) = 1 \tag{3.9}$$

where (3.8) is a linear non-homogeneous ordinary differential equation of first order. The general solution of (3.8) is given by:

$$x(t) = Ce^{-t} + \frac{3}{4}\left(\sin(t) - \cos(t) + 2\right) \tag{3.10}$$

with $C \in \mathbb{R}$. Hence, we get a first solution for the system (3.4), (3.5):

$$x(t) = \frac{1}{4}e^{-t} + \frac{3}{4}\left(\sin(t) - \cos(t) + 2\right) \qquad \forall t \in [0,1] \tag{3.11}$$

Note that (3.11) implies for $t \in [1,2]$:

$$\frac{dx(t)}{dt} = -x(t) + \frac{3}{2}\left(x(t-1) + \sin(t)\right) \tag{3.12}$$

$$= -x(t) + \frac{3}{2}\left(\frac{1}{4}e^{-t+1} + \frac{3}{4}\left(\sin(t-1) - \cos(t-1) + 2\right) + \sin(t)\right) \tag{3.13}$$

Hence, the original initial value problem can for $t \in [1,2]$ be reduced to:

$$\frac{dx(t)}{dt} = -x(t) + \frac{3}{2}\left(\frac{1}{4}e^{-t+1} + \frac{3}{4}\left(\sin(t-1) - \cos(t-1) + 2\right) + \sin(t)\right) \tag{3.14}$$

$$x(1) = \frac{1}{4e} + \frac{3}{4}\left(\sin(1) - \cos(1) + 2\right) \tag{3.15}$$

Again, we compute the general solution of (3.14):

$$x(t) = Ce^{-t} + \frac{3}{8}\left(e^{1-t}t + 2\sin(t) - 3\cos(1-t) - 2\cos(t) + 6\right) \tag{3.16}$$

with $C \in \mathbb{R}$. This leads to a solution of the Cauchy problem (3.4), (3.5) for $t \in [1,2]$:

$$x(t) = \frac{1}{4}e^{-t} + \frac{3}{8}\left(e^{1-t}t + 2\sin(t) - 3\cos(1-t) - 2\cos(t) + 6\right) \tag{3.17}$$

□

The above example illustrates that, while providing a decent algorithm for analytically solving initial value problems of delay differential equations, the method of steps is a cumbersome tool if we aim to derive statements about long-term dynamical behavior: Firstly, one cannot derive a solution on a time interval $[k\tau, (k+1)\tau] \subset \mathbb{R}$ with $k \in \mathbb{N}$ without computing solutions for all prior intervals. Secondly, the analytical results given by the method of steps are quickly reaching an uncomfortably high degree of complexity. We will see in the following chapter that at least for linear systems, this problems might be overcome by rewriting an initial value problem as a Volterra series expansion.

Chapter 4

Towards a Volterra Series Approach to Liquid State Machines

In the previous chapter, I have discussed how a liquid state machine can be realized with a single dynamical node whose dynamics are modeled by a certain delay differential equation. Differential equations provide a precise but only implicit description of the dynamics of a system in the presence of input. As we have seen in example 3.3.1, solving such an equation and thereby computing the actual system output can be a tedious task.

A more explicit point of view was adopted if we could express the output of a system directly as a function of its input. In this chapter, I will show that such an explicit description can in in fact be derived in some cases by rewriting a given ordinary differential equation as a sum of integral convolutions, usually called a Volterra series[1]. Subsequently, I will show in the final part of this thesis that by applying the Lambert W function, this is also possible for linear delay differential equations.

4.1 Obtaining Volterra-like Series Representations from linear Ordinary Differential Equations

The French mathematician Maurice Fréchet (1878-1973) showed in 1910 that any continuous functional can be approximated arbitrarily well on every compact set of points by a possibly infinite sum of integral convolutions [29]. Note that this result can be considered a generalization of the Stone-Weierstrass theorem we have already used in section 2.3 (see lemma 2.3.7). Furthermore, this 'Fréchet approximation theorem' indicates that for certain dynamical system, an explicit input-output relation can also be given by a series of integral convolutions. It was in fact proven rigorously by Boyd et al. in 1985 [14] that any nonlinear time invariant continuous operator can be approximated by an expansion of the following form[2]:

Definition 4.1.1 (Volterra series expansion)**.** *Let $u(t) \in \mathbb{R}^n$ denote the input and $x(t) \in \mathbb{R}^n$ denote the output of a possibly nonlinear time invariant system S at time $t \in \mathbb{R}$ with $n \in \mathbb{N}$.*

[1] For a detailed introduction to the Volterra series approach to dynamical systems see [27] and [28]

[2] Note that this work of Boyd et al. was already referenced in remark 2.3.9 and served as a template for the proof of theorem 2.3.11.

Let $(k_i)_{i\in\mathbb{N}}$ be a family of kernel functions such that:

$$x(t) = k_0 + \int_{-\infty}^{\infty} k_1(\tau_1)u(t-\tau_1)d\tau_1 + \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} k_2(\tau_1,\tau_2)u(t-\tau_1)u(t-\tau_2)d\tau_1 d\tau_2 + \cdots \tag{4.1}$$

$$= k_0 + \sum_{i=1}^{\infty} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} k_i(\tau_1,\tau_2,\ldots,\tau_n)u(t-\tau_1)\cdots u(t-\tau_i)d\tau_1\ldots d\tau_i \tag{4.2}$$

then (4.2) *is called the Volterra series expansion of S.*

Our aim will now be to find the Volterra series expansion of a dynamical system realized by the dynamical node described in definition 3.2.3. That is, we will try to derive an explicit input-output representation from a non-homogeneous Mackey-Glass equation. As I have already noted, this task will be restricted to the linear case.

While we can be confident that such a representation does in fact exist, there is unfortunately no straightforward procedure for finding it. As a starting point, let us consider how a Volterra series input-output representation can be obtained from linear non-homogeneous ordinary differential equations. The following method is taken from [28, p. 94] and will provide us with a fruitful intuition on how Volterra series representations could also be obtained from delay differential equations.

Theorem 4.1.2. *Let a dynamical system be implicitly given by the following initial value problem:*

$$\frac{dx(t)}{dt} = A(t)x(t) + u(t) \qquad \forall t \in [0,\infty) \tag{4.3}$$

$$x(0) = x_0 \tag{4.4}$$

where $x(t) \in \mathbb{R}^n$ with $n \in \mathbb{N}$ is the state vector at time $t \geq 0$ and $u\colon \mathbb{R} \to \mathbb{R}^n$, $A\colon \mathbb{R} \to \mathbb{R}^{n\times n}$ are both continuous functions. Furthermore, we assume that the system (4.3), (4.4) *has a unique solution on a interval $[0,T] \subset \mathbb{R}$ with $T > 0$. Then for $t \in [0,T]$,*

$$x(t) = \Phi(t,0)x_0 + \int_0^t \Phi(t,\tau)u(\tau)d\tau \tag{4.5}$$

is a solution of (4.3), (4.4) *where a transition matrix $\Phi(t,\tau) \in \mathbb{R}^{n\times n}$ with $t,\tau \in [0,T]$ is defined by:*

$$\Phi(t,\tau) = I + \sum_{i=1}^{\infty} \int_\tau^t A(s_1) \int_\tau^{s_1} A(s_2)\cdots \int_\tau^{s_{i-1}} A(s_i)ds_i\ldots ds_1 \tag{4.6}$$

where $I \in \mathbb{R}^{n\times n}$ is the identity matrix.

Proof. Let's first consider the homogeneous case. That is, we assume $u(t) = 0$ for all $t \geq 0$.

Let $t \in [0,T]$. Note that by simple integration, we get from (4.3) and (4.4):

$$x(t) = x_0 + \int_0^t A(s)x(s)ds \tag{4.7}$$

Now (4.7) can be substituted in the right hand side of (4.7), leading to:

$$x(t) = x_0 + \int_0^t A(s_1)x(s_1)ds_1 \tag{4.8}$$

$$= x_0 + \int_0^t A(s_1)\left(x_0 + \int_0^{s_1} A(s_2)x(s_2)ds_2\right)ds_1 \tag{4.9}$$

$$= \left(I + \int_0^t A(s_1)ds_1\right)x_0 + \int_0^t A(s_1)\int_0^{s_1} A(s_2)x(s_2)ds_2ds_1 \tag{4.10}$$

$$= \left(I + \int_0^t A(s_1)ds_1\right)x_0 + \int_0^t A(s_1)\int_0^{s_1} A(s_2)\left(x_0 + \int_0^{s_2} A(s_3)x(s_3)ds_3\right)ds_2ds_1 \tag{4.11}$$

By repeating this procedure indefinitely, we arrive at:

$$x(t) = \lim_{k\to\infty}\left(\begin{array}{l}\left(I + \sum_{i=1}^{k}\int_0^t A(s_1)\int_0^{s_1} A(s_2)\cdots\int_0^{s_{i-1}} A(s_i)ds_i\ldots ds_1\right)x_0 \\ + \int_0^t A(s_1)\int_0^{s_1} A(s_2)\cdots\int_0^{s_k} A(s_{k+1})x(s_{k+1})ds_{k+1}\cdots ds_2ds_1\end{array}\right) \tag{4.12}$$

It can be shown that the last summand in (4.12) approaches 0 while

$$\Phi(t,\tau) = \lim_{k\to\infty}\left(I + \sum_{i=1}^{k}\int_\tau^t A(s_1)\int_\tau^{s_1} A(s_2)\cdots\int_\tau^{s_{i-1}} A(s_i)ds_i\ldots ds_1\right) \tag{4.13}$$

converges uniformly for every $t,\tau \in [0,T]$ (see for example [30]).

Hence,

$$x(t) = \lim_{k\to\infty}\left(I + \sum_{i=1}^{k}\int_0^t A(s_1)\int_0^{s_1} A(s_2)\cdots\int_0^{s_{i-1}} A(s_i)ds_i\ldots ds_1\right)x_0 \tag{4.14}$$

$$= \Phi(t,0)x_0 \tag{4.15}$$

is the homogeneous solution of (4.3).

The infinite series given in (4.14) is typically referred to as the Peano-Baker series[3] while $\Phi(t,\tau)$ is called a state transition matrix, as it can be used to compute the transition between two states $x(\tau)$ and $x(t)$. The matrices $\Phi(t,\tau) \in \mathbb{R}^{n\times n}$ with $t,\tau \in [0,T]$ have two interesting properties which can be exploited to obtain the desired input-output representation. Firstly, a transition matrix can always be written as a product by using an intermediary state, that is:

$$\Phi(t,\tau) = \Phi(t,\sigma)\Phi(\sigma,\tau) \tag{4.16}$$

for all $t,\tau,\sigma \in [0,T]$. Furthermore, all transition matrices are invertible and it holds that:

$$\Phi^{-1}(t,\tau) = \Phi(\tau,t) \tag{4.17}$$

with $t,\tau \in [0,T]$.

[3]Note that the Peano-Baker series is mainly a tool for proofing the existence and uniqueness of solutions rather than actually computing them.

Let us now turn to the original non-homogeneous initial value problem. First note that:

$$\frac{d\Phi(t,0)}{dt} = \frac{d\left(I + \sum_{i=1}^{\infty} \int_0^t A(s_1) \int_0^{s_1} A(s_2) \cdots \int_0^{s_{i-1}} A(s_i) ds_i \dots ds_1\right)}{dt} \tag{4.18}$$

$$= \sum_{i=1}^{\infty} \frac{d \int_0^t A(s_1) \int_0^{s_1} A(s_2) \cdots \int_0^{s_{i-1}} A(s_i) ds_i \dots ds_1}{dt} \tag{4.19}$$

$$= \sum_{i=1}^{\infty} A(t) \int_0^t A(s_1) \int_0^{s_1} A(s_2) \cdots \int_0^{s_{i-1}} A(s_i) ds_i \dots ds_1 \tag{4.20}$$

$$= A(t) \left(I + \sum_{i=1}^{\infty} \int_0^t A(s_1) \int_0^{s_1} A(s_2) \cdots \int_0^{s_{i-1}} A(s_i) ds_i \dots ds_1\right) \tag{4.21}$$

$$= A(t)\Phi(t,0) \tag{4.22}$$

Hence, substituting $z(t) = \Phi^{-1}(t,0)x(t)$ in equation (4.3) gives:

$$\frac{d\left(\Phi(t,0)z(t)\right)}{dt} = A(t)\Phi(t,0)z(t) + u(t) \quad \Leftrightarrow \tag{4.23}$$

$$\Phi(t,0)\frac{dz(t)}{dt} + A(t)\Phi(t,0)z(t) = A(t)\Phi(t,0)z(t) + u(t) \quad \Leftrightarrow \tag{4.24}$$

$$\frac{dz(t)}{dt} = \Phi^{-1}(t,0)u(t) \tag{4.25}$$

with $t \in [0,T]$. This leads to the following reformulation of the initial value problem (4.3), (4.4):

$$\frac{dz(t)}{dt} = \Phi^{-1}(t,0)u(t) \qquad \forall t \in [0,\infty) \tag{4.26}$$

$$z(0) = x_0 \tag{4.27}$$

Again, we simply integrate and get:

$$z(t) = x_0 + \int_0^t \Phi^{-1}(\tau,0)u(\tau)d\tau \quad \Leftrightarrow \tag{4.28}$$

$$\Phi^{-1}(t,0)x(t) = x_0 + \int_0^t \Phi^{-1}(\tau,0)u(\tau)d\tau \quad \Leftrightarrow \tag{4.29}$$

$$x(t) = \Phi(t,0)x_0 + \Phi(t,0)\int_0^t \Phi(0,\tau)u(\tau)d\tau \quad \Leftrightarrow \tag{4.30}$$

$$x(t) = \Phi(t,0)x_0 + \int_0^t \Phi(t,\tau)u(\tau)d\tau \tag{4.31}$$

□

Note that the first summand in (4.7) depends on t, hence the given input-output representation is not a Volterra series expansion in the strict sense of definition 4.1.1. However, theorem 4.1.2 establishes the desired connection between the implicit description of a dynamical system given by linear differential equations and an explicit representation, where the output can directly be computed as a function of the input.

I'd also like to point out that if A wasn't a function of time but a constant matrix, the transition matrices in theorem 4.1.2 would reduce to a simple matrix exponential, that is:

$$\Phi(t,\tau) = e^{A(t-\tau)} \tag{4.32}$$

with $t, \tau \in [0,T]$. This is of course no surprise, as the initial value problem (4.3), (4.4) could then be solved by the variation of constants formula, yielding the exact same result.

Finally, it is important to realize that the Volterra-like series representation derived in theorem 4.1.2 is fully determined by the transition matrices $\Phi(\tau,t)$. This suggests that the task of obtaining Volterra-like input-output representations from delay differential equations could be strongly connected to the task of generalizing the concept of transition matrices to delay differential equations. We will see in the following section that this intuition is in fact correct.

4.2 Obtaining Volterra-like Series Representations from linear Delay Differential Equations

In 2003, Farshid Asl and A. Galip Ulsoy proposed a method of analytically solving linear delay differential equations using the Lambert W function [31]. One essential property of their approach is that it's closely related to the concept of state transition matrices we have already used in theorem 4.1.2. It was conjectured at the end of section 4.1 that such a method could also be used to obtain Volterra-like series representations from delay differential equations. We will see in the course of this section that this is indeed the case.

However, before discussing the approach brought forward by Asl et al., I'd like to briefly introduce the Lambert W function. The Lambert W function is named after the Swiss mathematician Johann Heinrich Lambert (1728 - 1777) as its origins can be traced back to his work on the so-called ´Lambert's Transcendental Equation' [32] (for a more elaborate historical perspective on the develepment of the Lambert W function, see [33]). It is defined to be the inverse of $x \mapsto xe^x$, that is:

Definition 4.2.1 (Lambert W function)**.** *Let $x \in \mathbb{C}$, then the Lambert W function is given by the functional equation:*

$$x = W(x)e^{W(x)} \tag{4.33}$$

Note that $x \to xe^x$ is not injective on $\mathbb{C}$, hence the value $W(x)$ may not be uniquely defined by equation (4.33). In fact, assuming complex values, the function W has infinitely many branches (see figure 4.1). The methods for computing these branches are given below:

Lemma 4.2.2 (Branches of the Lambert W function.)**.** *The principle branch $W_0 \colon \mathbb{C} \to \mathbb{C}$ of the Lambert W function can be computed by the following series:*

$$W_0(x) = \sum_{n=1}^{\infty} \frac{(-n)^{n-1}}{n!} x^n \tag{4.34}$$

with $x \in \mathbb{C}$.

Let $k \in \mathbb{Z}$. The k-th branch $W_k \colon \mathbb{C} \to \mathbb{C}$ of the Lambert W function can be represented by the following series:

$$W_k(x) = \ln_k(x) - \ln(\ln_k(x)) + \sum_{l=0}^{\infty} \sum_{m=1}^{\infty} C_{lm} \frac{(\ln(\ln_k(x)))^m}{(\ln_k(x))^{l+m}} \tag{4.35}$$

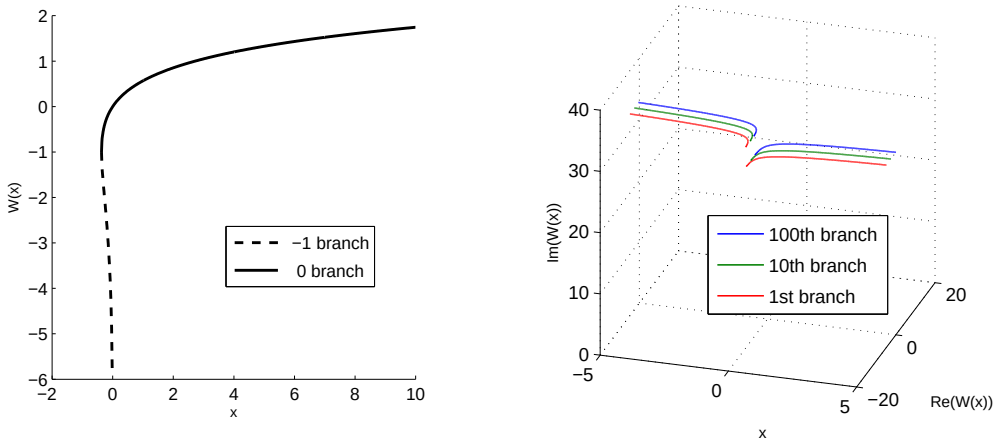

Figure 4.1: The left plot shows the graph of the principle branch $W_0(x)$ with $x \in [-e^{-1}, 10]$ and the graph of $W_{-1}(0)$ on $[-e^{-1}, 0]$ (dashed line). The right plot depicts the graphs of the first, 10th and 100th branch of the Lambert W function on $[-5, 5]$.

where $x \in \mathbb{C}$,

$$C_{lm} = \frac{1}{m!}(-1)^l \begin{bmatrix} l+m \\ l+1 \end{bmatrix} \tag{4.36}$$

and $\ln_k$ *denotes the* k*-th branch of the logarithmic function, that is:*

$$\ln_k(x) = \ln(x) + 2\pi i k. \tag{4.37}$$

Proof. For more details on the properties and derivation of (4.34) and (4.35), see [34] and [35]. □

The following theorem is taken from [31] and demonstrates how the Lambert W function can be used to analytically solve linear delay differential equations. This will eventually lead to a method of obtaining Volterra-like series representations from a linear Mackey-Glass system.

Theorem 4.2.3. *Let a homogeneous and linear initial value problem be given by:*

$$\frac{dx(t)}{dt} = \alpha x(t) + \beta x(t-\tau) \qquad \forall t \in [0, \infty) \tag{4.38}$$

$$x(t) = f(t) \qquad \forall t \in [-\tau, 0] \tag{4.39}$$

where $\alpha, \beta, \tau \in \mathbb{R}$ *and* $f\colon [-\tau, 0] \to \mathbb{R}$ *is a continuous initial function.*

The solution of the system (4.38), (4.39) *is then given by:*

$$x(t) = \sum_{k=-\infty}^{\infty} C_k e^{\left(\frac{1}{\tau} W_k(\beta\tau e^{-\alpha\tau}) + \alpha\right)t} \tag{4.40}$$

with $t \geq 0$*, where* W_k *denotes the* k*-th branch of the Lambert W function and coefficients* $C_k \in \mathbb{C}$ *are chosen such that:*

$$f(t) = \sum_{k=-\infty}^{\infty} C_k e^{\left(\frac{1}{\tau} W_k(-\beta\tau e^{\alpha\tau}) - \alpha\right)t} \tag{4.41}$$

for all $t \in [-\tau, 0]$.

Proof. Substituting the ansatz

$$x(t) = Ce^{st} \tag{4.42}$$

with $C \neq 0$ in (4.38) yields:

$$\begin{aligned}
\frac{dCe^{st}}{dt} &= \alpha Ce^{st} + \beta Ce^{s(t-\tau)} && \Leftrightarrow && (4.43)\\
Cse^{st} &= \alpha Ce^{st} + \beta Ce^{s(t-\tau)} && \Leftrightarrow && (4.44)\\
\beta &= se^{st}e^{-s(t-\tau)} - \alpha e^{st}e^{-s(t-\tau)} && \Leftrightarrow && (4.45)\\
\beta &= (s-\alpha)e^{s\tau} && \Leftrightarrow && (4.46)\\
\tau e^{-\alpha\tau}\beta &= \tau e^{-\alpha\tau}(s-\alpha)e^{s\tau} && \Leftrightarrow && (4.47)\\
\tau e^{-\alpha\tau}\beta &= \tau(s-\alpha)e^{\tau(s-\alpha)} && && (4.48)
\end{aligned}$$

If we'd manage to derive an explicit formula for s from (4.48), we could substitute it in (4.42) and thereby solve (4.38). Note that by setting $a = \tau e^{-\alpha\tau}\beta$ and $b = \tau(s-\alpha)$, (4.48) can be written as:

$$a = be^b \tag{4.49}$$

which is true if and only if $b = W(a)$. Thus, we get:

$$\begin{aligned}
W(\tau e^{-\alpha\tau}\beta) &= \tau(s-\alpha) && \Leftrightarrow\\
s &= \frac{1}{\tau}W(\beta\tau e^{-\alpha\tau}) + \alpha
\end{aligned}$$

We have already noted that there exist infinitely many branches of the Lambert W function, denoted W_k with $k \in \mathbb{Z}$. This means that there are also infinitely many choices for s. Furthermore, any linear combination of solutions of (4.38) is again a solution of (4.38). Hence, for an arbitrary[4] family of constants $(C_k)_{k\in\mathbb{Z}} \subset \mathbb{C}$,

$$x(t) = \sum_{k=-\infty}^{\infty} C_k e^{\left(\frac{1}{\tau}W_k(\beta\tau e^{-\alpha\tau})+\alpha\right)t} \tag{4.50}$$

with $t \in [0, \infty)$ is a solution of (4.38).

In order to find the particular solution satisfying (4.39), we simply have to choose $(C_k)_{k\in\mathbb{Z}} \subset \mathbb{C}$ such that:

$$f(t) = \sum_{k=-\infty}^{\infty} C_k e^{\left(\frac{1}{\tau}W_k(\beta\tau e^{-\alpha\tau})+\alpha\right)t} \tag{4.51}$$

for all $t \in [-\tau, 0]$. This is indeed possible, as it can be shown that the set of functions $\left(e^{\left(\frac{1}{\tau}W_k(\beta\tau e^{-\alpha\tau})+\alpha\right)t}\right)_{k\in\mathbb{Z}}$ is dense in $C([-T, 0], \mathbb{R})$. □

Remark 4.2.4. *Indeed, there is no explicit formula for computing the coefficients $C_k \in \mathbb{C}$ in* (4.40)*. However, they can be approximated arbitrarily well by simply minimizing the squared error*

$$SE = \sum_{j=0}^{n}\left(f(t_j) - \sum_{k\in I} C_k e^{\left(\frac{1}{\tau}W_k(\beta\tau e^{-\alpha\tau})+\alpha\right)t_j}\right) \tag{4.52}$$

[4]Of course, at least one C_k has to be non-zero.

where $n \in \mathbb{N}$, $I \subset \mathbb{Z}$ is a finite index set and $t_j = -\tau\frac{j}{n}$. A similar approach is also proposed in the appendix of [31].

Based on theorem 4.2.3, the following result can be derived for the non-homogeneous case.

Theorem 4.2.5. *Let a non-homogeneous and linear initial value problem be given by:*

$$\frac{dx(t)}{dt} = \alpha x(t) + \beta x(t-\tau) + u(t) \qquad \forall t \in [0,\infty) \tag{4.53}$$

$$x(t) = f(t) \qquad \forall t \in [-\tau, 0] \tag{4.54}$$

where $\alpha, \beta, \tau \in \mathbb{R}$, $f\colon [-\tau, 0] \to \mathbb{R}$ is a continuous initial function and $u\colon \mathbb{R} \to \mathbb{R}$ is a continuous input function.

The solution of the system (4.53), (4.54) is then given by:

$$x(t) = \sum_{k=-\infty}^{\infty} C_k e^{s_k t} + \int_0^t \sum_{k=-\infty}^{\infty} \left(e^{s_k(t-\sigma)} C'_k\right) u(\sigma) d\sigma \tag{4.55}$$

with appropriately chosen families of constants $(C_k)_{k\in\mathbb{Z}}, (C'_k)_{k\in\mathbb{Z}} \subset \mathbb{C}$ (see remarks 4.2.4 and 4.2.6) and

$$s_k = \frac{1}{\tau} W_k(\beta\tau e^{-\alpha\tau}) + \alpha. \tag{4.56}$$

where W_k denotes the k-th branch of the Lambert W function (for a precise derivation of s_k see the proof of theorem 4.2.3).

Proof. For a detailed derivation of equation (4.55) see [31] and [36]. □

Remark 4.2.6. *The computation of the family of constants $(C'_k)_{k\in\mathbb{Z}} \subset \mathbb{C}$ used in equation 4.55 can also be found in [36]. Unfortunately, however, it doesn't only depend on time but also on the input function $u\colon \mathbb{R} \to \mathbb{R}$. Concerning our main goal of deriving a Volterra-like series expansion, this is a significant drawback: being input-independent is an important feature of the kernels used in definition 4.1.1.*

4.3 Discussion

It was already indicated at the end of section 4.1 and in remark 4.2.6 that the input-output representation defined in theorem 4.2.5 is not a Volterra series expansion in the sense of definition 4.1.1. While it can be considered a minor issue that the left sum in equation 4.55 is not independent of t, the fact that the kernel $\sum_{k=-\infty}^{\infty} \left(e^{s_k(t-\sigma)} C'_k\right)$ depends on the input function $u(\cdot)$ is indeed a significant flaw. It causes one of the major merits of a Volterra series representation to vanish, namely that its kernel functions fully determine the behavior of the corresponding dynamical system. Hence, it might be impossible or at least extremely difficult to derive meaningful predictions concerning the impact changes in the parameters α and β have on the overall dynamical system from equation 4.55. In other words: I don't think that the result given in theorem 4.2.5 can live up to the hopes originally motivating this work.

Nonetheless, I think it's an intriguing result that despite the complex dynamics exhibited by delay systems, it is possible to analytically derive explicit input-output representations from

delay differential equations that go beyond the very limited scope of the method of steps that was introduced in section 3.3.

Concerning realizations of liquid state machines based on delay dynamical systems, I see three possibilities for continuing this work: Firstly, even though it doesn't look very promising, it could still be possible that information on how the parameters α and β are connected to the dynamics of the corresponding system are more accessible in equation 4.55 than in equation 4.53. Secondly, it might be possible to derive a proper Volterra series representation by using theorem 4.2.5 as a starting point. And finally, I think it could also be beneficial to investigate Volterra-kernels that were obtained numerically from realizations of Mackey-glass systems[5].

[5]Methods for numerically obtaining Volterra series expansions from nonlinear dynamical systems can be found in [28] and [27]

References

[1] Wolfgang Maass, Thomas Natschläger, and Henry Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14:2531–2560, 2002.

[2] Warren McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[3] Bernard Widrow and Michael A. Lehr. *The Handbook of Brain Theory and Neural Networks*, chapter Perceptrons, Adalines, and Backpropagaion, pages 719–724. MIT Press, 1995.

[4] Eduardo D. Sontag. *The Handbook of Brain Theory and Neural Networks*, chapter Automata and Neural Networks, pages 119–123. MIT Press, 1995.

[5] Wolfgang Maass and Henry Markram. On the computational power of recurrent circuits of spiking neurons. *Journal of Computer and System Sciences*, 69:593–616, 2004.

[6] Almut Schüz. *The Handbook of Brain Theory and Neural Networks*, chapter Neuroanatomy in a Computational Perpsective, pages 622–626. MIT Press, 1995.

[7] Simon Haykin. *Neural Networks A Comprehensive Foundation*. Prentice-Hall, 1999.

[8] John J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. USA*, 79:2554–2558, 1982.

[9] Rodney Douglas, Henry Markram, and Kevan Martin. *The Synaptic Organization of the Brain*, chapter Neocortex, pages 499–558. Oxford University Press, 2004.

[10] Gilad Silberberg, Anirudh Gupta, and Henry Markram. Stereotypy in neocortical microcircuits. *Trends in Neuroscience*, 25:227–230, 2002.

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

[12] Dirk Werner. *Funktionalanalysis*. Springer, 2011.

[13] Jean Dieudonné. *Foundations of Modern Analysis*. Academic Press, 1969.

[14] Stephen Boyd and Leon O. Chua. Fading memory and the problem of approximating nonlinear operators with volterra series. *IEEE Transactions on Circuits and Systems*, CAS-32:1150–1161, 1985.

[15] Wolfgang Maass, Thomas Natschläger, and Henry Markram. *Computational Neuroscience: A Comprehensive Approach*, chapter Computational models for generic cortical microcircuits. CRC-Press, 2002.

[16] L. Appeltant, M.C. Soriano, G. Van der Sande, J. Dacnkaert, S. Massar, J. Dambre, B. Schrauwen, C.R. Mirasso, and I. Fischer. Information processing using a single dynamical node as complex system. *Nature Communications*, 2(468), 2011.

[17] Dror G. Feitelson. *Optical Computing: A Survey for Computer Scientists*. MIT Press, 1988.

[18] Kristof Vandoorne, Wouter Dierckx, Benjamin Schrauwen, David Verstraeten, Roel Baets, Peter Bienstman, and Jan Van Vampenhout. Toward optical signal processing using photonic reservoir computing. *Optics Express*, 16:11182–11192, 2008.

[19] Kristof Vandoorne, Joni Dambre, David Verstraeten, Benjamin Schrauwen, and Peter Bienstman. Parallel reservoir computing using optical amplifiers. *IEEE Transactions on Neural Networks*, 22:1469–1481, 2011.

[20] Kensuke Ikeda and Kenji Matsumoto. High-dimensional chaotic behavior in systems with time-delayed feedback. *Physica*, 29D:223–2, 1987.

[21] I. Fischer, O. Hess, W. Elsässer, and E. Göbel. High-dimensional chaotic dynamics of an external cavity semiconductor laser. *Physical Review Letters*, 73:2188–2191, 1994.

[22] M. C. Mackey and L. Glass. Oscillation and chaos in physiological control systems. *Science*, 197:287–289, 1977.

[23] G. Doddington and T. Schalk. Speech recognition: turning theory to practice. *IEEE Spectrum*, 18:26–32, 1981.

[24] Herbert Jaeger. Adaptive nonlinear system identification with echo state networks. *Advances in Neural Information Processing*, 15:593–600, 2003.

[25] Michael Mackey, L. Glass. Mackey-glass equation. *Scholarpedia*, 4(12):6908, 2009.

[26] R. D. Driver. *Ordinary and Delay Differential Equations*. Springer-Verlag, 1977.

[27] Martin Schetzen. *The Volterra and Wiener Theories of Nonlinear Systems*. John Wiley & Sons, 1980.

[28] Wilson J. Rugh. *Nonlinear System Theory - The Volterra/Wiener Approach*. The Johns Hopkins University Press, 1981.

[29] Maurice Fréchet. Sur les fonctionnelles continues. *Annales scientifiques de L'Ecole Normale Superieure*, 27:193–216, 1910.

[30] Jeffrey J. Dacunha. Transition matrix and generlized matrix exponential via the peano-baker series. *Journal of Difference Equations and Applications*, 11:1245–1264, 2005.

[31] Farshid Maghami Asl and A. Galip Ulsoy. Analysis of a system of linear delay differential equations. *ASME J. Dynami Systems, Measurement and Control*, 125:215–233, 2003.

[32] Johann Heinrich Lambert. Observations variae in mathesin puram. *Acta Helvitica,*

physico-mathematio-anatomic-botanico-medica, 3:128–168, 1758.

[33] R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. Knuth. On the lambert w function. *Advances in Computational Mathematics*, 5:329–359, 1996.

[34] Constantin Caratheodory. *Theory of Functions of a Complex Variable*. Chelsea Publications, 1954.

[35] Nicolaas Govert de Bruijn. *Asymptotic Methods in Analysis*. North-Holland, 1961.

[36] Sun Yi and A. G. Ulsoy. Solution of a system of linear delay differential equations using the matrix lambert function. In *Proceedings of the 2006 American Control Conference*, 2006.

[37] Robert M. Corless, David J. Jeffrey, and Donald E. Knuth. A sequence of series for the lambert w function. In *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation, Maui, Hawaii*, 1997.