MASARYKOVA UNIVERZITA
Přírodovědecká fakulta
Ústav matematiky a statistiky

Diplomová práce

Brno 2025 **Štěpán Zapadlo**

MASARYKOVA
UNIVERZITA
PŘÍRODOVĚDECKÁ FAKULTA
ÚSTAV MATEMATIKY A STATISTIKY

Computational analysis in neuroscience

Diplomová práce

Štěpán Zapadlo

Vedoucí práce: doc. RNDr. Lenka Přibylová, Ph.D.

Brno 2025

Bibliografický záznam

Autor:	Štěpán Zapadlo Přírodovědecká fakulta, Masarykova univerzita Ústav matematiky a statistiky

Název práce:	Computational analysis in neuroscience
Studijní program:	Matematika
Studijní obor:	Modelování a výpočty
Vedoucí práce:	doc. RNDr. Lenka Přibylová, Ph.D.
Akademický rok:	2024/2025
Počet stran:	xiv + 94
Klíčová slova:	dynamické systémy; kontinuace; výpočetní neurologie; spřažené oscilátory; synchronizace

Bibliographic Entry

Author:	Štěpán Zapadlo Faculty of Science, Masaryk University Department of Mathematics and Statistics
Title of Thesis:	Computational analysis in neuroscience
Degree Programme:	Mathematics
Field of Study:	Modelling and computations
Supervisor:	doc. RNDr. Lenka Přibylová, Ph.D.
Academic Year:	2024/2025
Number of Pages:	xiv + 94
Keywords:	dynamical systems; continuation; computational neuroscience; coupled oscillators; synchronization

Abstrakt

Práce zkoumá potenciál dvou výpočetních metod pro analýzu biofyzikálně přesných modelů neuronů nebo neuronových shluků: relativně standardní přístup simulace na mřížce a kontinuační metoda založená na nových postupech. Obsahuje teoretické pojednání o dynamických a semidynamických systémech, zpožděných diferenciálních rovnicích, numerickém řešení diferenciálních rovnic a jednorozměrnou optimalizaci. Jsou uváženy různé fyziologické modely neuronu, z nichž každý se vztahuje k určitému neurosignálu (tzv. spiking, bursting atd.). V práci je představena metoda založená na simulaci spřaženého systému pro každou kombinaci parametrů na dané mřížce. Ta je postupně zlepšována a vyvíjena. Práce se dále zabývá numerickou kontinuací variety cyklů pro obyčejné i zpožděné diferenciální rovnice. Tato metodologie založená na teorii bifurkací je posléze aplikována na studovaný problém. V závěru jsou obě metody srovnány z různých pohledů.

Abstract

The thesis investigates the potential of two computational methods for analyzing biophysically accurate models of neurons or neural clusters: a relatively standard grid-based simulation approach and a continuation method based on novel methodologies. It covers the necessary theory of dynamical and semidynamical systems, delay differential equations, numerical integration, and one-dimensional optimization. Various physiological neuronal models are considered, each relating to a specific neuro-signal (spiking, bursting, etc.). An approach using a grid-based simulation of the coupled system for each parametric combination on the grid is introduced. It is then iteratively developed and improved. Afterward, a numerical continuation of the cycle manifold is discussed. This method based on the bifurcation theory of ODEs and DDEs is later applied to the studied problem. Finally, both approaches are compared from various perspectives.

MUNI
SCI

MASARYKOVA UNIVERZITA
PŘÍRODOVĚDECKÁ FAKULTA
KOTLÁŘSKÁ 2, 611 37 BRNO
IČ: 00216224
DIČ: CZ00216224

ZADÁNÍ DIPLOMOVÉ PRÁCE

Akademický rok: 2024/2025

Ústav:	Přírodovědecká fakulta
Student:	Bc. Štěpán Zapadlo
Program:	Aplikovaná matematika
Specializace:	Modelování a výpočty

Ředitel *ústavu* PřF MU Vám ve smyslu Studijního a zkušebního řádu MU určuje diplomovou práci s názvem:

Název práce:	Computational analysis in neuroscience
Název práce anglicky:	Computational analysis in neuroscience
Jazyk práce:	angličtina

Oficiální zadání:
The thesis will investigate the potential of computational methods for analyzing neuronal data and models. The aim will be to explore various novel algorithms and techniques related to neuronal signals and its modeling. Evaluate and compare different computational approaches and algorithms for analyzing neuroscience data, aiming to provide guidelines and best practices for researchers in the field.

Literatura:
KUZNECOV, Jurij Aleksandrovič. *Elements of applied bifurcation theory*. 2nd ed. New York: Springer-Verlag, 1998, xviii, 591. ISBN 0387983821.

BRUNTON, Steven L.; KUTZ, J. Nathan. Data-Driven Science and Enginee- ring: Machine Learning, Dynamical Systems, and Control. Cambridge University Press, 2019. Dostupné z doi: 10.1017/9781108380690

Vedoucí práce:	doc. RNDr. Lenka Přibylová, Ph.D.
Datum zadání práce:	18. 8. 2023
V Brně dne:	11. 3. 2025

Zadání bylo schváleno prostřednictvím IS MU.

Bc. Štěpán Zapadlo, 17. 10. 2023
doc. RNDr. Lenka Přibylová, Ph.D., 18. 10. 2023
RNDr. Jan Vondra, Ph.D., 19. 10. 2023

Poděkování

V prvé řadě bych chtěl poděkovat mé vedoucí doc. RNDr. Lence Přibylové, Ph.D., za její cenné rady, nadšení a nekonečnou trpělivost. Neméně významnou roli při vzniku této práce sehrál i celý tým Nelineární dynamiky, kterému bych chtěl taktéž vyjádřit svůj vděk. Děkuji také své partnerce Veronice za její neutuchající podporu a lásku, kterou mi opakovaně prokazovala spolu se dvěma nejlepšími psími konzultanty, Tuffym a Poincarém.

Tato práce byla podpořena z projektu *GAMU Interdisciplinary # MUNI/G/1213/2022* a projektu *Mathematical and Statistical Modeling # MUNI/A/1132/2022*. Výpočetní prostředky byly poskytnuty e-INFRA CZ projektem (ID:90254), podpořeným Ministerstvem školství, mládeže a tělovýchovy České republiky.

Prohlášení

Prohlašuji, že jsem tuto magisterskou práci vypracoval samostatně pod vedením vedoucí práce s využitím informačních zdrojů, které jsou v práci citovány.

Brno 05.05.2025

.....
Štěpán Zapadlo

Table of contents

Used Notation

For easier orientation in the text, we include a basic list of used notation throughout the text:

Symbol	Meaning
$\mathbb{R}, \mathbb{R}^+$	set of all real numbers, $\{x \in \mathbb{R} \mid x > 0\}$
$\mathbb{N}, \mathbb{N}_0$	set of all natural numbers, $\mathbb{N} \cup \{0\}$
$\|\cdot\|_k$	appropriate (vector, function, etc.) L^k-norm
$x, \boldsymbol{x}, \mathrm{X}, \mathbf{X}$	scalar, vector, random scalar, random vector variable, respectively
$\langle \boldsymbol{u}, \boldsymbol{v} \rangle$	scalar (dot) product between vectors $\boldsymbol{u}$ and $\boldsymbol{v}$
$f, \boldsymbol{f}$	scalar-valued, vector-valued function
$f(x)'$	first derivative of a scalar-valued function f with respect to scalar variable x
$\dot{x}(t), \dot{\boldsymbol{x}}(t)$	derivative of a scalar, vector variable (dependent on time) with respect to time t
$f^{(m)}(x), \boldsymbol{f}^{(m)}(x)$	m-th derivatives of a scalar-valued and vector-valued function with respect to a scalar variable x, respectively
$\nabla f, \nabla \boldsymbol{f}, \nabla_{\boldsymbol{y}} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y})$	gradient of f, Jacobian matrix of $\boldsymbol{f}$, Jacobian matrix of $\boldsymbol{f}$ with respect to the 2nd vector argument $\boldsymbol{y}$
J	Jacobian matrix of a corresponding function $\boldsymbol{f}$
$\Re x, \Im x$	real and imaginary parts of complex variable x, respectively
$[n]$	$\{i \in \mathbb{N} \mid i \leq n\} = \{1, 2, \ldots, n-1, n\}$
$\mathbb{X}, \mathbb{X}_C$	state space of a dynamical system, often $\mathbb{R}^n$, state space of a semidynamical system, often $C([-\tau, 0], \mathbb{X})$
$\mathbb{T}$	algebraic structure capturing time for (semi)dynamical system, often $(\mathbb{R}, +)$
$\boldsymbol{M}$	monodromy matrix of a cycle

Symbol	Meaning
$\boldsymbol{x}_t$	DDE history segment, $\boldsymbol{x}_t(s) := \boldsymbol{x}(t+s)$, $-\tau \le s \le 0$
$\nabla_{\tau_i} \boldsymbol{f}$	Jacobian matrix of $\boldsymbol{f}$ with respect to the state variable corresponding to delay τ_i
Inc	increment function associated with a given numerical ODE solver
err, le, lte	global, local and local truncation error of a numerical integration method, respectively
$\mathbb{M}, \mathscr{I}[\mathbb{M}]$	discretization of a given section of 2-parameter plane to a grid, its 2D indices
$\mathscr{M}, \mathscr{C}$	manifold, typically of a cycle, curve
$(a, b\rangle$	interval from a to b open on the left and open or closed on the right
Φ	phase condition for a collocation problem
S, A	solution operator, infinitesimal generator of a linearized DDE
φ	evolution operator of a (semi)dynamical system
$\mathscr{N}(\mu, \sigma^2)$	normal distribution with expected value μ and variance σ^2
$\mathrm{W}(t)$	Wiener process
$\mathscr{B}$	Borel σ-algebra

Introduction

> **Tip 1: Web version**
>
> This thesis is also available in web version[1].

Synchronization is a phenomenon commonly found in nature and, as is often the case, it takes many shapes and forms from the coordination between neurons in our brains or fire-fly lights to synchronization found in electrical grids and financial markets. What started in the 17th century with Huygens' investigation into the behavior of two weakly interacting pendulum clocks through a heavy beam, see [1], has since evolved into a field rich with both theory and applications.

The key application and motivation of this thesis lies, as the name may suggest, in physiological models of neurons in the brain. Several studies, see [2], [3], [4], [5], and [6] among others, have shown that fast to ultra-fast ripples (oscillations of frequencies ranging from 250 Hz to 2000 Hz, or even higher in the case of *ultra-fast oscillations*) in intracranial electroencephalographic (iEEG) recordings measured deep in the brain can be potentially used as biomarkers of epileptogenic zones of focal epilepsy. Furthermore, there is evidence they correlate with the severity of epilepsy. Research also suggests that very high-frequency oscillations (VHFOs) and ultra-fast oscillations (UFOs) are more local, i.e., spatially restricted than traditional biomarkers, high-frequency oscillations (HFOs), thus providing better guidance in locating the areas of epilepsy. Fast oscillations lie outside the realm of physiologically possible frequencies of single neurons. This indicates another mechanism must be at play, but its identification is an open question in neuroscience, see [7].

The primary goal of this thesis is to provide insight into a small part of computational analysis in neuroscience — phase synchronization[2] of coupled neurons and techniques used in its detection and exploration. In particular, we will study two possible approaches and analyze their strengths and shortcomings.

[1] https://thesis.uni.zapadlo.name/

[2] By phase synchronization we mean the state when the difference in phase of the oscillators remains bounded while, e.g., their amplitudes may differ, see [8].

Note that VFHOs and UFOs in the context of coupled neuronal models pose a very difficult problem from the point of view of applied mathematics. While in this thesis we only study numerical simulations of neuronal models, colleagues from the Nonlinear Dynamics team have explored the detection of VHFOs in iEEG signals, see [9]. Moreover, the scope and focus of the thesis, i.e., computation of periods (frequencies) and shifts of simulated weakly coupled neuronal models, arose naturally as an extension of research conducted in the Nonlinear Dynamics team in the course of the GAMU Interdisciplinary project # MUNI/G/1213/2022. This research direction was fueled by the necessity of understanding regions and types of synchrony of coupled neurons in the vicinity of an epileptogenic zone.

First and foremost, we shall introduce the necessary theory. Starting with a theoretical introduction to dynamical systems, we then shift our attention to delay differential equations and semidynamical systems induced by them. A discussion of methods of numerical integration for both ordinary and delay differential equations follows. Lastly, 1D optimization methods are studied, together with Newton's method, before shortly turning our attention to the models and principles from computational neuroscience.

In the second chapter, an approach for the computation of phase shift between two weakly coupled oscillators based on simulations of the corresponding differential equation is iteratively developed. Starting from a simple, naive algorithm, we progressively add various concepts to address present issues. We end this thesis by taking a different route and introducing the theory of localization of equilibria and periodic orbits. We then use these concepts from bifurcation theory to compute the phase shifts as continuations of cycles on a specific manifold.

Let us note the thesis is typeset using Quarto [10] and all figures are generated in Julia [11] using the Makie.jl package [12]. Both HTML and PDF versions are available. All source code for this thesis (and included numerical computations) is included in the Git repository[3] (if not stated otherwise).

[3] https://github.com/sceptri/masters-thesis

Chapter 1

Mathematics Behind Dynamics

We have motivated the entire thesis with the usefulness of the knowledge and the understanding of synchronization in neuroscience. However, we will not describe any experiments performed on real networks of neurons in a lab. Instead, we shall deal with the mathematical abstraction for the studied object, e.g., coupled neurons.

This abstraction is typically called a (mathematical) model. The model should, in theory, capture all the important characteristics of the underlying reality. If the state of the model evolves causally in time, e.g., a model of a neuron starts spiking when evolving in time, with dependence solely on the current state, we often use a *dynamical system* as the model. On the other hand, if the governing rules are noncausal with respect to only the current state due to the dependence on the history of the system, the arising dynamical system becomes infinite dimensional (and thus harder to work with). For better "mathematical ergonomics" we will define a *semidynamical system* to deal with delayed dependencies.

1.1 A Primer on Dynamical Systems

First and foremost, we will formally introduce the notion of a *dynamical system*, along with its properties. This section is mostly based on [13] and [14] in terms of content. It should be noted the notation follows [15], [16], [17], or [18] for better compatibility with semidynamical systems. Also, [19] was used as a valuable inspiration for the structure of this section.

Definition 1.1. Dynamical system (more precisely *deterministic dynamical system*) is a triple $(\mathbb{T}, \mathbb{X}, \varphi)$, where $(\mathbb{T}, +, \leq)$, $\mathbb{T} \subseteq \mathbb{R}$, is an ordered commutative semigroup with identity element 0, $\mathbb{X}$ (called *state space*) is a metric space, and φ is a continu-

ous[1] evolution map

$$\begin{aligned} \varphi : \mathbb{T} \times \mathbb{T} \times \mathbb{X} &\to \mathbb{X} \\ (t, s, x) &\mapsto \varphi(t, s, x), \end{aligned}$$

projecting a state x from time s to time t, which satisfies

1. (*deterministic property*): $\varphi(s, s, x) = x$ for each $x \in \mathbb{X}$ and $s \in \mathbb{T}$,
2. (*composition property*): $\varphi(t, s, x) = \varphi(t, r, \varphi(r, s, x))$ for each $x \in \mathbb{X}$ and $t, r, s \in \mathbb{T}$.

If $(\mathbb{T}, +)$ forms a group, we shall speak of an **invertible dynamical system**.

For simplicity, we shall assume the commutative monoid $(\mathbb{T}, +)$ is *cancellative*, i.e., $a + c = b + c \implies a = b$ for each $a, b, c \in \mathbb{T}$. Then it can be embedded in a Grothendieck group $\mathbb{T}_G$, see [20].

Definition 1.2. Autonomous dynamical system is a dynamical system such that the *autonomous property*,

$$\varphi(t, s, x) = \varphi(t + r, s + r, x) \tag{1.1}$$

for each $x \in \mathbb{X}$, $t, s \in \mathbb{T}$, and $r \in \mathbb{T}_G$, is fulfilled.

> **Note 1: Evolution operator as monoid action**
>
> In the case of Definition 1.2, we can always vanish the starting time by taking $r = -s$ in (1.1), which is ensured by $r \in \mathbb{T}_G$. Then, by abuse of notation, we will write $\varphi : \mathbb{T} \times \mathbb{X} \to \mathbb{X}$. Moreover, the *composition* and *autonomous* properties combine into the *semigroup property*,
>
> $$\varphi(t, \varphi(s, x)) = \varphi(t + s, x) \; \forall x \in \mathbb{X}, \forall t, s \in \mathbb{T},$$
>
> i.e., the map φ is a (left) monoid action of $(\mathbb{T}, +)$ on $\mathbb{X}$. The operator φ is typically called *flow* of the dynamical system.

In Definition 1.1, the time set $\mathbb{T}$ can take on various forms. In ecology, we often see a discrete $\mathbb{T} = \mathbb{Z}$ or $\mathbb{N}_0$ representing a yearly intervals between measurements of our system — given this choice of $\mathbb{T}$, we call the dynamical system **discrete**.

On the other hand, in physics (and neuroscience) we typically employ $\mathbb{T} = \mathbb{R}$ or $\mathbb{R}^+$ as we are concerned with even the shortest time intervals and associated changes — corresponding systems are called **continuous** or **continuous-time** dynamical systems. Similarly, the exact choice of the state space $\mathbb{X}$ is dependent on the system in question, but typically we use $\mathbb{R}^n$, although we will also encounter infinite-dimensional state space.

The *deterministic* property in Definition 1.1 guarantees the system does not abruptly

[1] The evolution map φ is continuous, if $t, s \in \mathbb{T}$, $x \in \mathbb{X}$ and if $\{(t_n, s_n)\}_n$ is a sequence in $\mathbb{T} \times \mathbb{T}$ and $\{x_n\}_n$ is a sequence in $\mathbb{X}$ such that $(t_n, s_n) \to (t, s)$ and $x_n \to x$, then $\varphi(t_n, s_n, x_n) \to \varphi(t, s, x)$.

change state. Should this condition be broken, we would call such a system *stochastic*, see Section 1.1.5. Moreover, a common assumption that the "laws of nature" do not change over time restricts us to autonomous dynamical systems, which will be exclusively used in this thesis, where it is captured by the *semigroup* (or *autonomous*) property. In particular, dependence on the past is permitted, but requires special attention, as we shall see in Section 1.2. If the system is *invertible*, the *(semi)group property* implies that the solution can be found both forward and backward in time.

Most often, a dynamical system is given implicitly by a *differential* or *difference* equation. For example, in population dynamics, one of the earliest models, the *Verhulst model of population*[2], can prescribed by an ordinary differential equation (ODE)

$$\dot{x}(t) = \frac{\mathrm{d}x(t)}{\mathrm{d}t} = x(t) \cdot r_0 \cdot \left(1 - \frac{x(t)}{K}\right), \tag{1.2}$$

or a *difference equation* (although these two models are *not* equivalent)

$$x(t+1) = x(t) \cdot r_0 \cdot \left(1 - \frac{x(t)}{K}\right). \tag{1.3}$$

1.1.1 Basic Concepts

In this section, we shall introduce basic concepts regarding dynamical systems, including, but not limited to, notions of certain special solutions and their stability. Little comment besides the definitions themselves will be provided, as an interested reader can find much more in [14], [21], or [13].

Definition 1.3 (Orbit)**.** An *orbit (trajectory)* with an *initial point* $x_0 \in \mathbb{X}$ is an ordered subset of the state space $\mathbb{X}$,

$$Or(x_0) = \{x \in \mathbb{X} \mid x = \varphi(t, x_0), \forall t \in \mathbb{T} \text{ such that } \varphi(t, x_0) \text{ is defined}\}$$

In the case of a continuous dynamical system, the orbits are *oriented curves* in the state space. For a discrete dynamical system, they become sequences of points in $\mathbb{X}$.

Definition 1.4 (Phase portrait)**.** A *phase portrait* of a dynamical system is a partitioning of the state space into trajectories.

Definition 1.5 (Equilibrium)**.** A point $x^* \in \mathbb{X}$ is called an *equilibrium* (fixed point) if $\varphi(t, x^*) = x^*$ for all $t \in \mathbb{T}$.

Definition 1.6 (Cycle)**.** A *cycle* is a periodic orbit, namely a non-equilibrium orbit L, such that each point $x_0 \in L$ satisfies $\varphi(t + T, x_0) = \varphi(t, x_0)$ with some $T > 0$, for all $t \in \mathbb{T}$. The smallest admissible T is called the *period* of the cycle L.

[2] The equation (1.2) describes the Verhulst model of a population (and its growth, characterized by r_0) in some closed environment with some finite capacity (controlled by K).

Definition 1.7 (Invariant set)**.** An *invariant set* of a dynamical system $(\mathbb{T}, \mathbb{X}, \varphi)$ is a subset $\mathbb{S} \subset \mathbb{X}$ which satisfies

$$x \in \mathbb{S} \implies \varphi(t, x) \in \mathbb{S} \; \forall t \in \mathbb{T}.$$

Definition 1.8 (ω-limit and α-limit point)**.** A point $x_* \in \mathbb{X}$ is called an *ω-limit point* (resp. α-limit point) of the orbit $Or(x_0)$ starting at $x_0 \in \mathbb{X}$ if their exists a sequence of times $\{t_k\}_{k=1}^{\infty} \subseteq \mathbb{T}$ with $t_k \to \infty$ (resp. $t_k \to -\infty$)[3], such that

$$\varphi(t_k, x_0) \underset{k\to\infty}{\longrightarrow} x_*.$$

Definition 1.9 (ω-limit and α-limit set)**.** A set $\Omega(Or(x_0))$ of all ω-limit points of the orbit $Or(x_0)$, see Definition 1.8, is called an ω-limit set. Similarly, a set $\mathbb{A}(Or(x_0))$ of all α-limit points of the orbit $Or(x_0)$ is called an α-limit set.

The set $\Lambda(Or(x_0)) = \Omega(Or(x_0)) \cup \mathbb{A}(Or(x_0))$ of all limit points of the orbit $Or(x_0)$ is called its *limit set*.

Definition 1.10 (Limit cycle)**.** A cycle of a continuous-time dynamical system, in a neighborhood of which there are no other cycles, is called a *limit cycle*.

> **i** Note 2: Equivalent definition of a limit cycle
>
> Equivalently to Definition 1.10, one can define a *limit cycle* as a cycle, which is the limit set, see Definition 1.9, of orbits in its neighborhood.

Definition 1.11. An invariant set S_0 is called:

1. **Lyapunov stable** if for any sufficiently small neighborhood $U \supset S_0$ there exists a neighborhood $V \supset S_0$ such that $\varphi(t, x) \in U$ for all $x \in V$ and all $t > 0$;
2. **asymptotically stable** if there exists a neighborhood $U_0 \supset S_0$ such that $\varphi(t, x) \to S_0$ for all $x \in U_0$, as $t \to +\infty$;
3. **stable** if it is both *Lyapunov* and *asymptotically stable*;
4. **unstable** if it is not *stable*.

A stable invariant set is called an *attractor*, whereas if the invariant set is unstable it is called a *repeller*.

1.1.1.1 Topologically Equivalent Dynamical Systems

So far, we have described dynamical systems mainly in general terms. Later, we will present concrete examples of dynamical systems, primarily from the neuroscience field, see Section 1.4, which will be too complex to apply certain techniques from bifurcation theory directly. Hence, we introduce the notion of (local) topological equivalence to remedy this issue.

[3] For a careful reader, let us note we assume $t_k \to \infty$ is well-defined, though it might restrict full generality of Definition 1.1.

Definition 1.12 (Topological equivalence)**.** A dynamical system $\mathbb{D}_1 = (\mathbb{T}, \mathbb{R}^n, \varphi)$ is said to be *topologically equivalent* to a dynamical system $\mathbb{D}_2 = (\mathbb{T}, \mathbb{R}^n, \psi)$, if there exists a *homeomorphism*[4] $\boldsymbol{h} : \mathbb{R}^n \to \mathbb{R}^n$, which maps orbits of system $\mathbb{D}_1$ to orbits of system $\mathbb{D}_2$, such that their orientation is kept. In such a case, their phase portraits are called *(topologically) equivalent*.

Definition 1.13 (Local topological equivalence)**.** A dynamical system $\mathbb{D}_1 = (\mathbb{T}, \mathbb{R}^n, \varphi)$ is said to be *locally topologically equivalent* in the neighborhood of its equilibrium $\boldsymbol{x}^*$ to a dynamical system $\mathbb{D}_2 = (\mathbb{T}, \mathbb{R}^n, \psi)$ in the neighborhood of its equilibrium $\boldsymbol{y}^*$, if there exists a *homeomorphism* $\boldsymbol{h} : \mathbb{R}^n \to \mathbb{R}^n$, such that

1. $\boldsymbol{h}$ is defined on a (small) neighborhood $\mathscr{O}(\boldsymbol{x}^*) \subset \mathbb{R}^n$,
2. satisfies $\boldsymbol{y}^* = \boldsymbol{h}(\boldsymbol{x}^*)$ and
3. maps orbits of dynamical system $\mathbb{D}_1$ in the neighborhood $\mathscr{O}(\boldsymbol{x}_0)$ to orbits of system $\mathbb{D}_2$ in the neighborhood $\mathscr{O}(\boldsymbol{y}^*)$, such that their orientation is kept.

1.1.2 Continuous-time Autonomous Systems

Definition 1.14. An *autonomous system of (ordinary) differential equations* is a system of the form

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}), \tag{1.4}$$

where $\boldsymbol{x} \in \mathbb{X} = \mathbb{R}^n$ and a vector-valued function $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n$ is sufficiently smooth. The symbol $\dot{\boldsymbol{x}}$ denotes a derivative of $\boldsymbol{x}(t)$ with respect to time $t \in \mathbb{T} = \mathbb{R}$.

It can be shown the system of ODEs (1.4) induces an *invertible continuous-time dynamical system* if $\boldsymbol{f}$ is sufficiently smooth, see [13] (Theorem 1.4).

In (1.4), we have given the ODE in explicit form, but, in general, it can also be defined by an implicit higher-order equation

$$\boldsymbol{F}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \ddot{\boldsymbol{x}}, \ldots, \boldsymbol{x}^{(k)}) = \boldsymbol{0}. \tag{1.5}$$

Such a system can always be translated into a first-order system of equations

$$\boldsymbol{G}(\boldsymbol{y}, \dot{\boldsymbol{y}}) = \boldsymbol{0}$$

and, if $\boldsymbol{F}$ (or $\boldsymbol{G}$, respectively) allows it, even to the explicit form (1.4).

> **Tip 2: Vector field**
>
> The function $\boldsymbol{f}$ is called a *vector field*, as it maps a vector $\boldsymbol{f}(\boldsymbol{x})$ to each point $\boldsymbol{x}$ of the state space. See Figure 1.1 for an example.

[4] In the context of topology, a *homeomorphism* (also called a *bicontinuous function*) is a bijective and continuous function, such that its inverse is also continuous.

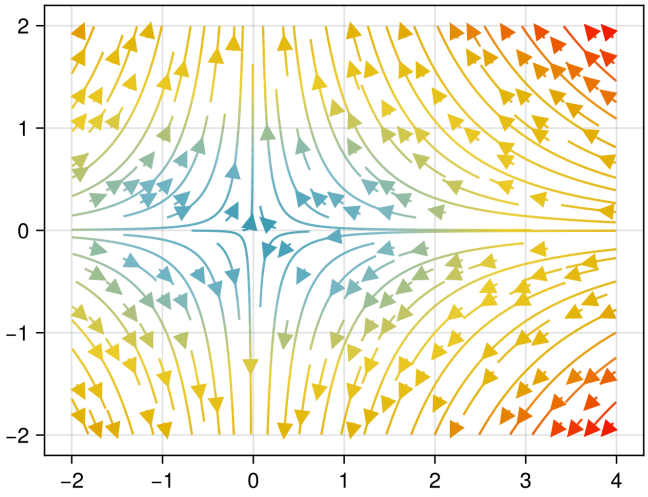


Figure 1.1: Vector field (or a phase portrait) corresponding to the dynamical system $\dot{x} = -x, \dot{y} = 2y$.

Theorem 1.1 (Lyapunov). *Consider a dynamical system (1.4) with an equilibrium $\boldsymbol{x}^*$. Let*

$$\mathrm{J}^* = \nabla \boldsymbol{f}(\boldsymbol{x}^*) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(\boldsymbol{x}^*) & \frac{\partial f_1}{\partial x_2}(\boldsymbol{x}^*) & \dots & \frac{\partial f_1}{\partial x_n}(\boldsymbol{x}^*) \\ \frac{\partial f_2}{\partial x_1}(\boldsymbol{x}^*) & \frac{\partial f_2}{\partial x_2}(\boldsymbol{x}^*) & \dots & \frac{\partial f_2}{\partial x_n}(\boldsymbol{x}^*) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1}(\boldsymbol{x}^*) & \frac{\partial f_n}{\partial x_2}(\boldsymbol{x}^*) & \dots & \frac{\partial f_n}{\partial x_n}(\boldsymbol{x}^*) \end{pmatrix}$$

denote a Jacobian matrix evaluated at $\boldsymbol{x}^*$. Then $\boldsymbol{x}^*$ is stable, *if all eigenvalues λ_i, where $i \in [n]$[5], of the matrix J^* satisfy $\Re\, \lambda_i < 0$.*

Proof. See [22], page 160, or [23], page 185. □

Definition 1.15 (Hyperbolic equilibrium). An equilibrium $\boldsymbol{x}^*$ of the system (1.4) is called *hyperbolic* if none of the eigenvalues corresponding to the Jacobian matrix $\mathrm{J}^* = \nabla \boldsymbol{f}(\boldsymbol{x}^*)$ lies on the imaginary axis.

Theorem 1.2 (Grobman-Hartman). *The system (1.4) in a neighborhood of its* hyperbolic *equilibrium $\boldsymbol{x}^*$ is* locally topologically equivalent, *in the sense of Definition 1.13, to its linearization*

$$\dot{\boldsymbol{x}} = \nabla \boldsymbol{f}(\boldsymbol{x}^*)\boldsymbol{x}. \tag{1.6}$$

Proof. See [22], page 306, or [23], page 120. □

This theorem is very prominently used in conjunction with *linear* continuous-time

[5]For conciseness, we use the following notation $[n] := \{1, \dots, n\}$.

dynamical systems, i.e.,

$$\dot{\boldsymbol{y}} = \boldsymbol{A}\boldsymbol{y},$$

where $\boldsymbol{y} \in \mathbb{R}^n$ and $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, with an equilibrium $\boldsymbol{y}^*$. Thus, we can partition the state space into disjoint linear (vector) subspaces:

- *stable subspace* $\mathbb{E}^s = \operatorname{span}\{\boldsymbol{v}^1, \dots, \boldsymbol{v}^{n_-}\}$ of dimension n_-, where $\boldsymbol{v}^1, \dots, \boldsymbol{v}^{n_-}$ are the eigenvectors of $\boldsymbol{A}$ corresponding to eigenvalues with *negative* real parts;
- *unstable subspace* $\mathbb{E}^u = \operatorname{span}\{\boldsymbol{u}^1, \dots, \boldsymbol{u}^{n_+}\}$ of dimension n_+, where $\boldsymbol{u}^1, \dots, \boldsymbol{u}^{n_+}$ are the eigenvectors of $\boldsymbol{A}$ corresponding to eigenvalues with *positive* real parts;
- *center subspace* $\mathbb{E}^c = \operatorname{span}\{\boldsymbol{w}^1, \dots, \boldsymbol{w}^{n_0}\}$ of dimension n_0, where $\boldsymbol{w}^1, \dots, \boldsymbol{w}^{n_0}$ are the eigenvectors of $\boldsymbol{A}$ corresponding to *strictly imaginary* eigenvalues, i.e., eigenvalues with zero real part.

Then it holds that $n_- + n_+ + n_0 = n$. Similarly, for linear discrete-time dynamical systems, one can perform an equivalent partitioning, such that $\mathbb{E}^s$ is spanned by eigenvectors corresponding to eigenvalues *lying inside the unit circle* (on the imaginary plane), see Theorem 1.5 below. Furthermore, $\mathbb{E}^u$ and $\mathbb{E}^c$ can be defined for discrete-time dynamical systems analogously.

Moreover, by Theorem 1.2 we know that these subspaces will be preserved in some neighborhood of the *hyperbolic* equilibrium of the nonlinear system (1.4). Specifically, because the equilibrium is *hyperbolic*, $\mathbb{E}^c = \{\boldsymbol{0}\}$.

1.1.2.1 Lyapunov's Direct Method

Definition 1.16. A function $V : \mathbb{R}^n \to \mathbb{R}$ is called *locally positive-definite (LPD)* at $\boldsymbol{x}^*$, if the following holds:

1. $V(\boldsymbol{x}^*) = 0$,
2. $V(\boldsymbol{x}) > 0$, $\boldsymbol{x} \in \mathcal{O}(\boldsymbol{x}^*) \setminus \{\boldsymbol{x}^*\}$ for some neighborhood $\mathcal{O}(\boldsymbol{x}^*)$.

If only $V(\boldsymbol{x}) \geq 0$ holds on a neighborhood $\mathcal{O}(\boldsymbol{x}^*)$, then the function is called *locally positive semi-definite (LPSD)*. Analogously, we can define a *locally negative definite* and *semi-definite* functions.

Definition 1.17 (Lyapunov function)**.** Let $\boldsymbol{\varphi}(t; \boldsymbol{x}_0)$ be a solution of the system (1.4) together with the initial $\boldsymbol{x}(0) = \boldsymbol{x}_0$. A function $V : \mathbb{R}^n \to \mathbb{R}$ is called *Lyapunov* at $\boldsymbol{x}^*$, if V is locally positive definite and also $\forall \boldsymbol{x}_0 \in \mathcal{O}(\boldsymbol{x}^*)$ is the function $V \circ \boldsymbol{\varphi}(t; \boldsymbol{x}_0)$ *non-increasing* for all $t > 0$.

Moreover, the function V is called *strictly Lyapunov* if $V \circ \boldsymbol{\varphi}(t; \boldsymbol{x}_0)$ is *(strictly) decreasing* for all $t > 0$.

> **i** Note 3: Monotonicity of $V \circ \boldsymbol{\varphi}(t; \boldsymbol{x}_0)$
>
> Equivalently to requiring a non-increasing $V \circ \boldsymbol{\varphi}(t; \boldsymbol{x}_0)$, one can instead demand the *derivatives with respect to the trajectories* of V, i.e., $\dot{V}(\boldsymbol{x}(t))$, to be non-positive. In other words, the *derivative w.r.t. the trajectories* $\dot{V}$ must be *locally negative*

semi-definite.

Theorem 1.3 (Lyapunov's direct method)**.** *If $\boldsymbol{x}^*$ is an equilibrium of the system (1.4) and V is a* Lyapunov *function for the system at $\boldsymbol{x}^*$, then $\boldsymbol{x}^*$ is* Lyapunov stable*. If, in addition, V is a* strict Lyapunov *function, then $\boldsymbol{x}^*$ is* stable*.*

Proof. See [22], page 24. □

1.1.3 Discrete Dynamical Systems

Definition 1.18. An *autonomous system of difference equations* is a system of the form

$$\boldsymbol{x} \mapsto \boldsymbol{f}(\boldsymbol{x}) \quad \Longleftrightarrow \quad \boldsymbol{x}_{m+1} = \boldsymbol{f}(\boldsymbol{x}_m), \tag{1.7}$$

where $\boldsymbol{x}, \boldsymbol{x}_m \in \mathbb{X} = \mathbb{R}^n$ and the function $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n$ is sufficiently smooth.

The system (1.7) induces a *discrete-time autonomous dynamical system*.

Theorem 1.4 (Lyapunov, analogous to Theorem 1.1)**.** *Consider a dynamical system (1.7) with a fixed point $\boldsymbol{x}^*$. Let $\mathrm{J}^* = \nabla \boldsymbol{f}(\boldsymbol{x}^*)$ denote the Jacobian matrix evaluated at $\boldsymbol{x}^*$. Then $\boldsymbol{x}^*$ is stable, if all eigenvalues*[6] *λ_i, where $i \in [n]$, of the matrix J^* satisfy $|\lambda_i| < 1$.*

Proof. See [24], page 222. □

Definition 1.19 (Hyperbolic fixed point)**.** A fixed point $\boldsymbol{x}^*$ of the system (1.7) is called **hyperbolic** if none of the eigenvalues corresponding to the Jacobian matrix $\mathrm{J}^* = \nabla \boldsymbol{f}(\boldsymbol{x}^*)$ have unit magnitude.

Theorem 1.5 (Grobman-Hartman, analogous to Theorem 1.2)**.** *The system (1.7) is locally topologically equivalent in the neighborhood of its* hyperbolic *fixed point $\boldsymbol{x}^*$ to its linearization*

$$\boldsymbol{x} \mapsto \nabla \boldsymbol{f}(\boldsymbol{x}^*)\boldsymbol{x}.$$

Proof. See [22], page 311. □

Example 1.1 (Fixed points of two-dimensional discrete-time dynamical system)**.** As an example, consider a two-dimensional discrete-time dynamical system

$$\boldsymbol{x}_{m+1} = \boldsymbol{f}(\boldsymbol{x}_m), \tag{1.8}$$

where $\boldsymbol{x}_m = (x_{m,1}, x_{m,2})^\top$ and $\boldsymbol{f} : \mathbb{R}^2 \to \mathbb{R}^2$ is smooth. Moreover, let us assume there exists a *hyperbolic equilibrium* $\boldsymbol{x}^* = \boldsymbol{f}(\boldsymbol{x}^*)$ of the system (1.8) and let $\mathrm{J}^* = \nabla \boldsymbol{f}(\boldsymbol{x}^*)$ denote the corresponding Jacobian matrix evaluated at $\boldsymbol{x}^*$. Then J^* has two eigenvalues λ_1, λ_2, which satisfy

$$\det(\mathrm{J}^* - \lambda \boldsymbol{I}_2) = \lambda^2 - \operatorname{tr} \mathrm{J}^* \lambda + \det \mathrm{J}^*.$$

[6] Eigenvalues of fixed points of discrete-time dynamical systems are often called *multipliers*, see [19].

Here, I_2 corresponds to a 2-by-2 identity matrix, $\operatorname{tr} \mathrm{J}^* = \lambda_1 + \lambda_2$ is the trace of the determinant and finally, $\det \mathrm{J}^* = \lambda_1 \lambda_2$ denotes the determinant of J^*.

By Theorem 1.5, we know that the system (1.8) is locally topologically equivalent to its linearization in a neighborhood of its hyperbolic fixed point $\boldsymbol{x}^*$. Interestingly, we can classify the said fixed point based on the number of stable (and unstable) eigenvectors of the corresponding J^*, i.e., the classification is based on the dimensions of $\mathbb{E}^s$, $\mathbb{E}^u$ and $\mathbb{E}^c$ of the partitioning of the state space per linearization.

A fixed point is classified as a *sink*, when both eigenvalues are real with a modulus below 1. Similarly, it is called a *spiral sink* if both eigenvalues are complex with modulus less than 1. Should one eigenvalue have modulus below 1 and the other modulus greater than 1, the resulting fixed point is a *saddle*. Finally, an equilibrium is a *source*, resp. *spiral source*, when both eigenvalues have modulus above 1 and are *real*, resp. *complex*. For an overview, see Figure 1.2.

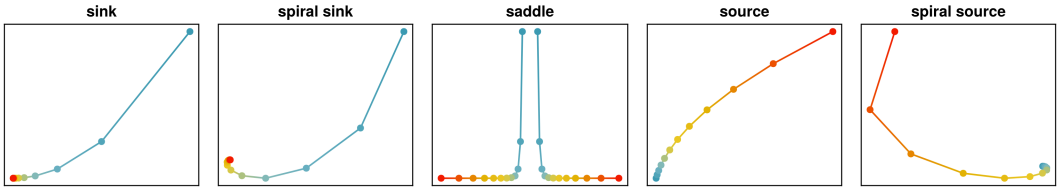


Figure 1.2: Orbits near different equilibria of a linear discrete-time dynamical system (time flows from blue to red).

For completeness' sake, we give a classical result from the theory of ODEs concerning the existence, uniqueness, and smooth dependence on the initial conditions of the solution for a given ODE.

Theorem 1.6 (Existence, uniqueness, and smooth dependence)**.** *Consider a system of ordinary differential equations*

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{x} \in \mathbb{R},$$

where $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n$ *is smooth in an open region* $U \subset \mathbb{R}^n$*. Then there is a unique function* $\boldsymbol{x} = \boldsymbol{x}(t, \boldsymbol{x}_0)$, $\boldsymbol{x} : \mathbb{R}^1 \times \mathbb{R}^n \to \mathbb{R}^n$*, that is smooth in* (t, x)*, and satisfies, for each* $\boldsymbol{x}_0 \in U$*, the following conditions:*

1. $\boldsymbol{x}(0, \boldsymbol{x}_0) = \boldsymbol{x}_0$*;*
2. *there is an interval* $\mathcal{I} = (-\delta_1, \delta_2)$*, where* $\delta_{1,2} = \delta_{1,2}(\boldsymbol{x}_0) > 0$*, such that, for all* $t \in \mathcal{I}$*,*

$$\boldsymbol{y}(t) = \boldsymbol{x}(t, \boldsymbol{x}_0) \in U,$$

and

$$\dot{\boldsymbol{y}}(t) = \boldsymbol{f}(\boldsymbol{y}(t)).$$

Proof. See [21], page 94. □

1.1.4 Poincaré Map

The study of continuous-time dynamical systems naturally leads to discrete-time dynamical systems. One possible route is via sampling the continuous orbit at discrete times, e.g., separated by Δt which induces a *time-shift map*. Another way to obtain a discrete-time dynamical system from a continuous-time one is through a so-called *Poincaré map*.

Consider a continuous-time dynamical system of the form

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n, \tag{1.9}$$

where $\boldsymbol{f}$ is smooth and assume that (1.9) has a periodic orbit L_0. Let $\boldsymbol{x}_0$ be a point on L_0 and denote Σ the *cross-section* to the cycle at this point, see Figure 1.3.

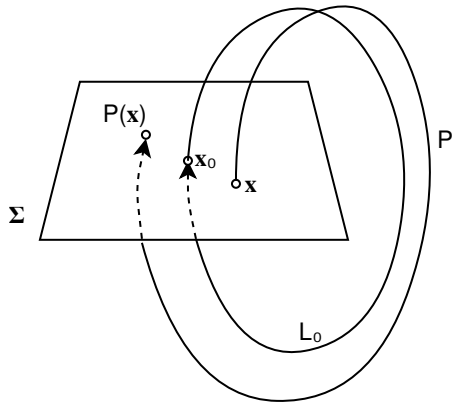


Figure 1.3: Poincaré map corresponding to the system (1.9), the cycle L_0 and its point $\boldsymbol{x}_0$.

The cross-section Σ is a smooth hypersurface of dimension $n-1$ (thus we say codim $\Sigma = 1$, i.e., the cross-section hypersurface Σ is of codimension one), which intersects L_0 at a nonzero angle. The nonzero angle requirement is called the *transversality* condition, which effectively dictates that the hypersurface is not parallel to a through-going trajectory, thus the trajectory truly intersects the cross-section Σ.

Consider now orbits of (1.9) near the cycle L_0 and recall that the cycle L_0 itself is an orbit which starts at point $\boldsymbol{x}_0$ on Σ and returns to the same point on Σ. As Theorem 1.6 guarantees, the solution of (1.9) depends smoothly on its initial condition, an orbit starting at $\boldsymbol{x} \in \Sigma$ sufficiently close to $\boldsymbol{x}_0$ will transversally intersect the hypersurface Σ at some other point $\tilde{\boldsymbol{x}}$ near $\boldsymbol{x}_0$. Therefore, this induces a map $\boldsymbol{P} : \Sigma \to \Sigma$,

$$\boldsymbol{x} \mapsto \tilde{\boldsymbol{x}} = \boldsymbol{P}(\boldsymbol{x}).$$

Definition 1.20 (Poincaré map)**.** The map $\boldsymbol{P}$ defined above is called a *Poincaré map* associated with the cycle L_0.

Similarly, we can characterize the Poincaré map $\boldsymbol{P}$ using local coordinates $\boldsymbol{\xi} = (\xi_1, \dots, \xi_{n-1})$ on Σ, such that the choice $\boldsymbol{\xi} = \mathbf{0}$ corresponds to $\boldsymbol{x}_0$. Then the Poincaré map can be locally defined as a function $\boldsymbol{P} : \mathbb{R}^{n-1} \to \mathbb{R}^{n-1}$, which maps $\boldsymbol{\xi}$ corresponding to $\boldsymbol{x}$ to $\tilde{\boldsymbol{\xi}}$ corresponding to $\tilde{\boldsymbol{x}}$, i.e.,

$$\boldsymbol{P}(\boldsymbol{\xi}) = \tilde{\boldsymbol{\xi}}.$$

In other words, the Poincaré map $\boldsymbol{P}$ prescribes a *discrete-time dynamical system* on the hypersurface Σ. Its origin $\boldsymbol{\xi} = \mathbf{0}$ is a fixed point of this mapping. To our advantage, the stability of the underlying cycle L_0 is then equivalent to the stability of the fixed point $\boldsymbol{\xi}_0 = \mathbf{0}$. By Theorem 1.4, we know the cycle is thus stable if all eigenvalues (also called *multipliers*) $\mu_1, \dots, \mu_{n-1}$ of the $(n-1) \times (n-1)$ Jacobian matrix of $\boldsymbol{P}$,

$$\mathrm{J}_{\boldsymbol{P}} = \nabla_{\boldsymbol{\xi}} \boldsymbol{P}(\boldsymbol{\xi}_0),$$

are located inside the unit circle $\|\boldsymbol{\mu}\| = 1$. The Poincaré map will throughout this thesis paint itself as a powerful tool in the bifurcation analysis of dynamical systems, and the following lemma hints at its usefulness.

Lemma 1.1. *The multipliers $\mu_1, \dots, \mu_{n-1}$ of the Jacobian matrix $\mathrm{J}_{\boldsymbol{P}}$ of the Poincaré map $\boldsymbol{P}$ associated with a cycle L_0 are independent of the point $\boldsymbol{x}_0$ on L_0, the cross-section Σ, and local coordinates on it.*

Proof. See [13], page 27. □

Consider now the cycle L_0 and let $\boldsymbol{x}^0(t)$ denote its corresponding periodic solution of (1.9) with the period T_0, i.e., $\boldsymbol{x}^0(t + T_0) = \boldsymbol{x}^0(t)$. Then any solution $\boldsymbol{x}(t)$ of (1.9) can be written as

$$\boldsymbol{x}(t) = \boldsymbol{x}^0(t) + \boldsymbol{u}(t),$$

where $\boldsymbol{u}(t)$ stands for the deviation of the solution from the referential periodic solution. Then,

$$\begin{aligned} \dot{\boldsymbol{u}}(t) &= \dot{\boldsymbol{x}}(t) - \dot{\boldsymbol{x}}^0(t) \\ &= \boldsymbol{f}(\boldsymbol{x}^0(t) + \boldsymbol{u}(t)) - \boldsymbol{f}(\boldsymbol{x}^0(t)) \\ &= \mathrm{J}(t)\boldsymbol{u}(t) + O(\|\boldsymbol{u}(t)\|^2). \end{aligned}$$

Omitting $O(\|\boldsymbol{u}(t)\|^2)$ terms gives us a linear T_0-periodic system

$$\dot{\boldsymbol{u}} = \mathrm{J}(t)\boldsymbol{u}, \quad \boldsymbol{u} \in \mathbb{R}^n, \tag{1.10}$$

where $\mathrm{J}(t) = \nabla_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x}^0(t))$ and $\mathrm{J}(t + T_0) = \mathrm{J}(t)$.

Definition 1.21. System (1.10) is called the *variational equation* about the cycle L_0.

As the variational equation describes the evolution of perturbations in the proximity of the cycle L_0, its stability naturally depends on the properties of the variational equation.

Definition 1.22. The time-dependent matrix $\boldsymbol{M}(t)$ is called the *fundamental matrix solution* of (1.9) if it satisfies

$$\dot{\boldsymbol{M}} = \mathrm{J}(t)\boldsymbol{M},$$

with the initial condition $\boldsymbol{M}(0) = \boldsymbol{I}_n$. The matrix $\boldsymbol{M}(T_0)$ is called a **monodromy matrix** of the cycle L_0.

Theorem 1.7 (Floquet exponents)**.** *The monodromy matrix $\boldsymbol{M}(T_0)$ has eigenvalues (called* Floquet exponents *or* multipliers*)*

$$1, \mu_1, \dots, \mu_{n-1},$$

where μ_i are the multipliers of the Poincaré map $\boldsymbol{P}$ associated with the cycle L_0, see Lemma 1.1.

Proof. See [13], page 30, for a sketch of the proof. □

1.1.5 Stochastic Dynamical Systems

So far, we have only considered deterministic dynamical systems. Their defining feature was the absence of abrupt changes, i.e., the total lack of randomness. Contrastingly, in practice one has to constantly deal with noise in measurements of the studied system. This effectively results in a stochastic dynamical system induced by the stochastic random process arising from the measurements. Let us note this section is primarily based on [25].

Let us first introduce the necessary theory of stochastic analysis.

Definition 1.23 (Stochastic process)**.** Let $(\Omega, \mathscr{A}, P)$ be a **probability space** and $\mathbb{T} \subseteq \mathbb{R}$ some index set of times. Then **random/stochastic process** $\{\mathrm{X}(t,\omega);\ t \in \mathbb{T}\}$ is a mapping

$$\mathrm{X} : \mathbb{T} \times \Omega \to \mathbb{R},$$

which is $\mathscr{A}$-measurable, i.e.,[7]

$$\forall t \in \mathbb{T},\ \forall B \in \mathscr{B} :\ \{\omega \in \Omega :\ \mathrm{X}(t,\omega) \in B\} \in \mathscr{A}.$$

For a fixed $\omega \in \Omega$, the mapping $\mathrm{X}(t) = \mathrm{X}(t,\omega)$ is called a **trajectory**. Similarly, for a fixed $t \in \mathbb{T}$ the resulting $\mathrm{X}(\omega) = \mathrm{X}(t,\omega)$ **is a random variable**.

Definition 1.24 (Wiener process)**. Wiener process**, sometimes also called **Brownian motion**, is a random process $\{\mathrm{W}(t,\omega);\ t \geq 0\}$ satisfying

1. $\mathrm{W}(0,\omega) = 0$,
2. trajectories are *almost surely* continuous, i.e.,

$$P(\{\omega \in \Omega :\ \text{trajectory } t \mapsto \mathrm{W}(t,\omega) \text{ is discontinuous}\}) = 0,$$

[7] We denote $\mathscr{B}$ the set of all Borel sets, i.e., Borel σ-algebra.

3. for all $0 \leq s_1 < t_1 \leq s_2 < t_2 \cdots \leq s_k < t_k \leq \cdots$, the **increments** of Wiener process
$$\{\Delta \mathrm{W}_1(\omega), \Delta \mathrm{W}_2(\omega), \dots\}$$
are **stochastically independent** random variables with Gaussian probability distribution
$$\Delta \mathrm{W}_k \sim \mathcal{N}(0, t_k - s_k),$$
where $\Delta \mathrm{W}_k(\omega) = \mathrm{W}(t_k, \omega) - \mathrm{W}(s_k, \omega)$ for $k = 1, 2, \cdots$.

Theorem 1.8 (Properties of Wiener process)**.** *Let* $\{\mathrm{W}(t);\ t \geq 0\}$ *be Wiener process. Then, for all* $t, s \geq 0$*, it holds*[8]

1. $\mu(t) = 0$,
2. $\gamma(t,t) = t$,
3. $\mathrm{W}(t) \sim \mathcal{N}(0,t)$,
4. $\gamma(s,t) = \min\{s,t\}$,
5. $\rho(s,t) = \sqrt{\frac{\min\{s,t\}}{\max\{s,t\}}}$.

Proof. See [25], page 28. □

In other words, in Definition 1.23, we have constructed a process that selects a random trajectory with each sample. Moreover, we have defined in Definition 1.24 a special process that is almost always continuous everywhere but differentiable nowhere (not unlike the Weierstrass function) with **stochastically independent** increments on disjunct intervals!

Our goal in this endeavour is to construct a sensible calculus on stochastic processes, which we could then use to understand deterministic dynamical systems perturbed by (small) stochastic noise. Let us consider a (trivial) differential equation (mainly focus on the differential formulation)
$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = ax(t) \iff \mathrm{d}x(t) = ax(t)\,\mathrm{d}t,$$
which has the solution $x(t) = e^{at}$, where $x(t)$ often represents size of some population. Taking $a = r + \sigma\varepsilon(t,\omega)$, where $\varepsilon(t,\omega)$ is a random process, the population size now becomes a random variable,
$$\mathrm{dX}(t,\omega) = (r + \sigma\varepsilon(t,\omega))\,\mathrm{X}(t,\omega)\,\mathrm{d}t,$$
which is a **stochastic differential equation** (SDE). For a moment, let us use *differences* instead of *differentials*, i.e.,
$$\begin{aligned}\Delta \mathrm{X}(t,\omega) &= (r + \sigma\varepsilon(t,\omega))\,\mathrm{X}(t,\omega)\,\Delta t \\ &= r\mathrm{X}(t,\omega)\,\Delta t + \sigma \mathrm{X}(t,\omega)\underbrace{\varepsilon(t)\,\Delta t}_{\Delta Z(t)}.\end{aligned}$$

[8] Let us note $\gamma(s,t) = \mathrm{cov}(\mathrm{X}(s),\mathrm{X}(t)) = \mathbb{E}\left((\mathrm{X}(s) - \mathbb{E}\,\mathrm{X}(s))(\mathrm{X}(t) - \mathbb{E}\,\mathrm{X}(t))\right)$ denotes the *autocovariance function* and $\rho(s,t) = \mathrm{cor}(\mathrm{X}(s),\mathrm{X}(t)) = \frac{\mathrm{cov}(\mathrm{X}(s),\mathrm{X}(t))}{\sqrt{\mathrm{var}\,\mathrm{X}(s)\,\mathrm{var}\,\mathrm{X}(t)}}$ the *autocorrelation function* of a stochastic process $X(t)$.

What is the form of the random process $Z(t)$, such that $\Delta Z(t) = \varepsilon(t)\,\Delta t$, where $\varepsilon(t)$ is a *Gaussian white noise*? To a careful reader, this may seem at least a little reminiscent of the Wiener process. In other words, differences $\Delta Z(t)$ are *Gaussian white noise*[9] and if we require (almost sure) continuity and $Z(0) = 0$, we get that $Z(t)$ is the Wiener process — that is, $\Delta W(t)$ has the same properties as $\varepsilon(t)\Delta t$. With the limit transition $\lim_{\Delta t \to 0}$ in the L^2 sense we get, arguably after much more mathematics,

$$\mathrm{d}W(t) = \varepsilon(t)\,\mathrm{d}t,$$

which can be understood as a *distributional derivative*. Now let us return to the original stochastic differential equation, that now becomes

$$\mathrm{d}X(t,\omega) = rX(t,\omega)\,\mathrm{d}t + \sigma X(t,\omega)\overbrace{\varepsilon(t,\omega)\,\mathrm{d}t}^{\mathrm{d}W(t)},$$

which we are tasked with solving for a random process $X(t)$, $t \geq 0$, that satisfies this SDE. This rather simple SDE prescribes so-called *geometric Brownian motion*. In general, a random process $X(t)$ is called **Itô process**, if it is a solution to

$$\mathrm{d}X(t) = U(t)\,\mathrm{d}t + V(t)\,\mathrm{d}W(t),\ X(0) = X_0,$$

where $U(t,\omega), V(t,\omega)$ are random processes satisfying certain conditions and X_0 is required to have finite second moment.

1.2 Dynamical Systems with Delays

To understand how delays influence dynamical systems and define a *semidynamical system*, we have to generalize our notion of a differential equation. This modification will rely on introducing a deviating argument, i.e., some of the derivatives of the differential equation will be evaluated at different argument values. This section will be primarily sourced from [26], [27], [28], [29], [18], and [16].

1.2.1 Functional Differential Equations

So far, we have mainly discussed *ordinary differential equations*, which are equations relating the values of an unknown function and some of its derivatives at a single and the same argument value, see (1.4) and (1.5), e.g., $F(t,x,\dot{x},\ddot{x}) = 0$.

In contrast, a *functional equation* (FE) is an equation connecting outputs of an unknown function evaluated at different argument values. As an example, $x(-t) + 3x(2t) = 0$, $x(x(t)) = x(t) + 2$, and $x(t) = tx(t+1) - (x(t-2))^3$ are all examples of functional equations. The differences between the argument values of an unknown function and t (its "default" argument) in an FE are called *argument deviations*. If all argument deviations are constant (like in the last example shown above), then the FE is called a *difference-functional equation*.

[9] A random variable is called Gaussian white noise if it has normal distribution with zero mean.

Although we have so far shown only examples of FEs with *discrete* (or *concentrated*) argument deviations, another possibility is FEs with *continuous* and *mixed* (both continuous and discrete) *argument deviations*. They are called *integral* and *integral-functional* (or *integral-difference*) equations.

We can further combine these notions of ordinary differential equations and functional equations, yielding in the process *functional differential equations* (FDEs). FDEs can also contain discrete, continuous, or mixed argument deviations. Thus, one can introduce *differential-difference equations*, and *integro-differential equations* in a similar way as seen above.

The aforementioned generalization in its full form leads to *functional differential equations* — equations describing the relation between the unknown function and some of its derivatives for, in general, different argument values. Our main subject of study will be a subclass of *differential-difference equations* called the *functional differential equations with aftereffects*, which are of the form

$$\boldsymbol{x}^{(m)}(t) = \boldsymbol{f}\left(t, \boldsymbol{x}^{(m_1)}(t-\tau_1(t)), \dots, \boldsymbol{x}^{(m_k)}(t-\tau_k(t))\right),$$

where $\boldsymbol{x}(t) \in \mathbb{R}^n$, $m_1, \dots, m_k \geq 0$, and $\tau_1(t), \dots, \tau_k(t) \geq 0$. They can be further classified as:

- *retarded FDEs* (RFDEs) or equivalently *delay differential equations* (DDEs), if $\max\{m_1, \dots, m_k\} < m$;
- *neutral FDEs* (NFDEs), if $\max\{m_1, \dots, m_k\} = m$;
- *advanced FDEs* (AFDEs), if $\max\{m_1, \dots, m_k\} > m$.

Let us also note that the argument deviations can, in theory, be dependent on the state, e.g., $\dot{x}(t) = f(t, x(t), x(t - h(t, x(t))))$, but we shall omit them from our considerations.

> **Caution 1: Note on naming convention**
>
> For clarity and brevity, throughout this thesis *RFDEs with (discrete) constant delays* will be called *delay differential equations* (DDEs), although the term can be more general in other literature.

1.2.2 Method of Steps

All throughout this work, we shall see many parallels between ODEs and DDEs. While their connection is intricate and would require a more thorough discussion, one way it can be partly understood is via the *method of steps* — a way of solving delay differential equations. This subsection is mainly based on [18] and [30].

To begin gently, we will start with a simple case of $x \in \mathbb{R}$. Let us thus consider a nonlinear DDE of the form

$$\dot{x}(t) = f(t, x(t), x(t-r)) \tag{1.11}$$

with a single delay $r > 0$. Assume that $f(t, x, y)$ and $f_x(t, x, y)$ are continuous on $\mathbb{R}^3$. Let $s \in \mathbb{R}$ and a continuous *history* function $\phi : [s - r, s] \to \mathbb{R}$ be given. We aim to find a solution of (1.11) such that

$$x(t) = \phi(t), \ s - r \le t \le s \tag{1.12}$$

and satisfying (1.11) on $s \le t \le s + \sigma$ for some $\sigma > 0$. As a careful reader might notice, we must interpret $\dot{x}(s)$ as a *right-hand* derivative at s, i.e., $\dot{x}(s)^+$.

The system of equations (1.11) and (1.12) can now be solved with the so-called *method of steps* as follows. For $s \le t \le s + r$, $x(t)$ must satisfy the initial-value problem (IVP) for the following ODE:

$$\dot{y}(t) = \underbrace{f(t, y(t), \phi(t - r))}_{g(t,y)}, \ y(s) = \phi(s), \ s \le t \le s + r.$$

From the continuity of f and f_x immediately follows the continuity of g and g_y, thus the existence of a unique local solution is guaranteed by classic results from the ODE theory, see, for example, [22]. Moreover, if this local solution $x(t)$ exists on the entire interval $s \le t \le s + r$, then, together with the history ϕ, we know the solution $x(t)$ of (1.11)-(1.12) on $[s - r, s + r]$ and we may repeat the presented argument to extend our solution to the right. Indeed, considering now the interval $s + r \le t \le s + 2r$, a solution of $x(t)$ of (1.11)-(1.12) must now satisfy the following ODE problem:

$$\begin{aligned} \dot{y}(t) &= f(t, y(t), x(t - r)), \quad s + r \le t \le s + 2r, \\ y(s + r) &= x(s + r). \end{aligned} \tag{1.13}$$

Just as before, the continuity of f and f_x yields a locally unique solution called, by abuse of notation, again $x(t)$, defined on some subinterval $[s + r, \sigma) \subset [s + r, s + 2r]$ or, possibly, the entire interval. Naturally, $x(t)$, now existing on $[s - r, \sigma)$ where $\sigma > s + r$, is again a solution of (1.11)-(1.12). Finally, if the solution exists on the entire interval $[s + r, s + 2r]$ we can perform another *step*, i.e., repeat this procedure, to extend $x(t)$ further to the right to $[s + 2r, s + 3r]$ or some subinterval.

Theorem 1.9 (Single-delay method of steps on $\mathbb{R}$)**.** *Let $f(t, x, y)$ and $f_x(x, y, t)$ be continuous on $\mathbb{R}^3$, $s \in \mathbb{R}$, and let $\phi : [s - r, s] \to \mathbb{R}$ be continuous history function. Then there exists $\sigma > s$ and a unique solution of the initial-value problem (1.11)-(1.12) on $[s - r, \sigma]$*

Proof. Follows directly from the calculations above. □

A careful reader might notice that Theorem 1.9 guarantees only a local solution, but we might want a solution for any $t \ge s$. We will use the notation $[s - r, \sigma\rangle$ to denote either the right-open interval, $[s - r, \sigma)$, or the right-closed one, $[s - r, \sigma]$.

From the uniqueness follows that for two solutions $x(t)$ on $[s - r, \sigma\rangle$ and $\hat{x}(t)$ on $[s - r, \rho\rangle$, the equality $x(t) = \hat{x}(t)$ must hold for all t, such that both sides of the

equality are defined. Moreover, if $[s-r,\sigma\rangle \subset [s-r,\rho\rangle$, then $\hat{x}$ is called an *extension* of x and we write $x \subset \hat{x}$.

One can prove there exists a *unique* maximally defined solution, i.e., one for which there are no extensions, $x : [s-r,\sigma) \to \mathbb{R}$, similarly as for ODEs, for $\sigma \in \mathbb{R} \cup \{\infty\}$. Such a solution is called *non-extendable* and is necessarily defined on a right-open interval $[s-r,\sigma)$. Indeed, if $\sigma < \infty$ and x were a solution on $[s-r,\sigma]$, then, by Theorem 1.9, it could be extended to a larger interval, which contradicts non-extendability of x.

Theorem 1.10 (Finite-time blow-up)**.** *Let f satisfy the conditions of Theorem 1.9 and let $x : [s-r,\sigma) \to \mathbb{R}$, $\sigma \in \mathbb{R} \cup \{\infty\}$, be the non-extendable solution of the initial value problem (1.11)-(1.12). If $\sigma < \infty$, then*

$$\lim_{t \to \sigma^-} |x(t)| = \infty.$$

Proof. See [18], page 26. □

> **Note 4: On multiple delays and $\boldsymbol{x} \in \mathbb{R}^n$**
>
> One can rather easily show that Theorem 1.9 and Theorem 1.10 extend to the case of $\boldsymbol{x}(\cdot) \in \mathbb{R}^n$ and $\boldsymbol{f} : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$. Similarly, it can also be generalized to multiple discrete delays $r_1 < \cdots < r_H$ where
>
> $$\boldsymbol{f} = \boldsymbol{f}(t, \boldsymbol{y}(t), \boldsymbol{y}(t-r_1), \ldots, \boldsymbol{y}(t-r_H))$$
>
> with little to no change.

So far, we have presented a method to calculate the solution of the IVP given by (1.11) and (1.12). Per Note 4, we can slightly generalize the studied equation and consider

$$\left.\begin{aligned} \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{x}(t-r)), \quad & t_0 \le t \le t_f, \\ \boldsymbol{x}(t) = \boldsymbol{\phi}(t), \quad & t \le t_0, \end{aligned}\right\} \tag{1.14}$$

where $\boldsymbol{x}(\cdot) \in \mathbb{R}^n$ and $\boldsymbol{f} : [t_0, t_f] \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$. From the discussion above follows that a unique solution exists for continuously differentiable function $\boldsymbol{f}$. Together with (1.14), we further obtain that $\boldsymbol{x}(\cdot)$ is also differentiable on each interval $(t_0 + kr, t_0 + (k+1)r) \subseteq (t_0, t_f)$ for appropriate $k \in \mathbb{N}_0$. Moreover, $\boldsymbol{x}(\cdot)$ is necessarily continuous on $[t_0, t_f]$ by (1.13) and Theorem 1.9.

However, the initial history function $\boldsymbol{\phi}$ might not *link smoothly* to the actual solution $\boldsymbol{x}$, i.e.,

$$\dot{\boldsymbol{\phi}}(t_0)^- \neq \dot{\boldsymbol{x}}(t_0)^+ = \boldsymbol{f}(t_0, \boldsymbol{\phi}(t_0), \boldsymbol{\phi}(t_0 - r)), \tag{1.15}$$

where $\dot{\boldsymbol{\phi}}(\cdot)^-$ denotes the left derivative of $\boldsymbol{\phi}$ with respect to t at t_0 and analogously for $\dot{\boldsymbol{x}}(t_0)^+$. As such, the discontinuity is then *propagated* to $t_0 + r$, where, if $\boldsymbol{f}$ and $\boldsymbol{\phi}$ are continuous, $\dot{\boldsymbol{x}}$ is now *continuous* but not *differentiable*. In general, the number and location of discontinuity points depends, in principle, on the so-called *deviated*

arguments,

$$\alpha(t) = t - r(t, \boldsymbol{x}(t)), \tag{1.16}$$

where we assume a more general version of (1.14) with $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{x}(\alpha(t)))$. For our purposes, we have $\alpha(t) = t - r$ (then $\alpha \in C^\infty(\mathbb{R})$).

Definition 1.25 (Vanishing delay)**.** Consider the generalized version of (1.14) with $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{x}(\alpha(t)))$. Then we say such system has a *vanishing delay*, if there exists a point $s \in (t_0, t_f)$ such that the delay $\alpha(t)$ *vanishes*, i.e.,

$$\alpha(s) = s.$$

Theorem 1.11. *Suppose s is a vanishing delay point of (1.14) with $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{x}(\alpha(t)))$ and α is continuous. Moreover assume (1.15) holds, i.e., the history $\boldsymbol{\phi}$ does not link smoothly to the actual solution $\boldsymbol{x}$. Then, there are infinitely many discontinuity points in any left neighborhood of s.*

Proof. See [30], page 24. □

Should $\boldsymbol{f}$, $\boldsymbol{\phi}$ and α be also differentiable, then $\ddot{\boldsymbol{x}}(t)$ exists for any t except the solutions $s_{1,i} \in (t_0, t_f)$ of

$$\alpha(s_{1,i}) = t_0 \quad \& \quad \alpha'(s_{1,i}) \neq 0, \tag{1.17}$$

i.e., the simple roots of $\alpha(s) - t_0 = 0$. We call $s_{1,i}$ the *first level primary discontinuities* (of $\boldsymbol{x}(\cdot)$). Indeed, for any smooth function $\boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{\xi})$ we can formally write (see (1.14) and (1.16))

$$\begin{aligned} \ddot{\boldsymbol{x}}(t)^{\pm} = {} & \nabla_t \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{x}(\alpha(t))) \\ & + \nabla_{\boldsymbol{x}} \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{x}(\alpha(t)))\dot{\boldsymbol{x}}(t) \\ & + \nabla_{\boldsymbol{\xi}} \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{x}(\alpha(t)))\dot{\boldsymbol{x}}(\alpha(t))^{\pm}\alpha'(t), \end{aligned} \tag{1.18}$$

therefore by (1.17)

$$\begin{aligned} \ddot{\boldsymbol{x}}(s_{1,i})^{+} = {} & \nabla_t \boldsymbol{f}(s_{1,i}, \boldsymbol{x}(s_{1,i}), \boldsymbol{x}(t_0)) \\ & + \nabla_{\boldsymbol{x}} \boldsymbol{f}(s_{1,i}, \boldsymbol{x}(s_{1,i}), \boldsymbol{x}(t_0))\dot{\boldsymbol{x}}(s_{1,i}) \\ & + \nabla_{\boldsymbol{\xi}} \boldsymbol{f}(s_{1,i}, \boldsymbol{x}(s_{1,i}), \boldsymbol{x}(t_0))\dot{\boldsymbol{x}}(t_0)^{+}\alpha'(s_{1,i}) \end{aligned} \tag{1.19}$$

and

$$\begin{aligned} \ddot{\boldsymbol{x}}(s_{1,i})^{-} = {} & \nabla_t \boldsymbol{f}(s_{1,i}, \boldsymbol{x}(s_{1,i}), \boldsymbol{x}(t_0)) \\ & + \nabla_{\boldsymbol{x}} \boldsymbol{f}(s_{1,i}, \boldsymbol{x}(s_{1,i}), \boldsymbol{x}(t_0))\dot{\boldsymbol{x}}(s_{1,i}) \\ & + \nabla_{\boldsymbol{\xi}} \boldsymbol{f}(s_{1,i}, \boldsymbol{x}(s_{1,i}), \boldsymbol{x}(t_0))\dot{\boldsymbol{\phi}}(t_0)^{-}\dot{\alpha}(s_{1,i}). \end{aligned} \tag{1.20}$$

By comparing (1.19) and (1.20) using (1.15), we show that $\ddot{\boldsymbol{x}}(s_{1,i})^{+} \neq \ddot{\boldsymbol{x}}(s_{1,2})^{-}$, i.e., the second derivative of $\boldsymbol{x}(\cdot)$ has a jump discontinuity at $s_{1,i}$ as we claimed. Furthermore, by differentiating (1.18) we obtain that 1-level primary discontinuities propagate to *2-level primary discontinuities* in $\ddot{\boldsymbol{x}}(\cdot)$ at any point $s_{2,j} > s_{1,i}$, such that $s_{2,j}$ is a simple root of $\alpha(s) - s_{1,i} = 0$ for some i.

In other words, if $\boldsymbol{f}$ is smooth and $\boldsymbol{\phi}$ is differentiable for (1.14), then $t_0 + r$ is the only 1-level primary discontinuity, $t_0 + 2r$ is the 2-level primary discontinuity, etc. Therefore, the solution $\boldsymbol{x}(t)$ of (1.14) is progressively smoother as the primary discontinuity level increases, i.e., as we go forward in time. This phenomenon is often called the *smoothing of the solution* of a DDE.

1.2.3 Notes on Delay Differential Equations

Consider a DDE of the form

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{x}(t-\tau_1), \dots, \boldsymbol{x}(t-\tau_H)), \tag{1.21}$$

with $0 = \tau_0 < \tau_1 < \cdots < \tau_H$. For brevity, we will denote $\tau := \tau_H$ and

$$\boldsymbol{x}_t(s) := \boldsymbol{x}(t+s), \ -\tau \le s \le 0,$$

which, informally, prescribes the value of solution $\boldsymbol{x}$ up to time t delayed by s. In other words, we can view the trajectory of our solution as the curve $t \to \boldsymbol{x}_t$ in the state space $\mathbb{X}_C = C([-\tau, 0], \mathbb{R}^n)$ with the supremum norm

$$\|\boldsymbol{\phi}\| = \sup_{-\tau \le s \le 0} \|\boldsymbol{\phi}(s)\|,$$

a Banach space of continuous functions mapping the interval $[-\tau, 0]$ to $\mathbb{R}^n$ with the topology of uniform convergence, see [27]. Thus, we can reformulate (1.21) as

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{F}(t, \boldsymbol{x}_t), \tag{1.22}$$

which can generally represent any RFDE, with $\boldsymbol{F} : \mathbb{T} \times \mathbb{X}_C \to \mathbb{R}^n$ defined as

$$\boldsymbol{F}(t, \boldsymbol{\phi}) = \boldsymbol{f}(t, \boldsymbol{\phi}(-\tau_0), \boldsymbol{\phi}(-\tau_1), \dots, \boldsymbol{\phi}(-\tau_H)). \tag{1.23}$$

Most often, we will encounter the following *autonomous* initial-value problem

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{x}(t-\tau_1), \dots, \boldsymbol{x}(t-\tau_H)), \\ \boldsymbol{x}_\sigma &= \boldsymbol{\phi}, \end{aligned} \tag{1.24}$$

where $\sigma \in \mathbb{R}$ is the initial time and $\boldsymbol{\phi} \in \mathbb{X}_C$ is the state of system at time σ, i.e., the τ-long history function corresponding to the interval $[\sigma - \tau, \sigma]$. For completeness' sake, we also add that general the RFDE initial-value problem reads as follows

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{F}(t, \boldsymbol{x}_t), \\ \boldsymbol{x}_\sigma &= \boldsymbol{\phi}. \end{aligned} \tag{1.25}$$

Last, but not least, we shall denote by

$$\nabla_{\tau_i} \boldsymbol{f} = \frac{\partial}{\partial \boldsymbol{\xi}_i} \boldsymbol{f}(t, \overbrace{\boldsymbol{x}(t-\tau)}^{\xi_0}, \overbrace{\boldsymbol{x}(t-\tau_1)}^{\xi_1}, \dots, \overbrace{\boldsymbol{x}(t-\tau_H)}^{\xi_H})$$

the Jacobian matrix of $\boldsymbol{f}$ corresponding to the i-th delayed state input. In other words, we may write the linearization of (1.24) around equilibrium $\boldsymbol{x}^*$, i.e., $\boldsymbol{f}(\boldsymbol{x}^*, \dots, \boldsymbol{x}^*) = \boldsymbol{0}$, as

$$\boldsymbol{f}(\boldsymbol{x}(t)) \approx \boldsymbol{f}(\boldsymbol{x}^*, \dots, \boldsymbol{x}^*) + \sum_{i=0}^{H} \nabla_{\tau_i} \boldsymbol{f}(\boldsymbol{x}^*, \dots, \boldsymbol{x}^*) \boldsymbol{x}(t - \tau_i), \tag{1.26}$$

which follows from the Taylor expansion of $\boldsymbol{f}$.

1.2.3.1 Backward Extension

Let us for a moment return to (1.25). So far, we have discussed its solution for $t > \sigma$, but, sometimes, we may solve it backward in time for $t < \sigma$, i.e., perform a *backward extension* of the initial history ϕ. Importantly, it should be evident there is a fundamental asymmetry between the past and the future for DDEs (recall they are only a special case of RFDEs), which is non-existent in the case of ODEs. Indeed, as one can see from, for example, [22], there is no fundamental distinction between solving forward or backward in time.

Per [18] and [27], we say that $\boldsymbol{x} : [\sigma - \tau - \alpha, \sigma] \to \mathbb{R}^n$ for $\alpha > 0$ is a (backward) solution of the initial-value problem given by (1.25) if $\boldsymbol{x}$ is continuous, $\boldsymbol{x}$ satisfies the initial condition and

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{F}(t, \boldsymbol{x}_t), \ t \in [\sigma - \alpha, \sigma].$$

Because $\boldsymbol{x}_\sigma = \boldsymbol{\phi}$ must hold as well, due to the initial condition, we get that $\boldsymbol{x}(t) = \boldsymbol{\phi}(t - \sigma)$ must be continuously differentiable for $t \in [\sigma - \alpha, \sigma]$. As $\boldsymbol{\phi} \in \mathbb{X}_C$, we can equivalently say that $\boldsymbol{\phi}$ must be continuously differentiable on $[-\min\{\alpha, \tau\}, 0]$. This significantly constrains the space of all admissible elements in $\mathbb{X}_C$. Furthermore, because the equality must hold at $t = \sigma$, we get

$$\dot{\boldsymbol{\phi}}(0) = \dot{\boldsymbol{x}}(\sigma) = \boldsymbol{f}(\sigma, \boldsymbol{x}_\sigma) = \boldsymbol{f}(\sigma, \boldsymbol{\phi}),$$

where $\dot{\boldsymbol{\phi}}(0)$ denotes the left-hand derivative at 0. In total, the history function $\boldsymbol{\phi}$ must thus belong to a very limited submanifold $\mathscr{M}$ of $\mathbb{X}_C$ where

$$\mathscr{M} = \left\{ \boldsymbol{\psi} \in \mathbb{X}_C \cap C^1([-\min\{\tau, \alpha\}, 0], \mathbb{R}^n) : \dot{\boldsymbol{\psi}}(0) = \boldsymbol{f}(\sigma, \boldsymbol{\psi}) \right\}.$$

It is reasonable to assume that for a typical history function $\boldsymbol{\phi} \in \mathbb{X}_C$, these conditions will not be met, and therefore backward extension will not be possible. However, if $\boldsymbol{x} : [\sigma - \tau, \sigma + A)$, $A > 0$ is a (forward) solution of (1.25) and if $\sigma_1 \in (\sigma, \sigma + A)$, then the initial-value problem corresponding to this intermezzo initial condition $(\sigma_1, \boldsymbol{\psi})$, where $\boldsymbol{\psi} = \boldsymbol{x}_{\sigma_1}$, has a backward solution $\boldsymbol{x}$. It can be shown that many (but not all) solutions of autonomous systems (1.24) extend to all $t \in \mathbb{R}$ — steady-state and periodic solutions can be given as examples.

1.2.3.2 Stability

We have already discussed stability for dynamical systems in Definition 1.11. Stability for DDEs can be defined analogously, which will be summarized in the following definition (per [18] and [27]).

Definition 1.26 (Stability in Delay Differential Equations)**.** Consider (1.25) and suppose that $\boldsymbol{F}(t, \mathbf{0}) = \mathbf{0}$, $t \in \mathbb{R}$, is satisfied so that $\boldsymbol{x}(t) = \mathbf{0}$ is a solution. The solution $\boldsymbol{x} = \mathbf{0}$ is called

- *stable* if for any $\sigma \in \mathbb{R}$ and $\varepsilon > 0$, there exists $\delta = \delta(\sigma, \varepsilon) > 0$ such that $\boldsymbol{\phi} \in \mathbb{X}_C$ and $\|\boldsymbol{\phi}\| < \delta$ implies that $\|\boldsymbol{x}_t(\sigma, \boldsymbol{\phi})\| < \varepsilon$, $t \geq \sigma$;[10]
- *asymptotically stable* if it is stable and if there exists $b(\sigma) > 0$ such that whenever $\boldsymbol{\phi} \in \mathbb{X}_C$ and $\|\boldsymbol{\phi}\| < b(\sigma)$, then $\boldsymbol{x}(t, \sigma, \boldsymbol{\phi}) \to 0$ as $t \to \infty$;
- *unstable* if it is not stable.

For a non-zero solution $\boldsymbol{y}(t)$ of (1.25) defined on $t \in \mathbb{R}$, its stability properties mimic those of the zero solution of

$$\dot{\boldsymbol{z}}(t) = \boldsymbol{f}(t, \boldsymbol{z}_t + \boldsymbol{y}_t) - \boldsymbol{f}(t, \boldsymbol{y}_t). \tag{1.27}$$

Indeed, let $\boldsymbol{\xi}(t)$ be another solution of (1.25) and put $\boldsymbol{z}(t) = \boldsymbol{\xi}(t) - \boldsymbol{y}(t)$. Then $\boldsymbol{z}_t = \boldsymbol{\xi}_t - \boldsymbol{y}_t$ and thus (by using (1.25) on $\boldsymbol{\xi}_t$ or $\boldsymbol{y}_t$)

$$\dot{\boldsymbol{\xi}}(t) - \dot{\boldsymbol{y}}(t) = \boldsymbol{f}(t, \boldsymbol{\xi}_t - \boldsymbol{y}_t + \boldsymbol{y}_t) - \boldsymbol{f}(t, \boldsymbol{y}_t),$$

which yields that $\boldsymbol{z}$ is a solution to (1.27), which was to be shown.

Moreover, we shall also define a *solution operator* (or *solution map*) for autonomous RFDEs. This will become useful later when discussing the stability of equilibria in the case of DDEs from the perspective of their linearization, see Section 3.1.2.1.

Definition 1.27. Consider the autonomous DDE problem (1.24). The *solution operator* $\mathsf{S}(t, \sigma) : \mathbb{X}_C \to \mathbb{X}_C$ from time σ to t, such that $t \geq \sigma$, is defined as

$$\mathsf{S}(t, \sigma)\boldsymbol{\phi} := \boldsymbol{x}_t(\sigma, \boldsymbol{\phi}). \tag{1.28}$$

In the case of $\sigma = 0$, we abuse the notation to write $\mathsf{S}(t)$.

In other words, the solution operator $\mathsf{S}(t)$ maps the initial segment $\boldsymbol{\phi}$ onto the solution segment $\boldsymbol{x}_t$ at time t. Note that (1.28) is equivalent to (considering $\sigma = 0$)

$$(\mathsf{S}(t)\boldsymbol{\phi})(\theta) = \boldsymbol{x}(t + \theta), \ -\tau \leq \theta \leq 0, t \geq 0. \tag{1.29}$$

It can be shown that, see [27],

$$\begin{aligned} \mathsf{S}(0) &= \mathrm{id}, \\ \mathsf{S}(t)\mathsf{S}(s) &= \mathsf{S}(t + s), \ t, s \geq 0, \\ \mathsf{S}(t)\boldsymbol{\phi} &\text{ is continuous in } (t, \boldsymbol{\phi}), \end{aligned} \tag{1.30}$$

[10] For clarity, we use $\boldsymbol{x}(t, \sigma, \boldsymbol{\phi})$ to denote the solution of (1.25).

thus $\{S(t)\}$, $t \geq 0$, is a *strongly continuous semigroup of transformations* on a subset of $\mathbb{X}_C$ (see [27] or [31] for more information on the subject). Let us remark we assume t, s are allowed to range over an interval which may be dependent on $\boldsymbol{\phi} \in \mathbb{X}_C$.

1.2.4 Semidynamical Systems Induced by Delay Differential Equations

So far, we have only seen examples of *continuous-time* dynamical systems, which have inherently been symmetric in time, i.e., *invertible*. In other words, these systems did not distinguish between the future and the past. As an example, consider the following ODE and its induced continuous-time dynamical system

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}), \ \boldsymbol{x}(s) = \boldsymbol{x}_0.$$

When solving such ODE, it is entirely up to us to integrate forward in time, but backward integration is equally possible. On the other hand, as we have seen in Section 1.2.3.1, delay differential equations are not always solvable backward in time.

Definition 1.28. Semidynamical system is a triple[11] $(\widetilde{\mathbb{T}}, \mathbb{X}_C, \varphi)$, where

$$\widetilde{\mathbb{T}} = \{(t, s) \in \mathbb{T} \times \mathbb{T} \mid t \geq s\},$$

$\mathbb{X}_C$ is a metric space and $\varphi : \widetilde{\mathbb{T}} \times \mathbb{X}_C \to \mathbb{X}_C$ is continuous map, which satisfies the *deterministic* and *composition* properties from Definition 1.1.

The difference between Definition 1.1 and Definition 1.28 is rather subtle. It lies in the restriction to $\widetilde{\mathbb{T}}$, which permits only projection forward in time. Indeed, by using $\widetilde{\mathbb{T}} = \mathbb{T} \times \mathbb{T}$, we get the usual dynamical system. Note that an autonomous semidynamical system is defined analogously as in Definition 1.2. Also, basic concepts defined for dynamical systems, see Section 1.1.1, mostly extend to semidynamical systems (for example, we only consider ω-limit point and set).

Lemma 1.2. *Semidynamical system $(\widetilde{\mathbb{T}}, \mathbb{X}_C, \varphi)$ is autonomous if and only if whenever $\boldsymbol{x}(t)$ is a solution defined on subset $I \subseteq \mathbb{T}$ and $\tau \in \mathbb{T}_G$, then $\boldsymbol{x}(t + \tau)$ is a solution on $I - \tau$, where $I - \tau = \{t - \tau \mid t \in I\}$.*

Proof. Can be adapted from [18], page 63. □

As an example of an autonomous system, one can consider the delayed logistic equation, see [32],

$$\dot{n}(t) = a \cdot \left(1 - \frac{n(t - T)}{K}\right) n(t),$$

which assumes, among other facts, a constant growth rate a. Should the growth rate

[11] For semidynamical systems, we will use $\mathbb{X}_C$ instead of $\mathbb{X}$ to emphasize that $\boldsymbol{x} \in \mathbb{X}_C$ will typically be some (vector-valued) history function.

$a = a(t)$ be subject to change in time (for example by external influences), yielding

$$\dot{n}(t) = a(t) \cdot \left(1 - \frac{n(t-T)}{K}\right) n(t),$$

then it prescribes a **non**-autonomous semidynamical system. Analogously to the dynamical system case, the evolution operator φ of an autonomous semidynamical system is called a *semiflow*.

For compatibility with the source material, see [18], we will now consider an initial-value problem (1.25) for an RFDE, though for our purposes a DDE would suffice,

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{F}(t, \boldsymbol{x}_t), \ \boldsymbol{x}_\sigma = \boldsymbol{\phi},$$

where $\sigma \in \mathbb{R}$, $\boldsymbol{F}$ is continuous, and $\boldsymbol{\phi} \in \mathbb{X}_C$. Let us assume there is a unique solution $\boldsymbol{x}(t, \sigma, \boldsymbol{\phi})$ of (1.25) defined for $t \geq \sigma$. While this condition is hard to ensure in practice, it is nonetheless necessary for the definition of the induced semidynamical system. Denote by $\boldsymbol{x}_t(\sigma, \boldsymbol{\phi}) \in \mathbb{X}_C$ the state of the system at time t, defined, as usual, by

$$\boldsymbol{x}_t(\sigma, \boldsymbol{\phi})(\theta) = \boldsymbol{x}(t + \theta, \sigma, \boldsymbol{\phi}), \ -\tau \leq \theta \leq 0.$$

Proposition 1.1. *The map $\varphi(t, \sigma, \boldsymbol{\phi}) = \boldsymbol{x}_t(\sigma, \boldsymbol{\phi})$ defines a semidynamical system on $\mathbb{X}_C$ with $\mathbb{T} = [\sigma - \tau, \infty)$ and its corresponding $\widetilde{\mathbb{T}}$.*

Proof. See [18], page 65. □

1.3 Numerical Methods

As the title may suggest, computational aspects of ODEs, DDEs, and other mathematical problems will play a key role in this thesis. For completeness' sake, a brief overview of numerical integrators (also called *solvers*) for ODEs, DDEs, and stochastic differential equations (SDEs) will be given, as well as basic methods of one-dimensional optimization. Lastly, the Newton-Raphson method will be discussed.

1.3.1 Numerical integration of ODEs

Solving, or numerically integrating, ordinary differential equations can be seen as the backbone of this thesis. As such, we will introduce two approaches, the Euler method and the Runge-Kutta method. This subsection is mostly based on [33], [34], and [35].

Consider an ODE of the form

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{F}(t, \boldsymbol{x}(t)), \ \boldsymbol{x}(t_0) = \boldsymbol{x}_0, \tag{1.31}$$

which we want to integrate on the interval $[t_0, t_N]$. By Taylor expansion at t_0, we can write

$$\boldsymbol{x}(t) = \boldsymbol{x}(t_0) + \dot{\boldsymbol{x}}(t_0)(t - t_0) + O(t^2). \tag{1.32}$$

Setting $t = t_0 + \Delta t_1$ yields

$$\boldsymbol{x}(t_0 + \Delta t_1) \approx \boldsymbol{x}(t_0) + \dot{\boldsymbol{x}}(t_0)(t_0 + \Delta t_1 - t_0) = \boldsymbol{x}(t_0) + \boldsymbol{F}(t_0, \boldsymbol{x}(t_0))\Delta t_1.$$

Repeating this process on the entire interval $[t_0, t_N]$ gives us Algorithm 1 called the *explicit Euler method*. Let us note that often an equidistant stepsize is used, e.g., $\Delta t_k = \Delta t = \frac{t_N - t_0}{N}$.

Algorithm 1: Euler method (explicit)

Input: ODE in form (1.31), stepsizes $\{\Delta t_i\}_{i=1}^N$
Output: Sequence of $N + 1$ points $\{\boldsymbol{x}_i\}_{i=0}^N$
begin
  for $i = 1$ **to** N **do**
    $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_{i-1} + \boldsymbol{F}(t_{i-1}, \boldsymbol{x}_{i-1})\Delta t_i$
  end
end

Whilst the Euler method is simple, it does have its drawbacks mainly in its accuracy. For quantification of this issue, we shall define several notions of errors of a numerical integration method.

Definition 1.29. Let us consider an ODE problem of the form (1.31) and the solution $\boldsymbol{x}(t)$ to the initial-value problem. Assume $\boldsymbol{x}_k$ is an approximate solution at time t_k and that $\boldsymbol{x}_{k+1}$ is given by $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta t_{k+1} \cdot \mathrm{Inc}\,(t_k, \boldsymbol{x}(t_k), \Delta t_{k+1}, \boldsymbol{F})$, where Inc is the *increment* function[12], which propagates the solution from t_k to t_{k+1} and represents the chosen method. *Local truncation error* (LTE) is defined as

$$\mathrm{lte}_k = \boldsymbol{x}(t_{k+1}) - \boldsymbol{x}(t_k) - \Delta t_{k+1} \cdot \mathrm{Inc}\,(t_k, \boldsymbol{x}(t_k), \Delta t_{k+1}, \boldsymbol{F}) \tag{1.33}$$

and represents the error the increment function causes when performing a single step. Similarly, *global error* is defined as the cumulative LTE from the initial condition,

$$\mathrm{err}_k = \boldsymbol{x}(t_k) - \boldsymbol{x}_k. \tag{1.34}$$

> *In general, we seek such a method, which fulfills given accuracy while using as few evaluations of $\boldsymbol{F}$ as possible — the assumption is that the evaluation of $\boldsymbol{F}$ is so costly the remaining operations are almost negligible.*

By using (1.33) on Algorithm 1 to obtain LTE for the Euler method, we get (recalling the Taylor expansion (1.32))

$$\mathrm{lte}_k = \boldsymbol{x}(t_{k+1}) - \boldsymbol{x}(t_k) - \Delta t_{k+1} \cdot \boldsymbol{F}(t_k, \boldsymbol{x}(t_k)) \implies \mathrm{lte}_k = O(\Delta t_{k+1}^2).$$

[12] In general, the increment function Inc at the $k+1$-st step may depend on $t_0, \ldots, t_k$, $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_k$, and $\Delta t_1, \ldots, \Delta t_{k+1}$.

We say that a method is of order p if lte $= O(\Delta t^{p+1})$, i.e., the Euler method is of first order.

This can surely be improved by using more terms of the Taylor series expansion, see (1.32), but the necessity of computing the derivatives of $\boldsymbol{F}$ is a major drawback, often outweighing any potential benefits. Another possibility would be to allow the computation of the next value to be dependent on multiple previous values, leading to multistep methods, see Section 1.3.2. The last option — and the one we shall use — is to modify a one-step method to produce intermediate estimates of the solution at multiple different points (called *stages*), which are then combined into the final approximation of the solution at the next step — this way, we obtain so-called **Runge-Kutta** methods.

Algorithm 2: s**-stage explicit Runge-Kutta (ERK) method**

Input: ODE in form (1.31), stepsizes $\{\Delta t_i\}_{i=1}^N$, $s \in \mathbb{N}$ (the "number of stages"), real coefficients $a_{21}, a_{31}, a_{32}, \dots, a_{s1}, a_{s2}, \dots, a_{s,s-1}$, $b_1, \dots, b_s$, and $c_2, \dots, c_s$
Output: Sequence of $N+1$ points $\{\boldsymbol{x}_i\}_{i=0}^N$
begin
  for $i = 1$ **to** N **do**
    calculate

$$\begin{aligned}
\boldsymbol{k}_1 &\leftarrow \boldsymbol{F}(t_0, \boldsymbol{x}_{i-1}),\\
\boldsymbol{k}_2 &\leftarrow \boldsymbol{F}(t_0 + c_2\Delta t_i, \boldsymbol{x}_{i-1} + \Delta t_i a_{21}\boldsymbol{k}_1),\\
\boldsymbol{k}_3 &\leftarrow \boldsymbol{F}(t_0 + c_3\Delta t_i, \boldsymbol{x}_{i-1} + \Delta t_i(a_{31}\boldsymbol{k}_1 + a_{32}\boldsymbol{k}_2)),\\
&\vdots\\
\boldsymbol{k}_s &\leftarrow \boldsymbol{F}(t_0 + c_s\Delta t_i, \boldsymbol{x}_{i-1} + \Delta t_i(a_{s1}\boldsymbol{k}_1 + \cdots + a_{s,s-1}\boldsymbol{k}_{s-1}));
\end{aligned}$$

    set $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_{i-1} + \Delta t_i(b_1\boldsymbol{k}_1 + \cdots + b_s\boldsymbol{k}_s)$
  end
end

Usually, we constrain the coefficients to fulfill

$$c_m = \sum_{n=1}^{m-1} a_{mn}, \tag{1.35}$$

which represents the assumption that all points, where $\boldsymbol{F}$ is evaluated, are first-order approximations of the solution. It can be shown that (1.35) significantly simplifies the derivation of order conditions (1.36), but for low order, it is not necessary.

Our task is now to choose the coefficients in Algorithm 2 so that the order of the method is maximized. This corresponds to computing the Taylor series expansion for each $\boldsymbol{k}_j$ in the state variable of $\boldsymbol{F}$. By combining these results, we can compare them with the Taylor expansion of $\boldsymbol{x}_i$. Hence, we obtain the following conditions

(given for a 4-stage method), using $c_1 = 0$:

$$\begin{aligned} \sum_j b_j &= 1, & \sum_j b_j c_j &= \frac{1}{2}, \\ \sum_j b_j c_j^2 &= \frac{1}{3}, & \sum_{j,m} b_j a_{jm} c_m &= \frac{1}{6}, \\ \sum_j b_j c_j^3 &= \frac{1}{4}, & \sum_{j,m} b_j c_j a_{jm} c_m &= \frac{1}{8}, \\ \sum_{j,m} b_j a_{jm} c_m^2 &= \frac{1}{12}, & \sum_{j,m,n} b_j a_{jm} a_{mn} c_n &= \frac{1}{24}. \end{aligned} \tag{1.36}$$

In other words, if the coefficients of the 4-stage ERK method comply with conditions (1.36), the method is of fourth order, i.e., $\mathrm{lte}_i = O(\Delta t_i^5)$. Typically, *"the Runge-Kutta"* method is used:

$$\left.\begin{aligned} \boldsymbol{k}_1 &= \boldsymbol{F}(t_0, \boldsymbol{x}_0) \\ \boldsymbol{k}_2 &= \boldsymbol{F}\left(t_0 + \frac{1}{2}\Delta t, \boldsymbol{x}_0 + \frac{1}{2}\boldsymbol{k}_1 \Delta t\right) \\ \boldsymbol{k}_3 &= \boldsymbol{F}\left(t_0 + \frac{1}{2}\Delta t, \boldsymbol{x}_0 + \frac{1}{2}\boldsymbol{k}_2 \Delta t\right) \\ \boldsymbol{k}_4 &= \boldsymbol{F}(t_0 + \Delta t, \boldsymbol{x}_0 + \boldsymbol{k}_3 \Delta t) \end{aligned}\right\} \boldsymbol{x}_1 = \boldsymbol{x}_0 + \Delta t\left(\frac{1}{6}\boldsymbol{k}_1 + \frac{2}{6}\boldsymbol{k}_2 + \frac{2}{6}\boldsymbol{k}_3 + \frac{1}{6}\boldsymbol{k}_4\right). \tag{1.37}$$

Throughout this thesis, we will use solvers available in DifferentialEquations.jl, see [36], for Julia programming language, see [11], and ode45 (or ode23 for stiff ODEs) for MATLAB, see [37].

1.3.2 Numerical Integration of DDEs

As invested readers can surely recall, we have already discussed the method of steps in Section 1.2.2 as an analytical way to solve DDEs. There, we reformulated the problem of integrating a DDE forward in time as a sequence of ODE initial-value problems, which needed to be integrated on smaller intervals. A detailed treatment of the numerical integration of DDEs can be found in [30].

Consider the following DDE IVP[13],

$$\left.\begin{aligned} \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{x}(t-\tau)), \quad t_0 \le t \le t_f, \\ \boldsymbol{x}_\tau = \phi, \end{aligned}\right\} \tag{1.38}$$

where $\boldsymbol{x}(\cdot) \in \mathbb{R}^n$ and $\boldsymbol{f} : [t_0, t_f] \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$, and a set of mesh points $\mathscr{T} = \{t_0, t_1, \ldots, t_N = t_f\}$, such that either $t_n - \tau < t_0$ or $t_n - \tau \in \mathscr{T}$. Note that the presented methods can be extended to IVPs with multiple delays, e.g., (1.24).

[13] In general, methods discussed in this subsection allow a time-dependent delay τ, i.e., $\tau(t)$.

Once such a method is available, any ODE numerical integrator that operates strictly on the mesh $\mathscr{T}$ may be used — these are typically called *linear multistep* (LMS) methods. For instance, Algorithm 3 shows the explicit (forward) Euler method adapted for (1.38).

Algorithm 3: Euler method (explicit) for DDEs

Input: DDE in form (1.38), time discretization mesh $\mathscr{T} = \{t_0, \dots, t_N = t_f\}$
Output: Sequence of $N+1$ points $\{\boldsymbol{x}_i\}_{i=0}^{N}$
begin
  for $i = 1$ **to** N **do**
    $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_{i-1} + \boldsymbol{f}(t_{i-1}, \boldsymbol{x}_{i-1}, \boldsymbol{x}_j)\Delta t_i$
  end
end

However, this approach puts strict requirements on the mesh $\mathscr{T}$, which renders the method unusable in many cases. For example, for the IVP

$$\begin{aligned} \dot{x}(t) &= x(t/2), \quad 0 \le t \le 1, \\ x(0) &= 1, \end{aligned}$$

no finite mesh $\mathscr{T}$ can be found, as for any $t \in \mathscr{T}$ also $t/2$ must belong to the mesh (thus no initial stepsize exists). Nevertheless, in general, one can construct a suitable finite mesh $\mathscr{T}$ for any bounded interval $[t_0, t_f]$ if there is no vanishing delay point in $[t_0, t_f]$, see Definition 1.25 and Theorem 1.11. Bellen and Zennaro [30], see page 38, outline a possible strategy for doing so, along with its pitfalls. Also, note that this approach can be rather easily extended to *Adams-like* methods and even some *Runge-Kutta* formulae (such as the *Heun* method).

Another possible approach is to replace the strict constraints on the time mesh $\mathscr{T}$ by an interpolation method. Nonetheless, most often the discontinuity points belonging to $[t_0, t_f]$ are included in $\mathscr{T}$, as ODE solvers generally require smoothness of the solution (or at least such property is highly beneficial for accuracy).

Throughout the years, the theory of *continuous ODE methods*, see Definition 1.30, has been developed. Continuous ODE methods build on a very general class of k-step numerical ODE integrators for (1.31) of the form

$$\boldsymbol{x}_{i+1} = \alpha_{i,1}\boldsymbol{x}_n + \cdots + \alpha_{i,k}\boldsymbol{x}_{i-k+1} + \Delta t_{i+1}\,\mathrm{Inc}\,(\boldsymbol{x}_i, \dots, \boldsymbol{x}_{i-k+1}; \mathscr{T}_i, \boldsymbol{F})\,, \tag{1.39}$$

$i \ge k-1$, where $\mathscr{T}_i = \{t_{i-k+1}, \dots, t_i, t_{i+1}\}$ and where the *increment* function Inc is globally Lipschitz continuous with respect to $\boldsymbol{x}$ arguments ($\boldsymbol{F}$ is also assumed to be globally Lipschitz continuous in $\boldsymbol{x}$).

Definition 1.30 (Continuous ODE method [30])**.** We call *continuous extension*, or *interpolant*, of the ODE method (1.39) the piecewise polynomial function $\boldsymbol{\eta}(t)$ given by the restrictions on each interval $[t_i, t_{i+1}]$ of any interpolant based on values

computed in a possibly larger interval $[t_{i-k_i}, t_{i+l_i+1}]$, $k_i, l_i \geq 0$, of the form

$$\begin{aligned}\eta(t_i + \theta\Delta t_{i+1}) = \beta_{i,1}(\theta)\boldsymbol{x}_{i+l_i} + \cdots + \beta_{i,l_i+k_i+1}(\theta)\boldsymbol{x}_{i-k_i} \\ + \Delta t_{i+1}\,\mathrm{Inc}\left(\boldsymbol{x}_{i+l_i}, \ldots, \boldsymbol{x}_{i-k_i}; \theta, \boldsymbol{F}, \mathscr{T}_i'\right), \quad 0 \leq \theta \leq 1,\end{aligned} \tag{1.40}$$

where $\mathscr{T}_i' = \left\{t_{i-k_i}, \ldots, t_{i+j_i}, t_{i+j_i+1}\right\}$, satisfying the continuity conditions

$$\eta(t_i) = \boldsymbol{x}_i \quad \& \quad \eta(t_{i+1}) = \boldsymbol{x}_{i+1}. \tag{1.41}$$

If $l_i = k_i = 0$, then the interpolation procedure is based on values belonging to the sole interval $[t_i, t_{i+1}]$ and is referred to as *one-step interpolation*. Otherwise, the interpolation procedure is called *multistep interpolation*. In short, any ODE method (1.39) endowed with its continuous extension (1.40) will be called a *continuous ODE method*.

An arbitrary ODE solver can then be used as a *continuous ODE method* (with $j_i = 0$ for computational reasons). Note that even *explicit* ODE methods in general lead to *implicit* continuous ODE methods. For brevity, discussion on convergence and other numerical properties is omitted. In implementations, we will predominantly use DelayDiffEq.jl package by [38] in Julia or dde23 in MATLAB.

> Tip 3: Improvements to dde23
>
> Throughout the work on this thesis and related publications, we have noticed a significantly worse performance of the dde23 method compared to the equivalent DDE solver in Julia, see this Git repository. Humusoft, the distributor of MATLAB for the Czech Republic and Slovakia, determined that the issue lies in the code to find appropriate index i such that $t - \tau \in [t_i, t_{i+1}]$ for a given t (necessary for determining, e.g., $\mathscr{T}_i'$). The main inefficiency was caused by searching for i starting from the most distant past, see [39] for reference on the implementation. This problem is fixed in MATLAB version R2024b onward.

1.3.3 Numerical Integration of SDEs

In Section 1.1.5, we have presented a framework for dealing with stochastic dynamical systems. These very often arise from noisy measurements of the underlying system. Using the discussed concepts, we can represent this as a discrete-time approximation of an Itô process. In particular, we will use the **Euler-Maruyama** approximation. Consider an Itô process $\mathrm{X} = \{\mathrm{X}(t) \mid t_0 \leq t \leq T\}$ satisfying the scalar SDE

$$\mathrm{d}\mathrm{X}(t) = a(t, \mathrm{X}(t))\,\mathrm{d}t + b(t, \mathrm{X}(t))\,\mathrm{d}\mathrm{W}(t), \ \mathrm{X}(t_0) = \mathrm{X}_0 \tag{1.42}$$

on $t_0 \leq t \leq T$.

For a given discretization $t_0 < t_1 < \cdots < t_n = T$ of the time interval $[t_0, T]$, an Euler-Maruyama approximation is a continuous-time stochastic process $\mathrm{Y} =$

$\{\mathrm{Y}(t) \mid t_0 \le t \le T\}$ satisfying the following iterative scheme (using $\mathrm{Y}(t_n) = \mathrm{Y}_n$)

$$\mathrm{Y}_{n+1} = \mathrm{Y}_n + a(t_n, \mathrm{Y}_n)(t_{n+1} - t_n) + b(t_n, \mathrm{Y}_n)(\mathrm{W}(t_{n+1}) - \mathrm{W}(t_n)) \tag{1.43}$$

for $n = 0, 1, \dots, N-1$ with initial value $\mathrm{Y}_0 = \mathrm{X}_0$. For conciseness, we shall denote $\Delta t_n = t_{n+1} - t_n$ and call $\delta = \max_n \Delta t_n$ the *maximum time step*. Most often, we shall use equidistant discretization, i.e., $t_n = t_0 + n\delta$, and in such case we will use $\Delta t = \delta \equiv \Delta t_n = (T - t_0)/N$. Similarly, let $\Delta \mathrm{W}_n := \mathrm{W}(t_{n+1}) - \mathrm{W}(t_n)$ be the random independent increments of the Wiener process, see Definition 1.24, corresponding to the discretization. From the definition immediately follows that

$$\Delta \mathrm{W}_n \sim \mathscr{N}(0, \Delta t_n) \quad \& \quad \Delta \mathrm{W} \sim \mathscr{N}(0, \Delta t)\,.$$

Thus, for equidistant discretization, (1.43) becomes

$$\mathrm{Y}_{n+1} = \mathrm{Y}_n + a(t_n, \mathrm{Y}_n)\Delta t + b(t_n, \mathrm{Y}_n)\Delta \mathrm{W}, \tag{1.44}$$

and as such $\{\mathrm{Y}_n\}_n$ can be interpreted as a Markov chain. This can be easily extended to a vector-valued SDE,

$$\mathrm{d}\mathbf{X}(t) = \boldsymbol{a}(t, \mathbf{X}(t))\,\mathrm{d}t + \boldsymbol{B}(t, \mathbf{X}(t))\,\mathrm{d}\mathbf{W}(t),\ \mathbf{X}(t_0) = \mathbf{X}_0, \tag{1.45}$$

where, assuming $\mathbf{X}(t, \omega) \in \mathbb{R}^n$, we have $\boldsymbol{a}(t, \mathbf{X}(t, \omega)) \in \mathbb{R}^n$, $\boldsymbol{B}(t, \mathbf{X}(t, \omega)) \in \mathbb{R}^{n \times n}$, and $\mathrm{d}\mathbf{W}(t)$ is a differential corresponding to a vector of Wiener processes with prescribed correlation (and hence $\Delta \mathbf{W}_n \in \mathbb{R}^{n \times n}$ has a matrix structure relating to the underlying correlation). Finally, this is captured in Algorithm 4.

Algorithm 4: Euler-Maruyama method

Input: SDE in form (1.45), time discretization $\{\Delta t_i\}_{i=1}^N$
Output: Discretized random process $\mathbf{Y}$ to $\{t_i\}_{i=0}^N$
begin
  set $\mathbf{Y}_0 \leftarrow \mathbf{X}_0$ by (1.45)
  for $i = 0$ **to** $N - 1$ **do**
    $\mathbf{Y}_{n+1} \leftarrow \mathbf{Y}_n + \boldsymbol{a}(t_n, \mathbf{Y}_n)\Delta t_n + \boldsymbol{B}(t_n, \mathbf{Y}_n)\Delta \mathbf{W}_n$
  end
end

Let us note that for implementation, we use the `EM` method from DifferentialEquations.jl by [36] for Julia and a custom implementation for MATLAB, see [37]. Furthermore, the relevant theory of stochastic functional differential equations (SFDEs) will not be covered in this document to ensure a focused and concise scope. To an interested reader, we recommend [40] for a general overview and [41] for more details on linear and affine SFDEs. As an example of actual use, in [6], we have employed the Euler-Maruyama method to compare stochastic and deterministic simulations of the Pinsky-Rinzel model, see [42] and [43], for specific values of parameters, deduced from preceding bifurcation analysis, producing notable signal patterns comparable with those measured experimentally.

1.3.4 One-Dimensional Optimization

Optimization is nigh ubiquitous in both nature and mathematics. For us, it will prove most fruitful to study optimization in one dimension, where the situation is, fortunately, the easiest. Most of the material is taken from [44] and [45].

The methods we will present can be collectively called *bracketing*, as they rely on iteratively shrinking an interval inside which a local minimum lies. Furthermore, a single-minimum requirement shall be placed on functions to be optimized. For functions on $\mathbb{R}$, this translates to so-called *unimodality*.

Definition 1.31 (Unimodal function)**.** Let $I \subset \mathbb{R}$ be an interval and $f : I \to \mathbb{R}$ a function. We say f is **unimodal** if there exists $x^* \in I$ such that

- $f(x_1) > f(x_2)$ for arbitrary $x_1, x_2 \in I$ satisfying $x_1 < x_2 < x^*$;
- $f(x_1) < f(x_2)$ for arbitrary $x_1, x_2 \in I$ satisfying $x^* < x_1 < x_2$.

> **i** Note 5: Unimodality
>
> As a careful reader might notice, a unimodal function is necessarily *monotone decreasing* on $(-\infty, x^*) \cap I$ and *monotone increasing* on $I \cap (x^*, \infty)$. Moreover, unimodality implies only *quasi-convexity*[14] and, on the other hand, only *strict convexity* implies unimodality of a function on $f : I \to \mathbb{R}$.

Lemma 1.3. *Let $f : I \to \mathbb{R}$ be unimodal on I and consider $x_1, x_2 \in I$ such that $x_1 < x_2$, then*

- $f(x_1) \leq f(x_2) \implies x^* \leq x_2$;
- $f(x_1) \geq f(x_2) \implies x_1 \leq x^*$.

Proof. Follows directly from Definition 1.31. □

Consider now the following optimization problem

$$\min_{x \in I} f(x), \quad \text{resp.} \quad \operatorname*{argmin}_{x \in I} f(x), \tag{1.46}$$

for an **unimodal** function $f : I \to \mathbb{R}$, $I = [a, b]$. Furthermore, let N denote the *allowed number of evaluations* of f and assume the accuracy of presented methods is $|\bar{x} - x^*|$, where x^* is the exact solution of (1.46) and $\bar{x}$ its approximation found by the used method. Using Lemma 1.3, one can easily find the initial bracket to be successively shrunk if it is not provided in the definition of the optimization problem (1.46).

[14] A function $f : S \to \mathbb{R}$, where S is a real vector space, is *convex* on S if for all $\boldsymbol{x}, \boldsymbol{y} \in S$ and $\lambda \in [0, 1]$ holds $f(\lambda \boldsymbol{x} + (1 - \lambda)) \leq \lambda f(\boldsymbol{x}) + (1 - \lambda) f(\boldsymbol{y})$ and *strictly convex* if the inequality is strict. Furthermore, we call it *quasi-convex* on S if for any $\boldsymbol{x}, \boldsymbol{y} \in S$ and $\lambda \in [0, 1]$ we have $f(\lambda \boldsymbol{x} + (1 - \lambda)\boldsymbol{y}) \leq \min\{f(\boldsymbol{x}), f(\boldsymbol{y})\}$.

1.3.4.1 Bisection Method

Consider now $N = 2k$, $k \in \mathbb{N}$, choose $\delta \in \left(0, \frac{b-a}{2}\right]$ sufficiently small and let $a_0 := a, b_0 := b$ set the initial bracket $[a_0, b_0]$. We then define

$$x_i^- := \frac{a_{i-1} + b_{i-1}}{2} - \delta \;\&\; x_i^+ := \frac{a_{i-1} + b_{i-1}}{2} + \delta \tag{1.47}$$

for $i = 1, \dots, k$. Now we can examine in which "half" of the interval (including the offset δ) the minimum lies, i.e., we evaluate $f(x_i^-), f(x_i^+)$ and adjust the bracket accordingly (by Lemma 1.3)

- $f(x_i^-) < f(x_i^+) \implies [a_i, b_i] = [a_{i-1}, x_i^+]$;
- $f(x_i^-) > f(x_i^+) \implies [a_i, b_i] = [x_i^-, b_{i-1}]$.

The center $\bar{x}_k$ of the k-th bracket is assumed to approximate the exact solution x^* with accuracy $l_k/2$, where

$$l_k = \frac{b-a}{2^k} + \frac{(2^k - 1)\delta}{2^{k-1}}$$

denotes the length of the k-th bracket. It can be seen that $l_k \to 2\delta$ for $k \to \infty$.

Algorithm 5: Bisection method

Input: Optimization problem (1.46), initial interval $I = [a, b]$, $\delta \in \left(0, \frac{b-a}{2}\right]$, number of steps $k \in \mathbb{N}$
Output: Approximation of minimum $\bar{x}_k$
begin
  set $a_0 \leftarrow a, b_0 \leftarrow b$
  for $i = 1$ **to** k **do**
    calculate $x_i^- \leftarrow \frac{a_{i-1}+b_{i-1}}{2} - \delta$ & $x_i^+ \leftarrow \frac{a_{i-1}+b_{i-1}}{2} + \delta$
    if $f(x_i^-) < f(x_i^+)$ **do**
      set $[a_i, b_i] \leftarrow [a_{i-1}, x_i^+]$
    else if $f(x_i^-) > f(x_i^+)$ **do**
      set $[a_i, b_i] \leftarrow [x_i^-, b_{i-1}]$
    else do
      set[15] $[a_i, b_i] \leftarrow [x_i^-, b_{i-1}]$
    end
    set $\bar{x}_i \leftarrow \frac{a_i+b_i}{2}$
  end
end

1.3.4.2 Golden Section Method

The bisection method we have presented above has one significant flaw. Every iteration, i.e., every bracket improvement, requires two separate evaluations which

[15] In the case of equality, both options for a new bracket are equally valid.

are, in general, never used again. Thus, the Golden section method was constructed, requiring only a single evaluation.

Tip 4: Golden ratio

Before delving into the technical details, let us note in this subsection $\tau \approx 1.618$ denotes the positive solution of

$$\tau^2 - \tau - 1 = 0, \tag{1.48}$$

i.e., $\frac{1}{\tau} \approx 0.618$, and is generally referred to as the *golden ratio*. Equivalently, it is also given as

$$\frac{a+b}{a} = \frac{a}{b} = \tau \tag{1.49}$$

for $a > b > 0$ which satisfy the above equality.

Consider a unimodal function $f : I \to \mathbb{R}$ on the interval $I = [a, b]$ subject to an optimization problem (1.46) and evaluation-count restriction $N \geq 2$ (or desired accuracy ε). Let us relax the bisection method such that

$$x_i^- = a_{i-1} + \delta_-(b_{i-1} - a_{i-1}),\ x_i^+ = a_{i-1} + \delta_+(b_{i-1} - a_{i-1}) \tag{1.50}$$

and we shall attempt to find a values of $\delta_-, \delta_+ \in (0, 1)$ such that only 1 iteration is needed. Assume $f(x_i^-) < f(x_i^+)$, then (by Lemma 1.3 as in Algorithm 5)

$$[a_i, b_i] = [a_{i-1}, x_i^+] = [a_{i-1}, a_{i-1} + \delta_+(b_{i-1} - a_{i-1})]$$

and

$$x_{i+1}^- = a_i + \delta_-(b_i - a_i)\ x_{i+1}^+ = a_i + \delta_+(b_i - a_i).$$

If we want to reuse the $f(x_i^-)$ evaluation, then necessarily

$$\begin{aligned} x_{i+1}^+ &= x_i^- \\ a_i + \delta_+(b_i - a_i) &= a_{i-1} + \delta_-(b_{i-1} - a_{i-1}) \\ a_{i-1} + \delta_+(a_{i-1} + \delta_+(b_{i-1} - a_{i-1}) - a_{i-1}) &= a_{i-1} + \delta_-(b_{i-1} - a_{i-1}) \\ \delta_+^2(b_{i-1} - a_{i-1}) &= \delta_-(b_{i-1} - a_{i-1}) \\ \implies \delta_+^2 &= \delta_-. \end{aligned}$$

Now we shall focus on $f(x_i^-) \geq f(x_i^+)$, denoting $\delta = \delta_+$, which by the same argument leads to

$$[a_i, b_i] = [x_i^-, b_{i-1}] = \left[a_{i-1} + \delta^2(b_{i-1} - a_{i-1}), b_{i-1}\right],$$

which together with the analogous condition $x_{i+1}^- = x_i^+$ yields (using $l_i = b_i - a_i$)

$$\begin{aligned}
x_{i+1}^- &= x_i^+ \\
a_i + \delta^2(b_i - a_i) &= a_{i-1} + \delta(b_{i-1} - a_{i-1}) \\
a_{i-1} + \delta^2(b_{i-1} - a_{i-1}) + \delta^2\left(b_{i-1} - a_{i-1} - \delta^2(b_{i-1} - a_{i-1})\right) &= a_{i-1} + \delta(b_{i-1} - a_{i-1}) \\
\delta^2 l_{i-1} + \delta^2 l_{i-1} - \delta^4 l_{i-1} &= \delta l_{i-1} \\
\delta(\delta^3 - 2\delta + 1) &= 0 \\
\delta(\delta - 1)(\delta^2 + \delta - 1) &= 0.
\end{aligned}$$

Recall we defined $\delta_-, \delta_+ \in (0,1)$, thus $\delta \in (0,1)$ ruling out $\delta = 0$ and $\delta = 1$ roots. What remains is $\delta^2 + \delta - 1 = 0$, which has the solution $\delta = \frac{\sqrt{5}-1}{2}$.

Moreover, notice that

$$\frac{x_{i+1}^- - a_i}{x_{i+1}^+ - a_i} = \frac{a_i + \delta^2 l_i - a_i}{a_i + \delta l_i - a_i} = \delta = \frac{a_i + \delta l_i - a_i}{l_i} = \frac{x_{i+1}^+ - a_i}{l_i},$$

i.e., we split the interval $[a_i, b_i]$ such that the ratio between the whole $[a_i, b_i]$ and the bigger part $[a_i, x_{i+1}^+]$ is the same as between the smaller part $[a_i, x_{i+1}^-]$ and the bigger part. Rephrasing this ratio equality as (using $\delta^2 = 1 - \delta$)

$$\begin{aligned}
\frac{1}{\delta} = \frac{x_{i+1}^+ - a_i}{x_{i+1}^- - a_i} &= \frac{l_i}{x_{i+1}^+ - a_i} \\
&= \frac{x_{i-1}^+ - a_i + b_i - x_{i-1}^+}{x_{i+1}^+ - a_i} \\
&= \frac{x_{i-1}^+ - a_i + b_i - a_i - \delta l_i}{x_{i+1}^+ - a_i} \\
&= \frac{x_{i-1}^+ - a_i + l_i - \delta l_i}{x_{i+1}^+ - a_i} \\
&= \frac{x_{i-1}^+ - a_i + (1-\delta) l_i}{x_{i+1}^+ - a_i} \\
&= \frac{x_{i-1}^+ - a_i + \delta^2 l_i}{x_{i+1}^+ - a_i} \\
&= \frac{x_{i-1}^+ - a_i + a_i + \delta^2 l_i - a_i}{x_{i+1}^+ - a_i} \\
&= \frac{x_{i-1}^+ - a_i + x_{i-1}^- - a_i}{x_{i+1}^+ - a_i}
\end{aligned}$$

gives us the relation (1.49) defining the golden ratio, i.e., $\tau = \frac{1}{\delta}$. The method bears its name due to this fact. Lastly, see Algorithm 6 for implementation.

1.3.4.3 Fibonacci method

Let us now approach the problem from a different direction. Given the minimization problem (1.46) and maximum evaluations N, we want to shrink the initial bracket as much as possible.

Suppose only 2 evaluations are allowed. Then one possibility is to query f at thirds of the initial interval. The final bracket will have a two-thirds length of the initial interval, but it can be improved. Consider the distance between the evaluations ε, then the final bracket is guaranteed to shrink to half of the original interval (i.e., shrinkage by a factor of 2) as $\varepsilon \to 0$ in the limit sense, see Figure 1.4.

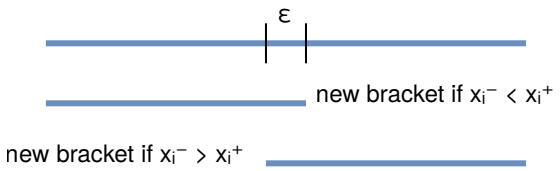


Figure 1.4: Shrinkage by ε-distance between the only two allowed evaluations.

Suppose $N = 3$ evaluations are now permitted. Then the best course of action is to divide the initial into thirds, which results in the next bracket having $\frac{2}{3}$-length. If we perform the last f-query in the ε-distance from the interior evaluation of the previous round (one is guaranteed to lie inside the next-iteration bracket), then by $\varepsilon \to 0$, in the limit sense, we get a shrinkage by a factor of 3, see Figure 1.5.

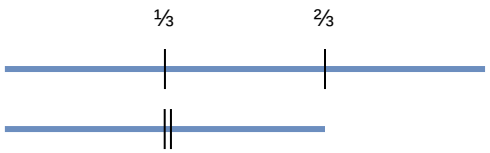


Figure 1.5: Sketch of the Fibonacci method bracket shrinkage when $N = 3$.

In general, this leads to evaluation given by the Fibonacci numbers

$$F_n = \begin{cases} 1, & n \leq 2, \\ F_{n-1} + F_{n_2}, & n > 2, \end{cases}$$

which prescribe the queries at

$$x_i^- = a_{i-1} + \frac{F_{N-i-1}}{F_{N-i+1}} l_{i-1}, \quad x_i^+ = a_{i-1} + \frac{F_{N-i}}{F_{N-i+1}} l_{i-1}. \tag{1.51}$$

For more details on the derivation, see [45]. Moreover, the Fibonacci series can be written explicitly using *Binet's formula*:

$$F_n = \frac{\tau^n - (1-\tau)^n}{\sqrt{5}}, \tag{1.52}$$

where τ is the *golden ratio*, see Tip 4, from which can be shown

$$\frac{1}{\rho(i)} = \frac{F_i}{F_{i-1}} = \tau \frac{1 - s^i}{1 - s^{i-1}}, \tag{1.53}$$

where $s = \frac{1-\sqrt{5}}{1+\sqrt{5}}$. The algorithm of the Fibonacci method, heavily based on implementation from [44], is provided in Algorithm 7.

1.3.5 Newton's Method

Last, but not least, we shall discuss Newton's, or Newton-Raphson, method for finding the roots of a given equation (or a system thereof). The treatment provided here follows [46]. Suppose we are given a system of equations

$$\begin{aligned} f_1(x_1, \dots, x_n) &= 0 \\ f_2(x_1, \dots, x_n) &= 0 \\ &\vdots \\ f_n(x_1, \dots, x_n) &= 0 \end{aligned} \iff \boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{0}. \tag{1.54}$$

Let $\boldsymbol{x}^i \in \mathbb{R}^n$ be known and fixed, for example, a previous iteration, then (1.54) can be approximated using Taylor's series in the neighborhood of $\boldsymbol{x}^i$. Corresponding first-order Taylor expansion is

$$f_k(\boldsymbol{x}) \approx f_k(\boldsymbol{x}^i) + \sum_{j=1}^{n} \frac{\partial f_k(\boldsymbol{x}^i)}{\partial x_j}(x_j - x_j^i).$$

Using the vector notation, we can collectively approximate (1.54) as

$$\boldsymbol{F}(\boldsymbol{x}^i) + \mathrm{J}_i(\boldsymbol{x} - \boldsymbol{x}^i) = \boldsymbol{0}, \tag{1.55}$$

where $\mathrm{J}_i = \nabla_{\boldsymbol{x}} \boldsymbol{F}(\boldsymbol{x}^i)$, i.e., $(\mathrm{J}_i)_{k,j} = \frac{\partial f_k(\boldsymbol{x}^i)}{\partial x_j}$. Certainly a solution to (1.55) approximates the true solution to (1.54). Then, by sequentially approximating the true solution from points getting closer, the approximations also get more accurate. In other words, we can write the following iterative process to find the true solution (to an arbitrary accuracy)

$$\boldsymbol{x}^{i+1} = \boldsymbol{x}^i - \mathrm{J}_i^{-1} \boldsymbol{F}(\boldsymbol{x}^i). \tag{1.56}$$

While this formula for Newton's method is very common, we shall also present the *residual* form

$$\mathrm{J}_i \cdot \Delta \boldsymbol{x}^{i+1} = \boldsymbol{R}(\boldsymbol{x}^i), \tag{1.57}$$

where $\boldsymbol{R}(\boldsymbol{x}) = -\boldsymbol{F}(\boldsymbol{x})$ is the *residual*, and $\Delta \boldsymbol{x}^{i+1} = \boldsymbol{x}^{i+1} - \boldsymbol{x}^i$ is the *displacement*, which play the role of unknowns in (1.57). As a careful reader might notice, instead of requiring an inverse of the Jacobian matrix J_i, we have translated it into a linear system of equations. If J_i^{-1} exists and is numerically stable[16], then we can use it to solve (1.57). On the other hand, if J_i is badly conditioned (or singular), so J_i^{-1} cannot be reliably found, then one can use iterative methods to find an approximate solution. Assuming $\Delta \boldsymbol{x}^{i+1}$ is the solution to (1.57), next iteration for (1.56) is computed as

$$\boldsymbol{x}^{i+1} = \boldsymbol{x}^i + \Delta \boldsymbol{x}^{i+1}.$$

[16] In such case we can calculate it straight or implicitly via so-called *left division*, which in Julia relies on pivoted QR decomposition.

There are many possible stopping conditions for this iterative Newton's method, but often one can see

$$\left\|\boldsymbol{x}^i - \boldsymbol{x}^{i-1}\right\| \leq \varepsilon_{\text{err}} \;\&\; \left\|\boldsymbol{R}(\boldsymbol{x}^i)\right\| \leq \varepsilon_{\text{res}},$$

where ε_{err} is the given tolerance for the solution error and ε_{res} for the residual.

> **Note 6: Modifications to Newton's method**
>
> Here, we have only discussed the standard Newton's method, but there exist several modifications including the Newton-chord method, Newton-Broyden method, and various quasi-Newton methods.

1.4 Neuronal Dynamics

Neuron is the fundamental building block of the neural system of many living organisms on Earth. As such, how we understand it directly impacts a significant portion of our (perception of the) world. Naturally, mathematical models and concepts from computational neuroscience play a key role in this understanding. Let us note this section is based on [47] and [48], with relevant publications referenced when needed.

1.4.1 Physiology of a Neuron

Every cell has a *cell membrane* on its boundary, consisting of a phospholipid bilayer and irregularly spaced ion channels, which separate the internal structures of the cell from the surrounding external environment. However, from our point of view, its most important feature is the *selective permeability*, which allows it to permit the passage of certain chemicals while restricting others. As a simplification, we may assume it lets the potassium cations K^+ go through freely but heavily obstructs the passage of sodium cations Na^+. Thanks to this property, a *membrane potential* emerges for most cells. The potential is induced by the lower concentration of positively charged particles (typically potassium cations K^+ diffuse outward) inside the cell than outside. By the interplay of diffusion and Coulomb's law acting on particles in and outside of the cell, the *membrane potential* stabilizes at -70 mV (typically called the *resting potential*).

Certain cells, among which are neurons, possess the ability to control the permeability of their membrane. A local change of the permeability is called the *action potential* if the change triggers a characteristic sequence of events resulting in the return to the resting state. On the contrary, a small change is generally compensated by slow regulating mechanisms.

> **Note 7: All-or-nothing law**
>
> The threshold for activation of the action potential and its constant characteristic behavior induce the so-called *all-or-nothing law*, which says that a neuron sends a signal if and only if it receives a strong enough impulse and that the signal is *de facto* the same every time, see [49].

During the action potential, the corresponding part of the cell membrane opens its ion channels permeable to Na^+. Driven by the electrochemical gradient, these ions enter the cell, making the membrane potential more positive, a process known as depolarization. This change triggers the opening of voltage-gated potassium (K^+) channels, allowing K^+ ions to flow out of the cell, thereby restoring the membrane potential in a process called repolarization. This, in turn, *hyperpolarizes* the cell membrane (the membrane potential is even lower than the resting value at this point). The entire process is then slowly reversed as sodium cations exit the cell and are replaced by potassium cations. Note that the overpopulation of Na^+ inside the cell restricts its ability to undergo the *action potential* until it sufficiently lowers, limiting the possible spiking frequency of an individual neuron.

Although the *action potential* is fundamentally a local behavior of the neuronal cell membrane, it triggers a chain reaction around the initial point of excitation, which causes neighboring parts of the membrane to undergo the same process.

1.4.2 Conductance-based Models

As can be seen from the previous subsection, a key quantity for a mathematical model of a single isolated neuron is likely to be the *membrane potential*, ideally affected by variables describing the permeability of various ion channels. Indeed, conductance-based models, see [50], are established on an equivalent circuit representation of a cell membrane, introduced by Hodgkin and Huxley [51]. Here, ion channels of an excitable cell are modeled as conductances and its lipid bilayer by a capacitor. In its simplest form, a conductance-based model represents a neuron by a single compartment with homogeneous potential, disregarding ion movements between subcellular compartments, and captures only ion movements between the inside and outside of the cell.

The original model by Hodgkin and Huxley [51], constructed based on measurements of the giant axon of a squid, reads as

$$\begin{aligned} I_{\text{ext}} &= C\dot{V} + g_K n^4 (V - V_K) + g_{Na} m^3 h (V - V_{Na}) + g_L (V - V_L), \\ \dot{n} &= \alpha_n(V)(1-n) - \beta_n(V) n, \\ \dot{m} &= \alpha_m(V)(1-m) - \beta_m(V) m, \\ \dot{h} &= \alpha_h(V)(1-h) - \beta_h(V) h, \end{aligned} \tag{1.58}$$

where $I_c = C_m \dot{V}$ describes the current flowing into the lipid bilayer capacitance (C is the capacity of the membrane) and $I_{\text{ion}_i} = g_i(V - V_i)$ current through a given

ion channel[17] i as the product of its conductance g_i and the corresponding driving potential $V - V_i$. Here, V_i denotes so-called *reverse potentials*, which capture associated resting potentials. Lastly, m, n, h model the probabilities of respective ion channels being open. The exact formulas for functions α_i, β_i with $i = m, n, h$, and parameter values can be found in the original article.

The several conductance-based models, founded on the Hodgkin–Huxley formalism, appeared after the breakthrough publication of Hodgkin and Huxley. Among the most influential is the Morris–Lecar model, see [52], which they formulated after studying the muscle fiber of a barnacle. Nevertheless, nowadays it is used to model the dynamics of human neurons. It reduces the 4-dimensional Hodgkin–Huxley model into 2 dimensions, see [53],

$$\begin{aligned} C\dot{V} &= -g_{Ca} m_\infty(V)(V - V_{Ca}) - g_K w(V - V_K) - g_L(V - V_L) + I_{\text{ext}}, \\ \dot{w} &= \frac{w_\infty(V) - w}{T_w(V)}, \end{aligned} \tag{1.59}$$

where w is the *"recovery variable"* for the K^+ channel prescribing the probability of the corresponding ion channel being open. Moreover, the open-state probability functions m_∞, w_∞ are derived from assumptions about the *steady-state* and its distribution of open and closed ion channels. Finally, T_w is used to set a correct timescale for w.

The last model we shall consider is the *interneuron* model proposed by White et al., see [54]. It was used to describe the synchronization of oscillations and the loss thereof in a network of inhibitory neurons. Nonetheless, it allows us to fast-spiking neuron cells. The governing equations read as, see [4],

$$\begin{aligned} C\dot{V} &= I_{\text{ext}} - \overbrace{g_L(V - V_L)}^{I_L} - \overbrace{g_{Na} m^3 h(V - V_{Na})}^{I_{Na}} - \overbrace{g_K n^4 (V - V_K)}^{I_K}, \\ \dot{h} &= \frac{h_\infty(V) - h}{T_h(V)}, \\ \dot{n} &= \frac{n_\infty(V) - n}{T_n(V)}, \end{aligned} \tag{1.60}$$

where C again indicates the capacitance of the neuronal membrane, I_{ext} is the external stimulus and I_L, I_{Na}, I_K incorporate the relationship between the membrane voltage V (typically in mV) and currents through respective channels. Analogously, g_i and V_i act as the maximum conductance and reverse potential, respectively, for an ion channel i. Researchers noticed that the recovery variable m in (1.58) almost immediately converges to its corresponding steady state, thus using the *slow-fast* dimensionality reduction method, they replaced it with its asymptotic

[17] Note that L denotes the abstract *leak* ion channel, which aggregates all the smaller, less-significant ion channels into one.

value (so-called *quasi-equilibrium*)[18],

$$m = m_\infty(V) = \frac{1}{1 + \exp\left(-0.08(V + 26)\right)}.$$

The dynamics of activation variables h, n are further related to their corresponding steady-state distributions and correctly scaled in time, i.e.,

$$h_\infty(V) = \frac{1}{1 + \exp\left(0.13(V + 38)\right)}, \qquad T_h(V) = \frac{0.6}{1 + \exp\left(-0.12(V + 67)\right)},$$
$$n_\infty(V) = \frac{1}{1 + \exp\left(-0.045(V + 10)\right)}, \quad T_n(V) = 0.5 + \frac{2}{1 + \exp\left(0.045(V - 50)\right)}.$$

Our choice of parameters is based on [5],

$$\begin{aligned} C = 1, \quad g_L = 0.1, \quad g_{Na} = 30, \quad g_K = 20, \\ I_{\text{ext}} = 24, \quad V_L = -60, \quad V_{Na} = 45, \quad V_K = -80. \end{aligned} \tag{1.61}$$

1.4.3 Couplings

So far, we have only studied single neurons, but in nature, a far more common occurrence is a (large) coupled network of neurons. However, for simplicity, we shall devote our attention only to a case of 2 weakly coupled neurons. For more information, see [55].

In general, couplings can be either chemical or electrical. In the case of electrical coupling, there is a direct connection from one neuron to another, facilitated via a channel or a gap junction. Although more common is the (slower) chemical synapse (also called the *synaptic coupling*), per [54] and [56], we will consider solely the electrical coupling (our goal is to study high-frequency oscillations, which would be inhibited by the use of slow coupling).

We model the gap-junctional coupling effect as external currents directly proportional to the differences between membrane potentials for both coupled neurons. In particular, it yields

$$\begin{aligned} C_i \dot{V}_i(t) &= I_{\text{ext}} - I_i^{\text{channels}}(t) - \lambda(V_i(t) - V_{\hat{\imath}}(t - \tau)), \\ &\vdots \end{aligned} \tag{1.62}$$

where V_i marks the membrane potential of the i-th neuron, C_i capacity of its membrane, λ is the coupling strength, $\hat{\imath}$ denotes the "opposite" coupled neuron, and $\tau > 0$ a possible delay in the information introduced by the coupling. As such,

[18] An attentive reader may notice that a deviation from the quasi-equilibrium is often used for modelling of recovery variables, see, for instance, (1.59) or (1.60).

the system of two λ-weakly coupled interneurons, see (1.60), becomes

$$\left.\begin{aligned} C_1\dot{V}_1(t) &= I_{\text{ext}} - g_L(V_1(t)-V_L) - g_{Na}m_\infty^3(V_1(t))h_1(t)(V_1(t)-V_{Na}) \\ &\quad - g_K n_1^4(t)(V_1(t)-V_K) - \lambda(V_1(t)-V_2(t-\tau)), \\ \dot{h}_1(t) &= \frac{h_\infty(V_1(t))-h_1(t)}{T_h(V_1(t))}, \\ \dot{n}_1(t) &= \frac{n_\infty(V_1(t))-n_1(t)}{T_n(V_1(t))}, \\ C_2\dot{V}_2(t) &= I_{\text{ext}} - g_L(V_2(t)-V_L) - g_{Na}m_\infty^3(V_2(t))h_2(t)(V_2(t)-V_{Na}) \\ &\quad - g_K n_2^4(t)(V_2(t)-V_K) - \lambda(V_2(t)-V_1(t-\tau)), \\ \dot{h}_2(t) &= \frac{h_\infty(V_2(t))-h_2(t)}{T_h(V_2(t))}, \\ \dot{n}_2(t) &= \frac{n_\infty(V_2(t))-n_2(t)}{T_n(V_2(t))}. \end{aligned}\right\} \tag{1.63}$$

In practice, numerical experiments in neuroscience are often hard to set up due to the complicated nature of the formulas of the systems and sheer number of their parameters. As such, throughout the work on this thesis, we developed a Julia package *NeuronToolbox.jl*, see its Git repository, to aid in this process. We illustrate this on a coupled pair of an interneuron model (1.60) and a Morris-Lecar model (1.59). The coupling is facilitated through a gap junction, see (1.62), with the coupling strength $\lambda = 0.01$ and corresponding delay $\tau = 0.01$.

```
using NeuronToolbox, ComponentArrays

IN_params = ComponentVector(C=1, I=24, gL=0.1, EL=-60,
   gNa=30, ENa=45, gK=20, EK=-80)
ML_params = Models.MorrisLecarUtils.example_parameters()

IN = Models.Interneuron(IN_params)
ML = Models.MorrisLecar(ML_params)

lambda = 0.01
tau = 0.01
sync_coupling = Couplings.FirstVarDifference(lambda)
async_coupling = Couplings.OneStepDelayed(sync_coupling,
   partition(2, [3, 2]), tau)

coupled_pair = Networks.CoupledPair(IN, ML, async_coupling)
```

The resulting network `coupled_pair` can then be used as a normal function prescribing a (delay) differential equation with DifferentialEquations.jl. Also note that we provide various different models and network types, e.g., an $m \times m$ adjacency matrix coupling for a system of m models.

Chapter 2

Grid Simulation Approach

2.1 Problem Formulation

Neuroscience is among the richest applications of mathematics in the current day and age. In neuronal dynamics, synchronization phenomena occurring in networks of individual neurons are of particular importance. These complex patterns have been tied to the very fundamental capabilities of the brain, be it information processing, memory encoding, or the generation of rhythmic brain activity, see [57], [58].

Although it is likely safe to assume a certain level of robustness of brain functions regarding anomalies in the synchronization pattern, significant abnormalities in the rhythmic brain activity can be attributed to neurological disorders. The sheer variety of neurological disorders invalidates any sort of comprehensive study in a mere master's thesis, thus we shall limit ourselves to focal epilepsy. Let us note that it is still a vast topic in both neurology and applied mathematics and we will only manage to scratch the surface.

From the perspective of neuronal dynamics, these abnormalities (in the case of focal epilepsy) manifest themselves as anomalous synchronization within the neuronal network and can lead to spontaneous seizures, see [59]. This type of synchronization is often closely linked to the presence of fast ripples (oscillations in the frequency range 250–600 Hz), very-fast ripples (600 - 1000 Hz), ultra-fast ripples (1000–2000 Hz), and ultra-fast oscillations (UFOs, above 2000 Hz) within depth electroencephalographic (EEG) recordings from patients with focal epilepsy. This possible connection between focal points of drug-resistant epilepsy and high-frequency EEG reading in their vicinity has been the cause of significant research interest due to the potential use as biomarkers for epileptogenic zones, see [60], [61], [62], [63], [64], [3], [65], and [2]. Let us note the exact mechanism governing these high-frequency oscillations in EEG signals remains unknown, when physiological limitations of action potential firing rates are taken into account, see [7].

Conducted research sheds light on the role of anti-phase synchronization as a

possible mechanism for the emergence of HFOs and VFHOs in epileptic EEG signals, see [5], [4], and [56]. The applied part of this thesis is mainly based on [56] (the author of this thesis also contributed to the publication), although our focus will be the comparison between bifurcation- and simulation-based approaches to the same problem.

Let us thus consider two weakly coupled oscillators (e.g., neurons, clusters thereof, neural masses, etc.). For illustration, these will be two interneuron models connected by a (possibly delayed) gap-junctional coupling, see Section 1.4. By phase synchronization, we understand two (or more) cyclic signals featuring a consistent relationship between their phases. Note that the amplitudes or even frequencies may differ. Understanding of this phenomenon, often called *phase locking* in the context of bifurcation theory, can be aided by examining corresponding *Arnold tongues*, which are regions where a phase synchronization occurs, see [66], [13].

We shall be primarily interested in a system consisting of 2 weakly coupled oscillators with a possible delay in the coupling. In such systems, regions of phase-synchrony between the two oscillators may arise for certain values of the coupling strength and some parameter, which influences the oscillation frequency of one of the oscillators. Then, by the continuous dependence of the solution on parameters and initial values for both the ODEs, see [22] (Theorem 1.3), and DDEs, see [27] (Chapter 2.2), we can explore the phase-shift in the coupled system via small changes in both parameters.

In general, the system of interest takes the following DDE form,

$$\begin{aligned} \dot{\boldsymbol{x}}_1(t) &= \boldsymbol{f}(\boldsymbol{x}_1(t); \omega_1) + \lambda \boldsymbol{K}(\boldsymbol{x}_1(t), \boldsymbol{x}_2(t-\tau), \sigma), \\ \dot{\boldsymbol{x}}_2(t) &= \boldsymbol{f}(\boldsymbol{x}_2(t); \omega_2) + \lambda \boldsymbol{K}(\boldsymbol{x}_1(t-\tau), \boldsymbol{x}_2(t), \sigma), \end{aligned} \tag{2.1}$$

where $\boldsymbol{x}_1(t), \boldsymbol{x}_2(t) \in \mathbb{R}^n$ are the state variables, $\omega_1, \omega_2 \in \mathbb{R}$ parameters influencing oscillation frequencies of the respective models, σ is an arbitrary fixed phase parameter, and $\lambda \in \mathbb{R}_0^+$ the coupling strength. Lastly, $\boldsymbol{K}$ represents the coupling function. Note that $\boldsymbol{x}_i$ does not have to represent a single neuron (or model thereof), but even synchronized clusters of neurons or neural masses[1].

2.2 Grid-Based Phase-Shift Computation

Throughout this section, we shall iteratively develop a framework for computation of the *phase-shift* on the anti-phase tongue for 2 weakly coupled oscillators, most often neurons, their synchronized clusters, or neural masses. Note that the algorithms and procedures described in this section had been formed and tested gradually throughout the research process and are the original work of the author. This should also explain the lack of references.

[1] Neural mass models approximate the behavior of large clusters of neurons based on statistically reasoned simplifications.

Thus consider the system (2.1) and choose ω_1 or ω_2 as the free parameter[2]. For illustration purposes, we shall use 2 interneuron models, see (1.63), with delayed gap-junctional coupling, see Section 1.4.3, e.g., $\tau = 0.01$, and treat ω_2 (through C_2 for (1.63)) as the free parameter. Moreover, assume bounds $\omega_- \leq \omega_2 \leq \omega_+$ and corresponding discretization

$$\underbrace{\overbrace{\omega_2^0}^{\omega_-} < \omega_2^1 < \cdots < \overbrace{\omega_2^{N_\Omega}}^{\omega_+}}_{\Omega},$$

with analogous bounds and discretization chosen for λ as well, producing set Λ of permissible values for λ. Both are most often chosen as equidistant partitions of intervals stemming from respective bounds. We shall denote $\mathbb{M} = \Omega \times \Lambda$ the mesh of all possible combinations of ω_2^i and λ^i and let

$$\mathcal{I}[\mathbb{M}] = \left\{ \boldsymbol{i} = (i_1, i_2) \in \mathbb{N}_0^2 \mid 0 \leq i_1 \leq N_\Omega, 0 \leq i_2 \leq N_\Lambda \right\}$$

be the set of all possible indices in $\mathbb{M}$.

The most straightforward way to determine for which pairs $(\omega_2, \lambda) \in \mathbb{M}$ the coupled system (2.1) features a *stable*[3] anti-phase cycle is to solve the (2.1) forward in time from a fixed initial condition $\boldsymbol{\phi} \in \mathbb{X}_C$ on a fixed time interval $[0, t_{\text{end}}]$. This is a reasonable methodology if we assume the system converges to the synchronized cycle and that we chose a long-enough transient, included in the interval $[0, t_{\text{end}}]$, such that we end up in an acceptably small neighborhood around the limit cycle.

In case there is no delay in the coupling, i.e., $\tau = 0$, then we can integrate the system forward in time, see Section 1.3.1. Similarly, for delayed coupling, one can employ the method of steps, which allows for an analogous forward-in-time integration, see Section 1.3.2. Either way, this results in 2 trajectories, both corresponding to evaluations at times $\left\{t^i\right\}_{i=0}^N$, namely $\left\{\boldsymbol{x}_1^i\right\}_{i=0}^N$ and $\left\{\boldsymbol{x}_2^i\right\}_{i=0}^N$, each for one oscillator. Moreover, denote by

$$\left\{\boldsymbol{\xi}^i\right\}_{i=1}^N = \left\{ \begin{pmatrix} \boldsymbol{x}_1^i \\ \boldsymbol{x}_2^i \end{pmatrix} \right\}_{i=0}^N$$

the joint trajectory. For simplicity, we shall assume that the time interval is divided equidistantly, i.e., there is a constant time-step Δt.

Now, we face 2 main challenges:

1. How to determine the period T of the coupled system (2.1)?
2. Knowing the period T, how to calculate the phase-shift β between the oscillators?

Let us note the following numerical experiments and calculations were accomplished in Julia using a custom-made package `GridWalker.jl`.

[2] This choice is without loss of generality in case $\boldsymbol{K}(\cdot, \cdot)$ is symmetric.

[3] Note that this method is effectively unable to compute unstable cycle manifolds due to the methodology itself.

2.2.1 Period Searching

To calculate the joint period T of (2.1) from $\{\boldsymbol{\xi}^i\}_{i=0}^N$, we propose two distinct methods. The first method relies on the fact that for capacitance-based models of neurons, there are certain notable components, e.g., the membrane potential, which should predominantly determine the overall behavior of the neuronal model. Moreover, we assume the periodic signal of this state variable behaves rather nicely, attaining a single (local) minimum and maximum on the repeating section of the minimal period. However, in practice, we distinguish two main types of neuronal oscillations (behaviors). Neuronal spiking, for which our assumptions hold, is a simple type of neuronal oscillation. It is complemented by a more complex bursting behavior, as is present in pyramidal cells, see [6]. In such a case, an envelope must be constructed, which then allows us to use our methodology on the envelope itself. Nevertheless, by choosing the j-th element of the state vector corresponding to the coupled model, we get a 1-dimensional time series $\{\xi_j^i\}_{i=0}^N$. One can then compute first differences $\{\Delta\xi_j^i\}_{i=1}^N$, where $\Delta\xi_j^i := \xi_j^i - \xi_j^{i-1}$.

Clearly, (local) extrema of the time series $\{\xi_j^i\}_{i=0}^N$ occur when the differences $\{\Delta\xi_j^i\}_{i=1}^N$ change sign[4]. In other words, we attempt to detect peaks (which should reflect the true period) as indices satisfying

$$i \text{ is a peak} \iff \Delta\xi_j^i > 0 \ \wedge \ \Delta\xi_j^{i+1} < 0.$$

Let us denote $I_j^{\uparrow}$ the set of all peaks originating from the choice of notable index j, then the period T can be approximated as the Δt multiple of the *mean difference* $\overline{\Delta I_j^{\uparrow}}$ of the peak indices, i.e.

$$T \approx \hat{T}_\Delta = \Delta t \cdot \overline{\Delta I_j^{\uparrow}}.$$

We shall call $\hat{T}_\Delta$ the *differences period estimate*. Note that while this method is rather inexpensive from the computational point of view, it depends on the appropriate choice of j and the necessary assumption that the period of ξ_j accurately reflects the period of the joint oscillation. For an example, see Figure 2.1.

> **Tip 5: On the length of the monotony of ξ_j^i**
>
> Currently, we require the ξ_j^i to be strictly increasing for only one step left of a peak and decreasing also for only one step right of the peak. This can be extended to require ξ_j^i to be increasing for k_+ steps on the left and decreasing for k_- steps on the right of a peak. Such modification allows us to detect periods of much more complicated signals.

[4] The order determines whether its a local minimum or maximum.

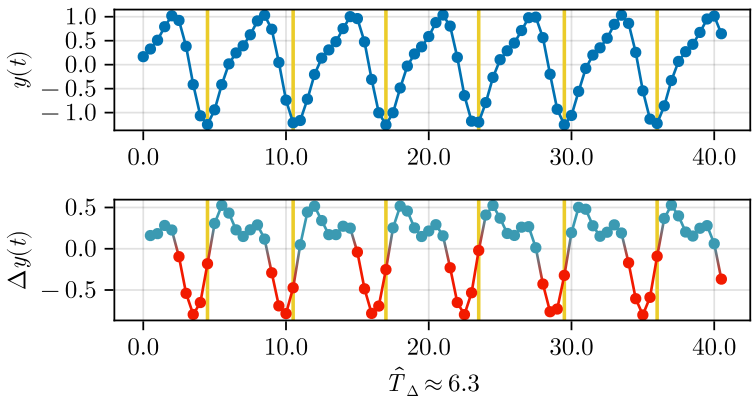


Figure 2.1: Approximation of the period of a test function $y(t) = \sin(t) + \frac{\cos(2t+\frac{\pi}{3})}{3}$ (with true period 2π) by *differences period* estimation method.

The second approach one can use is less heuristic but potentially more computationally intensive. Naturally, the joint period T should minimize the discrepancy between t and $t+T$ states, i.e.,[5]

$$T = \min\left\{\operatorname*{argmin}_{T>0} \int_a^b \|\xi(t) - \xi(t+T)\| \, \mathrm{d}t\right\}, \quad (a,b) \subset \mathbb{R}. \tag{2.2}$$

As it stands, (2.2) is infeasible to solve computationally, but one can rather easily discretize it using $\{\xi^i\}_{i=0}^N$ to obtain

$$\hat{T}_{\min,P} = \operatorname*{argmin}_{T\in P} \overbrace{\sum_{i=0}^{i_{\text{end}-T}} \left\|\xi^i - \xi^{i+\lfloor T/\Delta t\rfloor}\right\|}^{f_\xi(T)}, \tag{2.3}$$

where $P = (p_-, p_+)$, with $p_+ > p_- > 0$, is a chosen interval and $i_{\text{end}-T}$ denotes the maximum index i such that $t^i \le t_{\text{end}} - T$. The discretized problem (2.3) can now be solved using discussed optimization methods for one-dimensional optimization, see Section 1.3.4, under the assumption of *unimodality*![6]

Note that $\hat{T}_{\min,P}$ is *not* guaranteed to approximate the minimal period. In practice, we often repeat solving (2.3) for multiple values of P_i and choose the estimate $\hat{T}_{\min,P_i}$ with the least error. In particular, we found that using $P_i = \left(p_-, \frac{p_+}{2^{i-1}}\right)$ gives satisfactory results. Moreover, for a short interval P, the unimodality of the loss function corresponding to (2.3) is likely to be satisfied. In addition, Figure 2.2 illustrates the algorithm.

[5] We are interested in the *minimal period*, thus we add min to select the least of all permissible values in (2.2).

[6] Without unimodality, these methods converge to a local minimum near the starting point.

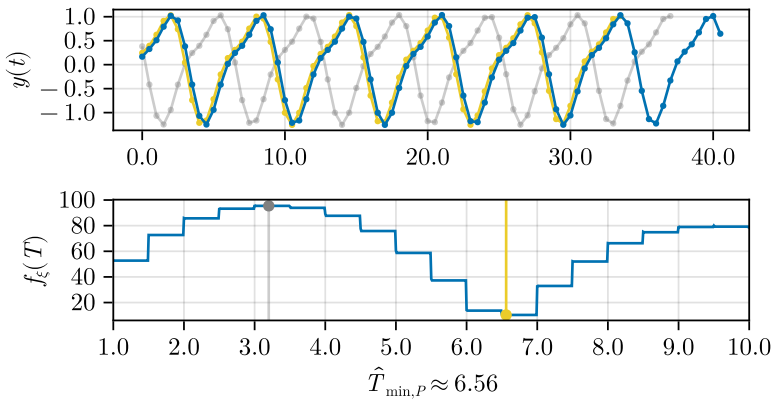


Figure 2.2: Approximation of the period of a test function $y(t) = \sin(t) + \frac{\cos(2t+\frac{\pi}{3})}{3}$ (with true period 2π) by *optimal period* estimation method.

> **Tip 6: Modifications to the optimal period method**
>
> In (2.3), we have used the full state vector for discrepancy calculation, but similarly as with the computation of $\hat{T}_\Delta$, one can choose only a sub-vector as well. Another common modification is to calculate the dissimilarity in (2.3) after a certain transient has passed (such that we presume we have converged to the cycle after the transient), i.e., on the interval with indices from i_{start} to $i_{\text{end}-T}$.

2.2.2 Shift Searching

After one has successfully estimated the period in the joint model, what remains is to approximate the shift between the two oscillators of (2.1). This can be achieved by solving a very similar problem (2.3), namely

$$\hat{\beta}_d = \underset{\beta \in Q}{\operatorname{argmin}} \sum_{i=i_{\text{start}}}^{i_{\text{end}-\beta}} d\left(\boldsymbol{x}_1^i, \boldsymbol{x}_2^{i+\lfloor \beta/\Delta t \rfloor}\right), \tag{2.4}$$

where $Q = (q_-, q_+)$ is a chosen interval with $0 \leq q_- < q_+ < \hat{T}$ and d is a metric. Commonly, metrics induced by L^2-norm $\|\cdot\|_2$ or maximum-norm $\|\cdot\|_\infty$ are used. Moreover, i_{start} can again be used to offset the computation of shift β after a transient has passed.

Note that we shall primarily focus on calculating a shift β between periodic trajectories of a notable component j in each of the oscillators, i.e.,

$$\hat{\beta}_{d,j} = \underset{\beta \in Q}{\operatorname{argmin}} \sum_{i=i_{\text{start}}}^{i_{\text{end}-\beta}} d\left(x_{1,j}^i, x_{2,j}^{i+\lfloor \beta/\Delta t \rfloor}\right). \tag{2.5}$$

In total, (2.4) and (2.5) both lead to a one-dimensional optimization problem on a given interval and as such can be solved using methods proposed in Section 1.3.4. In particular, in the case of shift searching, the unimodality of the corresponding loss function seems to be satisfied more generally.

For a comparison of computed shifts in regards to period estimation based on differences and optimization, see Figures 2.3 and 2.4. There, it is apparent the central Arnold tongue has a rather sharp boundary at some places, which is desired from the theoretical point of view, see [13] and [56], as it is known that Arnold tongues are bordered by limit point of cycle curves, see Note 8. On the other hand, for both low and high values of λ, our approximation of the tongue is either smoothed out (likely signaling that our choice of the initial condition was suboptimal) or scattered.

> **Note 8: Arnold tongue**
>
> In general, Arnold tongues are regions (in parameter space) of rational[7] synchrony near a Neimark-Sacker bifurcation. For a continuous-time dynamical system, this, in essence, means that a given cycle switches stability and a torus with the opposite stability emerges around it (i.e., a stable cycle loses stability when it undergoes the supercritical Neimark-Sacker bifurcation while a stable torus appears around the now-unstable cycle). The dynamics on the torus then provide the two "directions of oscillation", for which we can study its synchrony. Furthermore, two weakly coupled oscillators themselves are governed by dynamics on a torus, such that the influence of coupling strength λ mimics the transition generically originating from the Neimark-Sacker bifurcation.

2.2.3 Starters, Iterators, and Indexers

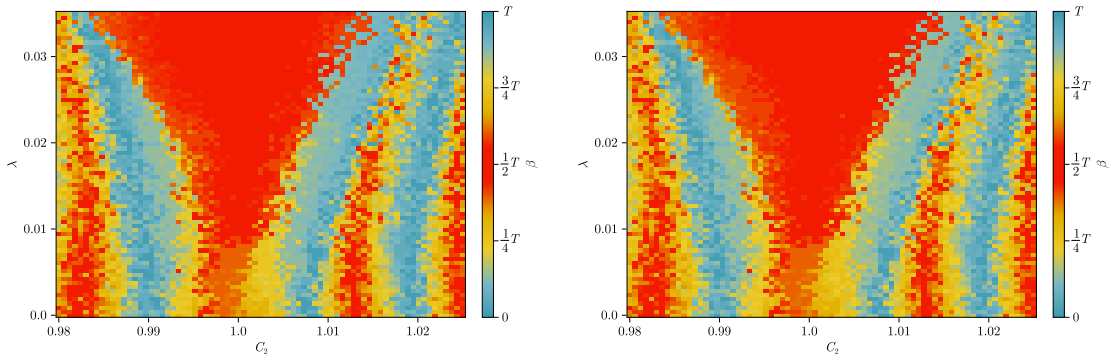
2.2.3.1 Starters

So far, we have utilized a rather naive approach of always using the same predetermined starting condition — we shall refer to this as a *cold* start[8]. Nevertheless, we have no guarantee such a choice was optimal for any grid point, much less for the entire grid.

A natural solution to this problem is to reuse trajectories we have already computed for some other combination of parameters. In general, it is going to be beneficial to use choose as an initial condition the endpoint of the trajectory corresponding

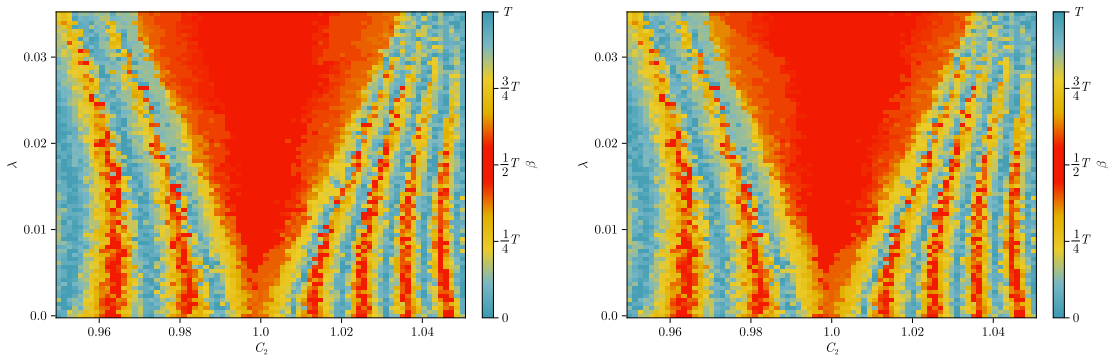
[7] By *rational synchrony* we mean phase-locked synchronization in the form $p : q$, $p, q \in \mathbb{N}$ indivisible, i.e., one period consists of p revolutions around the first axis and q revolutions around the second. For two weakly coupled oscillators, this translates to a period of the full system being completed after p periods of the first oscillators and q of the second one.

[8] For reference on the terminology, see the cold start Wikipedia entry.



(a) *Differences* period estimation (b) *Optimal* period estimation

Figure 2.3: Heatmap of shifts β between two λ-weakly coupled interneuron models with respect to C_2 without delay. Grid is determined as 75-point discretizations of intervals of interest for both C_2 and λ. At every point the same initial condition $\boldsymbol{\phi}(\cdot) \equiv \boldsymbol{u}_0$ was used. The left figure presents the shift heatmap based on the *differences* period, whereas the right figure shows the *optimal* period-based shift. Trajectories are computed for 1500 ms, periods are computed on the last 20%, and shifts on the last 10% of their lengths. Note that the optimal period is calculated by halving the upper constraint up to 4 times. The shift is estimated based on the first state variable, i.e., V, in both interneurons.



(a) *Differences* period estimation (b) *Optimal* period estimation

Figure 2.4: Heatmap of shifts β between two λ-weakly coupled interneuron models with respect to C_2 with delay $\tau = 0.05$. Note that except for the time integration (performed by the continuous ODE method, see Section 1.3.2), the same methods were used as in Figure 2.3.

to parameter values *close* to our current ones. Moreover, we want to consider only those nearby parameter values such that their trajectories lead to anti-phase synchronization. In total, the optimal reference index starter($\boldsymbol{i}$) $\in \mathcal{I}[\mathbb{M}]$, from which to read the new initial condition, for the shift computation at $\boldsymbol{i}$ (using $\boldsymbol{i} \in \mathcal{I}[\mathbb{M}]$ to denote the combined index in both axis of the grid $\mathbb{M}$) is given by the following minimization problem,

$$\text{starter}_{\mathcal{I}[\mathbb{M}]}(\boldsymbol{i}) = \underset{\boldsymbol{j}\in\mathcal{I}[\mathbb{M}]}{\operatorname{argmin}} \operatorname{loss}(\boldsymbol{i}, \boldsymbol{j}, \beta_{\boldsymbol{j}}), \tag{2.6}$$

where we use either the *multiplicative* loss,

$$\text{loss}_{\text{mult}}(\boldsymbol{i}, \boldsymbol{j}, \beta) = \|\boldsymbol{i} - \boldsymbol{j}\| \cdot \left(\left|\beta - \frac{1}{2}\right| + c\right), \tag{2.7}$$

where $c > 0$ is an offset to the discrepancy in the shift from anti-phase, or the *additive* loss,

$$\text{loss}_{\text{add}}(\boldsymbol{i}, \boldsymbol{j}, \beta) = c_1 \|\boldsymbol{i} - \boldsymbol{j}\| + c_2 \left|\beta - \frac{1}{2}\right|, \tag{2.8}$$

where $c_1, c_2 > 0$ are chosen weights, see GridWalker.jl Git repository. Note that we will use additive loss (2.8) if not stated otherwise, as it seemed to perform generally better for our needs. Moreover, we deduced from our experimentation that it was better to prioritize minimizing the loss difference as opposed to distance[9].

While this approach is ideal in theory, in practice we run into two main issues. Firstly, during the computation, some of the indices are not yet evaluated, thus we do not know their respective values of shifts $\beta_{\boldsymbol{i}}$. This can be remedied by defining the set of all *evaluated* indices $\mathcal{I}_E[\mathbb{M}]$ and restricting (2.6) to it, yielding

$$\text{starter}_{\mathcal{I}_E[\mathbb{M}]}(\boldsymbol{i}) = \underset{\boldsymbol{j}\in\mathcal{I}_E[\mathbb{M}]}{\operatorname{argmin}} \operatorname{loss}(\boldsymbol{i}, \boldsymbol{j}, \beta_{\boldsymbol{j}}). \tag{2.9}$$

2.2.3.2 Indexers

The second issue is present even in (2.9), and it is the dependence on the entire grid. Indeed, for distant indices the loss function, be it (2.7) or (2.8), will be large and irrelevant for the minimum. Hence, we can define an *indexer* function

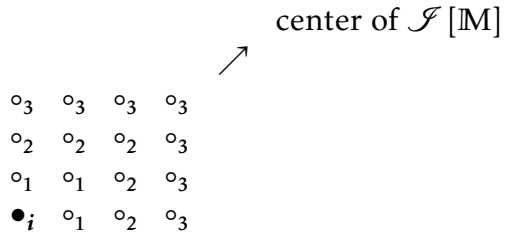
$$\text{indx} : \mathcal{I}[\mathbb{M}] \to 2^{\mathcal{I}[\mathbb{M}]} \tag{2.10}$$

specifying a permissible neighborhood for every point of the grid.

Arnold tongues have a distinct shape, see [19], [13], or [56], which we can exploit to restrict the search space of a starter. Indeed, if we assume the Arnold tongue with the anti-phase synchrony is present approximately in the middle of our grid

[9] As will be evident later, by using an *indexer* we limit possible discrepancy in indices (i.e., the parametric distance). This effectively supercharges the effect of c_1 in (2.8), as it is technically included in the construction of the indexer (thus forcing us to prioritize shift discrepancy).

$\mathbb{M}$, then elements closer to the center are more likely to live inside the tongue (and therefore be a good initial guess for anti-phase synchronized trajectories for nearby parameter values). Thus, choosing a starter only among evaluated indices on the diagonal from the current index to the center of $\mathscr{S}[\mathbb{M}]$ up to some distance away, as can be seen in the following illustration, should not worsen our initial guess as compared to the full *warm starter*.



We refer to the described method as the *diagonal indexer* of length $l_{\text{indx}} \in \mathbb{N}$ (example shown with $l_{\text{indx}} = 3$). The combination of a *warm starter* with *indexer*, which we shall call an *indexed starter*, yields the minimization problem

$$\text{starter}_{\text{indx}_E}(\boldsymbol{i}) = \underset{\boldsymbol{j} \in \mathscr{S}_E[\mathbb{M}] \cap \text{indx}(\boldsymbol{i})}{\text{argmin}} \text{loss}(\boldsymbol{i}, \boldsymbol{j}, \beta_{\boldsymbol{j}}). \tag{2.11}$$

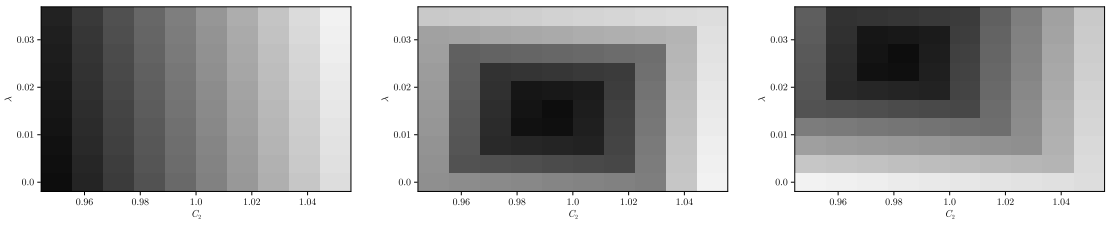
Depending on the choice of the indexer, an indexed starter might bring a very important computational benefit of *constant complexity* with respect to grid size. Indeed, by employing a fixed-length diagonal indexer, the indexed starter[10] $\text{starter}_{\text{indx}_E}(\boldsymbol{i})$ requires comparison of at most $l_{\text{indx}}^2 - 1$ values of $\text{loss}(\boldsymbol{i}, \boldsymbol{j}, \beta_{\boldsymbol{j}})$, where $l_{\text{indx}} \ll \min\{N_\Omega, N_\Lambda\}$.

2.2.3.3 Iterators

So far, we have discussed how to choose initial conditions from endpoints of already computed trajectories and how to limit the number of candidate indices. The last undetermined process is the actual order of evaluation of indices. For cold starts, the order could be arbitrary, as it did not influence the rest of the algorithm. However, for *warm* (2.9) and *indexed* starters (2.11), we have an explicit dependence on already evaluated indices and their corresponding trajectories. In both cases, if no viable indices are found, the starter falls back to a fixed initial condition (i.e., to cold starter).

The "natural" order of iteration along the mesh is deduced from its representation in the Julia code as two separate ranges. Then iterating over their cartesian product, i.e., the grid $\mathbb{M}$, amounts to subsequently evaluating each column of $\mathbb{M}$, see Figure 2.5a. Notice that such evaluation order is suboptimal for the *diagonal indexer*, which will likely default to a cold start for indices on the left side of the grid.

[10] The naive warm starter has a time complexity $O(N_\Omega \cdot N_\Lambda)$ when considering the computation of shift to have constant complexity for all indices $\boldsymbol{i}$.

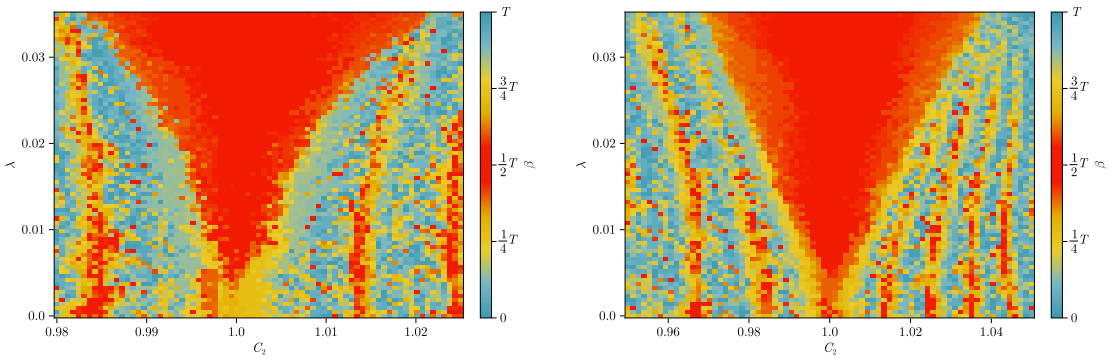


(a) Line iterator (b) Centered spiral iterator (c) Shifted spiral iterator

Figure 2.5: Heatmaps indicating evaluation order (from black to white) for different kinds of iterators from `GridWalker.jl`.

From the perspective of stability of tongue detection, a more robust strategy is to start the evaluation order from the (presumed) center of the tongue and spirally iterate outwards. This way, an indexed starter can likely initialize from a parametrically nearby trajectory in anti-phase synchronization because suitable candidates are evaluated first, see Figure 2.5b and Figure 2.5c.

All these concepts can now be combined. In Figure 2.6, one can see the same heatmaps corresponding to the shift between two weakly coupled interneurons as in Figures 2.3 and 2.4. In particular, an indexed starter $\text{starter}_{\text{indx}}$ was used with diagonal indexer of length $l_{\text{indx}} = 10$ on a 75-by-75 grid $\mathbb{M}$. An additive loss (2.8) was used for the starter in conjunction with a spiral iterator from the true center. The rest of the experiment setup follows Figure 2.3.



(a) $\tau = 0$ (b) $\tau = 0.025$

Figure 2.6: Heatmap of shifts β between two λ-weakly coupled interneuron models with respect to C_2 with and without delay. The heatmaps are computed by utilizing introduced techniques of starters, indexers, and iterators. The rest of the procedure follows Figure 2.3 and Figure 2.4.

Notice that in Figure 2.6a, the boundary of the tongue for higher coupling strengths is sharper than in Figure 2.3. Also, sub-tongues present in all previous results now appear thinner and more distorted.

2.2.4 Periodicity Checkers

In Figures 2.3 to 2.6, we have always computed a trajectory for each parameter combination belonging to a point on the grid $\mathbb{M}$ and then attempted to find the corresponding period and shift. This relied on the assumption that the (joint) trajectory is always periodic, and thus all necessary quantities are well-defined and can be computed. In case the assumption does not hold, we will still get an output, but potentially a nonsensical one.

In other words, one should check the periodicity of the trajectory for each parameter combination. If periodicity is not confirmed, we can skip the rest of the algorithm, saving valuable computing time.

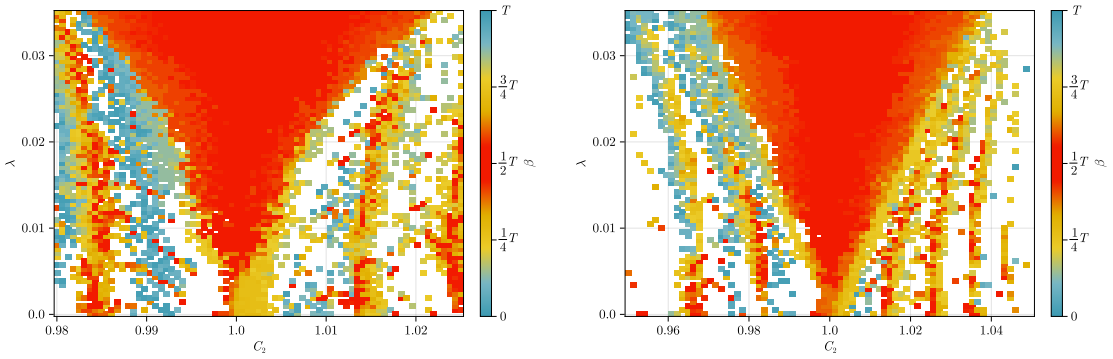
There are several ways one might go about determining periodicity. One possibility is to consider a band around a reference value with respect to $\boldsymbol{x}_{1,i}$, the i-th coordinate of the state corresponding to the first oscillator — local maxima work especially well for this approach. Then we can select indices of all data points lying inside this band and study the distribution of i-th coordinate of trajectory points for the 2nd oscillator, i.e., $\boldsymbol{x}_{2,i}$, at these indices. However, this method is very sensitive to the width of the band and the reference value, rendering it almost unusable in practice.

An alternative approach is to detect peaks of $\boldsymbol{x}_{1,i}$, the i-th coordinate of the trajectory of the 1st oscillator. This again prescribes an index set I, which we can apply to $\boldsymbol{x}_{2,i}^I$. Then, we can check whether $\boldsymbol{x}_{2,i}^I$ follows chosen criteria, e.g., stay within a predefined range. If this check is successful, we assume the trajectory as a whole is periodic and execute the rest of the procedure, see Figure 2.7.

> **Note 9: Peak-based periodicity checker**
>
> A careful reader might notice the peak-based periodicity checker closely resembles *differences* period searcher, see Section 2.2.1. Indeed, it should be possible to merge these two concepts into a single algorithm. Nevertheless, we decided against it for code-readability reasons. Moreover, the computational complexity of the peak-based periodicity checker is almost negligible when considering the entire run of the algorithm over the whole grid, even more so when taking into account the time saved on shift searching for skipped indices $\boldsymbol{i} \in \mathcal{J}[\mathbb{M}]$.

2.2.4.1 Sub-tongues

Throughout this chapter, we have discussed and implemented various measures to ensure optimal estimation of the true shift between the oscillators. However, none of our modifications have removed the sub-tongues, which can be seen in every shift heatmap. By examining the trajectories at their corresponding points, one can verify both the periodicity and the computed shift. Their emergence is likely due to our choice of the methodology. In particular, they are probably caused by



(a) $\tau = 0$ (b) $\tau = 0.025$

Figure 2.7: Heatmap of shifts β between two λ-weakly coupled interneuron models with respect to C_2 with and without delay, such that for each parameter combination a periodicity is tested before continuing with the computation. The rest of the computation follows Figure 2.6.

either insufficiently long transient (i.e., the convergence to the attractor is very slow and the oscillators remain near the initial anti-phase), the discretization of the parameter-space to the grid, or even possibly the choice of norms/distances for period and shift searching. Either way, these phenomena could produce a similar fractal structure.

However it may be, this is *the* problem of the grid simulation, but also in a way its strength. Indeed, from what we have computed, we cannot guarantee these sub-tongue regions of anti-phase synchronization actually exist and are not just artifacts from our calculation. Nevertheless, the grid simulation *along with a suitable choice of iterator* might tell us about the behavior of the studied system in nature, because it reflects a continuous change of parameters, which might not always have enough time to converge to the real attractor. Note that the *spiral iterator* is an especially good choice, as it radially propagates the parametric change, avoiding sudden jumps and discontinuities.

2.2.5 Multi-threading and Grid Computing

The Figures 2.3 - 2.7 presented so far have all been computed on a Thinkpad T490s with Intel Core i7-8565U CPU and 16 GiB of RAM. We have utilized at most 6 threads and the computations took at most approximately 3 minutes. However, if we were to increase the grid density, the computation time would increase quadratically. Indeed, the calculation of a 1200-by-1200 grid would take at least approximately 768 minutes (considering 16 GiB of RAM would be enough, and thus swapping would not be needed, which could cause a significant slow-down).

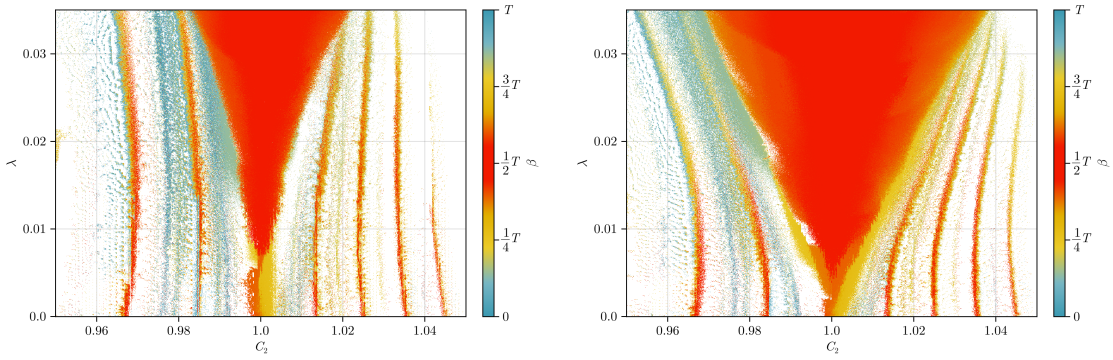
Note that multi-threading is not trivial to implement for our algorithm. A wrong execution order might entirely defeat the purpose of the chosen iterator. Therefore, we must force each thread to evaluate correct indices, e.g., l-th thread might be

restricted to $\{\boldsymbol{i}_{l(k+1)-1}\}_{k=0}^{N_l}$ (for appropriate $N_l < N_{\mathbb{M}}$).

In general, scientific computing is often too demanding for a personal computer. One possible solution is to offload the computation to grid infrastructure, namely MetaCentrum.

> *MetaCentrum is the leading national grid computing infrastructure in the Czech Republic, providing researchers, scientists, and students with advanced computing resources to tackle complex computational challenges.*

For illustration purposes, we ran the script generating Figure 2.7 for a 1200-by-1200 grid instead of 75-by-75, see Figure 2.8. As we have said, on a regular consumer-grade laptop, such computation would take at the *very least* 750 minutes simply extrapolating from the 75-by-75 grid computation. On MetaCentrum, we reserved 20 CPU cores, 36 GiB of RAM, and 150 GiB scratch storage with a maximal runtime of 16 hours. Using this setup, the computation took 82 minutes for the experiment without delays and 443 minutes with delay.



(a) $\tau = 0$ (b) $\tau = 0.025$

Figure 2.8: Heatmap of shifts β between two λ-weakly coupled interneuron models with respect to C_2 with and without delay computed on MetaCentrum grid. Save for grid density, the setup of the experiment mimics Figure 2.7.

Caution 2: Stochastic simulations

So far, we have only used deterministic simulations for every computation of the shifts heatmap. In theory, one could use the Euler-Maruyama method, see Algorithm 4, to generate a stochastic realization of the heatmap. Then, using a sort of Monte Carlo approach, we could repeat the stochastic grid generation to obtain the distribution of shift for each point of the discretized grid. However, such modification would drastically increase the computational time simply due to the repeated computations, but also because of the less "efficient" Euler-Maruyama solver (as compared to Runge-Kutta methods or similar).

Chapter 3

Bifurcation Theory Approach

As we have mentioned several times before, the fundamental goal of this thesis is to study the computational aspects of continuation/detection of stable and unstable manifolds of periodic orbits (and what challenges come with it). As such, we shall properly discuss what a continuation is and how we compute it. We begin with a theoretical treatment of bifurcation theory, focused solely on cycle continuation and stability identification, followed by original results concerning coupled models of neurons. The theoretical introduction is mainly based on [13], [16], and [18], but many more related articles are cited throughout the discussion.

> *We should also disclose that while this chapter is named "Bifurcation Theory Approach", we will spend relatively little time discussing actual bifurcations. Our main focus lies in the continuation of (un)stable cycles.*

Although we primarily rely on existing implementations (MATCONT, see [67] and [68], for continuation in ODEs and DDE-BIFTOOL, see [69], in DDEs), in this chapter we also provide a rather thorough theoretical and numerical overview of the underlying methods and algorithms. Moreover, honoring the saying "do not reinvent the wheel", we can base our conclusions on peer-reviewed and community-tested tools, further strengthening our claims. Lastly, a novel technique along with its implementation for the continuation of shift between two oscillators is provided (through a referenced co-authored article).

3.1 Theory of Localizations

Although the primary subjects of our study are periodic orbits, we will first delve into the localization of equilibrium. Indeed, one might recall we discussed the relation of equilibria (or fixed points) in discrete dynamical systems and periodic orbits of continuous-time dynamical systems in Section 1.1.4. This fact alone suggests it is beneficial to first study the simpler case of equilibria localization, building tools to later tackle the periodic case.

3.1.1 Dynamical Systems

3.1.1.1 Equilibrium Localization

To start relatively simply, consider a continuous-time dynamical system induced by an autonomous ODE of the form (1.4), i.e.,

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}),\ \boldsymbol{x} \in \mathbb{R}^n.$$

Its equilibrium $\boldsymbol{x}^*$ is then given by

$$\boldsymbol{f}(\boldsymbol{x}^*) = 0. \tag{3.1}$$

Note that similarly by setting $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{x}$ we can get the same equilibrium condition for a discrete-time dynamical system $\boldsymbol{x} \mapsto \boldsymbol{g}(\boldsymbol{x})$.

It is easy to see that if the equilibrium $\boldsymbol{x}^*$ is stable, one can integrate the system (1.4) forward in time starting from some point $\boldsymbol{x}$ in the basis of attraction of $\boldsymbol{x}^*$, since

$$\lim_{t\to\infty} \left\| \boldsymbol{\varphi}(t, \boldsymbol{x}) - \boldsymbol{x}^* \right\| = 0$$

by Definition 1.11. Similarly, if $\boldsymbol{x}^*$ is *totally unstable* (i.e., *repelling* from all directions), we can simply reverse time (as we are in the *ODE* case) and repeat the procedure[1].

Nonetheless, in general, this approach will not work, as equilibria are rarely stable. The standard procedure is to start from a point in a small neighborhood around the equilibrium (determined either analytically or via numerical integration) and construct a sequence of points $\{\boldsymbol{x}^i\}_{i=0}^{\infty}$, which converges to $\boldsymbol{x}^*$ under general conditions. For this purpose, we can use Newton's method, see Section 1.3.5, or one of its modifications.

Suppose we have found $\boldsymbol{x}^*$ with the desired accuracy. Our next task is then to determine its stability, i.e., how the phase portrait behaves in its vicinity. By Theorem 1.3, Theorem 1.2, and the following discussion, we know that eigenvalues of the Jacobian matrix J^* determine the stability of $\boldsymbol{x}^*$. In other words, we need to compute the roots of the *characteristic polynomial*

$$p(\lambda) = \det(\mathrm{J}^* - \lambda \boldsymbol{I}).$$

Note that there are other methods of computing stability, which in particular do not require eigenvalues, but only computation of certain determinants of J^*. We refer an interested reader to [13], page 470.

[1] Note that this behavior is *local* and global properties are often more complicated, which is even more prevalent in higher dimensions.

3.1.1.2 Periodic Orbit Localization

As we have seen, localization of an equilibrium from a sufficiently close initial guess was rather simple both theoretically and computationally. Unfortunately, for limit cycles, the situation is more complicated. For clarity, [70] was heavily used as a source material for this section.

Assume again we have a dynamical system prescribed by (1.4),

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}),\ \boldsymbol{x} \in \mathbb{R}^n,$$

such that it has an isolated periodic orbit L_0. Let $\boldsymbol{x}^0(t + T_0) = \boldsymbol{x}^0(t)$ be the corresponding solution with minimal period $T_0 > 0$. Further assume that the cycle L_0 has corresponding multipliers, see Section 1.1.4 and Lemma 1.1,

$$\mu_1, \dots, \mu_n \in \mathbb{C},$$

which are by Theorem 1.7 the eigenvalues of the $n \times n$ monodromy matrix $\boldsymbol{M}(T_0)$, where $\boldsymbol{M}(t)$ satisfies

$$\left.\begin{aligned} \dot{\boldsymbol{M}}(t) &= \nabla \boldsymbol{f}(t)\boldsymbol{M}(t), \\ \boldsymbol{M}(0) &= \boldsymbol{I}_n. \end{aligned}\right\} \tag{3.2}$$

Recall there is always a trivial multiplier $\mu_1 = 1$. Also, if $|\mu_i| < 1$ for all $i \in [n]$, then the cycle is stable. In contrast, if there exists $i \in [n]$ such that $|\mu_i| > 1$, the cycle is unstable (and if all multipliers have a modulus greater than one, the cycle is called *totally unstable* or *repelling*).

If the system features a stable limit cycle L_0, then we can find it again by numerical integration from a point in its basin of attraction. In contrast to the case of equilibria, backward-time numerical integration will, in general, not help us find a repelling periodic orbit.

Suppose we have a system in *2-dimensional* state space, which possesses a periodic orbit, see Figure 3.1. From the theory of differential equations, we know that in the case of an autonomous ODE (1.4), its trajectories do not intersect. Thus a periodic orbit L_0 constitutes a Jordan curve[2]. Then by Jordan curve theorem, see [71], such orbit partitions the state space into inside and outside of the cycle (and the cycle itself)[3]. Thus if L_0 is attracting, the trajectories are guaranteed to converge to L_0 forward in time, whereas for repelling L_0, they will converge backward in time.

On the other hand, for three- or more dimensional dynamical systems, a periodic orbit does not partition the state space (or any local neighborhood). Then, we cannot

[2] By (1.4) and the periodicity constraint, there exists a map $\boldsymbol{\phi}(t) = \varphi(t, \boldsymbol{x}(t))$ such that it maps the interval $[t_0, t_0 + T_0]$ onto the cycle L_0 and is surely continuous by Definition 1.1 and Definition 1.14 for sufficiently smooth $\boldsymbol{f}$ (otherwise $\boldsymbol{f}$ would not give rise to a dynamical system in the first place). Moreover, $\boldsymbol{\phi}(t)$ is injective on $[t_0, t_0 + T_0)$ as there can be no intersections. In total, $L_0 = \boldsymbol{\phi}([t_0, t_0 + T_0])$ is a Jordan curve.

[3] In practice, it suffices to consider a sufficiently large connected region containing the cycle instead of the entire state space

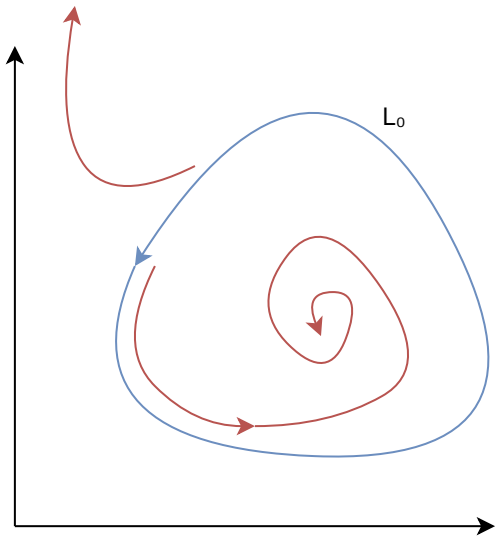


Figure 3.1: Unstable periodic orbit L_0 of a 2-dimensional dynamical system.

guarantee a convergence to the repelling cycle by backward-time integration no matter the initial vicinity. Moreover, it is needless to say the numerical integration localization approach will surely fail if the cycle is neither stable nor repelling. In other words, we cannot even *detect* the cycle if it is not attracting. Note that this very fact is the crucial support for the bifurcation theory-based approach. If our studied system has a stable cycle for some parameters, then we can continue with respect to a given parameter. Should the periodic orbit become unstable, this approach, as we shall see later, will still allow us to detect and continue the cycle manifold. However, no matter the modifications, this is simply not possible with a simulation-on-grid approach as in Chapter 2.

From now on, we will assume we know the location of the periodic orbit approximately. A correction method, most often Newton-Raphson, is then applied repeatedly until satisfactory convergence is achieved. This is a rather common scenario in practice, where we often know the location of the cycle for a given parameter value and wish to "continue" by varying the parameter and correcting the cycle afterward. This process is commonly referred to as *continuation*.

Usually, we take the period T of L_0 as an unknown and formulate the *boundary-value problem* (BVP) on a fixed interval. In particular, assume T is a parameter and construct a "time-normalized" system

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}s} = T\boldsymbol{f}(\boldsymbol{u}),\ s \in [0,1], \tag{3.3}$$

which rescales (1.4) by a time-scaling factor T, such that the new time is denoted s. Now, if a solution $\boldsymbol{u}(s)$ to (3.3) also satisfies the *periodic boundary conditions*

$$\boldsymbol{u}(0) = \boldsymbol{u}(1), \tag{3.4}$$

then it corresponds to a T-periodic solution of (1.4). However, these two conditions do not prescribe the period orbit of (1.4) uniquely. Indeed, any time shift of the solution to the BVP (3.3), (3.4) also determines the same periodic orbit.

Therefore, an extra condition, called the *phase condition*, must be added. Such condition is most often written in the form

$$\Phi[\boldsymbol{u}] = 0, \tag{3.5}$$

where Φ is a scalar *functional* defined on the space of periodic solutions. The exact choice of phase condition is up to the user, but the most frequent one is the *integral phase condition*

$$\Phi[\boldsymbol{u}] = \int_0^1 \langle \boldsymbol{u}(s), \dot{\boldsymbol{v}}(s) \rangle \, \mathrm{d}s, \tag{3.6}$$

where $\boldsymbol{v}(s) \in C([0,1], \mathbb{R}^n)$ is the reference period-one solution.

Lemma 3.1. *The integral phase condition (3.6) is a necessary condition for a local minimum of the time-shift L_2-distance of period-1 smooth functions $\boldsymbol{u}, \boldsymbol{v} : \mathbb{R} \to \mathbb{R}^n$*

$$\rho(\sigma) = \int_0^1 \| \boldsymbol{u}(s+\sigma) - \boldsymbol{v}(s) \|_2^2 \, \mathrm{d}s,$$

such that the minimum is obtained at shift $\sigma = 0$.

Proof. See [70], lemma 11. □

We can now collect all the conditions to obtain the periodic BVP

$$\left.\begin{aligned} \dot{\boldsymbol{u}}(s) &= T\boldsymbol{f}(\boldsymbol{u}(s)), \ s \in [0,1] \\ \boldsymbol{u}(0) &= \boldsymbol{u}(1), \\ \int_0^1 \langle \boldsymbol{u}(s), \dot{\boldsymbol{v}}(s) \rangle \, \mathrm{d}s &= 0. \end{aligned}\right\} \tag{3.7}$$

If $(\boldsymbol{u}(\cdot), T_0) \in C^1([0,1], \mathbb{R}^n) \times \mathbb{R}$ satisfies (3.7), then $\boldsymbol{x}(t) = \boldsymbol{u}\left(\frac{t}{T_0}\right)$ gives the T_0-periodic solution of (1.4) with $\boldsymbol{x}(0) = \boldsymbol{u}(0)$.

Moreover, for the fundamental matrix solution $\boldsymbol{M}(t)$ of (1.4) we define $\boldsymbol{\Gamma}(s)$, such that for the monodromy matrix of L_0 holds $\boldsymbol{M}(T) = \boldsymbol{\Gamma}(1)$ and

$$\dot{\boldsymbol{\Gamma}}(s) - T\nabla \boldsymbol{f}(\boldsymbol{u}(s))\boldsymbol{\Gamma}(s) = \boldsymbol{0}, \ \boldsymbol{\Gamma}(0) = \boldsymbol{I}_n. \tag{3.8}$$

The eigenvalues of $\boldsymbol{\Gamma}(1)$ correspond exactly to the aforementioned multipliers of the cycle L_0. We also define the **adjoint monodromy matrix** $\boldsymbol{\Psi}(1)$ as the solution of

$$\dot{\boldsymbol{\Psi}}(s) - T\nabla \boldsymbol{f}^\top(\boldsymbol{u}(s))\boldsymbol{\Psi}(s) = \boldsymbol{0}, \ \boldsymbol{\Psi}(0) = \boldsymbol{I}_n$$

evaluated at 1. It follows that

$$\boldsymbol{\Psi}(s) = \left(\boldsymbol{\Gamma}(s)^{-1}\right)^\top. \tag{3.9}$$

In general, any solution $\boldsymbol{\xi} \in C^1([0,1],\mathbb{R}^n)$ of an inhomogeneous linear system

$$\dot{\boldsymbol{\xi}} - T\nabla \boldsymbol{f}(\boldsymbol{u}(s))\boldsymbol{\xi} = \boldsymbol{b}(s),$$

where $\boldsymbol{b} \in C^0(\mathbb{R},\mathbb{R}^n)$, can be written as (by (3.9))[4]

$$\boldsymbol{\xi}(s) = \boldsymbol{\Gamma}(s)\left(\boldsymbol{\xi}(0) + \int_0^s \boldsymbol{\Gamma}^{-1}(\sigma)\boldsymbol{b}(\sigma)\,\mathrm{d}\sigma\right) = \boldsymbol{\Gamma}(s)\left(\boldsymbol{\xi}(0) + \int_0^s \boldsymbol{\Psi}^\top(\sigma)\boldsymbol{b}(\sigma)\,\mathrm{d}\sigma\right). \tag{3.10}$$

Definition 3.1. A cycle L is called **simple** if the trivial multiplier $\mu_1 = 1$ has algebraic multiplicity[5] equal to 1.

Let $\boldsymbol{q}_0, \boldsymbol{p}_0 \in \mathbb{R}^n$ be the left and right eigenvectors of the monodromy matrix corresponding to the trivial multiplier μ_1,

$$\begin{aligned}(\boldsymbol{\Gamma}(1) - \boldsymbol{I}_n)\boldsymbol{q}_0 &= (\boldsymbol{\Psi}(1) - \boldsymbol{I}_n)\boldsymbol{p}_0 = \boldsymbol{0},\\ (\boldsymbol{\Gamma}(1) - \boldsymbol{I}_n)^\top \boldsymbol{p}_0 &= (\boldsymbol{\Psi}(1) - \boldsymbol{I}_n)^\top \boldsymbol{q}_0 = \boldsymbol{0},\end{aligned}$$

such that $\|\boldsymbol{p}_0\|_2 = \|\boldsymbol{q}_0\|_2 = 1$. Also, $\boldsymbol{q}_0 = c_0\boldsymbol{f}(\boldsymbol{u}(0))$ for $c_0 \in \mathbb{R}\setminus\{0\}$, which follows from (3.8) using (3.3), (3.4):

$$\begin{aligned}\frac{\partial}{\partial s}(c_0\boldsymbol{f}(\boldsymbol{u}(0))) - T\boldsymbol{f}(\boldsymbol{u}(1))c_0\boldsymbol{f}(\boldsymbol{u}(0)) &= c_0\nabla\boldsymbol{f}(\boldsymbol{u}(0))\dot{\boldsymbol{u}}(0) - T\nabla\boldsymbol{f}(\boldsymbol{u}(0))c_0\boldsymbol{f}(\boldsymbol{u}(0))\\ &= c_0\nabla\boldsymbol{f}(\boldsymbol{u}(0))T\boldsymbol{f}(\boldsymbol{u}(0)) - T\nabla\boldsymbol{f}(\boldsymbol{u}(0))c_0\boldsymbol{f}(\boldsymbol{u}(0))\\ &= \boldsymbol{0}.\end{aligned}$$

As we have mentioned, the workflow for localization of a periodic orbit is to start from an initial guess $(\boldsymbol{u},T)$, which we correct by $(\boldsymbol{w},S) \in C^1([0,1],\mathbb{R}^n)\times\mathbb{R}$, i.e.,

$$(\boldsymbol{u},T) \mapsto (\boldsymbol{u}+\boldsymbol{w}, T+S),$$

where $(\boldsymbol{w}(\cdot),S)$ is the solution of the linearized inhomogeneous BVP (for reference see (3.7))

$$\left.\begin{aligned}\dot{\boldsymbol{w}}(s) - T\nabla\boldsymbol{f}(\boldsymbol{u}(s))\boldsymbol{w} - S\boldsymbol{f}(\boldsymbol{u}(s)) &= -\dot{\boldsymbol{u}}(s) + T\boldsymbol{f}(\boldsymbol{u}(s)),\ s\in[0,1],\\ \boldsymbol{w}(0) - \boldsymbol{w}(1) &= -\boldsymbol{u}(1) + \boldsymbol{u}(0),\\ \int_0^1 \langle \dot{\boldsymbol{v}}(\sigma), \boldsymbol{w}(\sigma)\rangle\,\mathrm{d}\sigma &= -\int_0^1 \langle \dot{\boldsymbol{v}}(\sigma), \boldsymbol{u}(\sigma)\rangle\,\mathrm{d}\sigma.\end{aligned}\right\} \tag{3.11}$$

[4] In other words, $\boldsymbol{\Gamma}(s)$ is a *fundamental matrix* of (3.8). Moreover, this viewpoint explains (3.10) as a corollary of the method of variation of constants.

[5] Recall the algebraic multiplicity of an eigenvalue λ of a matrix $\boldsymbol{A}$ is its multiplicity as a root of the characteristic polynomial. Furthermore, geometric multiplicity of λ is defined as $\dim\ker(\boldsymbol{A}-\lambda\boldsymbol{I})$. The algebraic multiplicity is always greater than or equal to the geometric multiplicity for any given eigenvalue λ of $\boldsymbol{A}$.

The left-hand side of (3.11) can be expressed as a matrix operator of the form

$$\underbrace{\begin{bmatrix} \nabla - T\nabla \boldsymbol{f}(\boldsymbol{u}) & -\boldsymbol{f}(\boldsymbol{u}) \\ \eta_0 - \eta_1 & 0 \\ \mathrm{Int}_{\dot{\boldsymbol{v}}} & 0 \end{bmatrix}}_{\mathsf{L}_{\boldsymbol{u},T}} \begin{pmatrix} \boldsymbol{w} \\ S \end{pmatrix},$$

where ∇ denotes the differentiation operator, η_a is the evaluation operator at $t = a$, i.e., $\eta_a \boldsymbol{w} = \boldsymbol{w}(a)$, and

$$\mathrm{Int}_{\dot{\boldsymbol{v}}} \boldsymbol{w} = \int_0^1 \langle \dot{\boldsymbol{v}}(\sigma), \boldsymbol{w}(\sigma) \rangle \, \mathrm{d}\sigma.$$

By taking $\boldsymbol{v} = \boldsymbol{u}$, where $\boldsymbol{u}$ is a solution of (3.7), we get $\dot{\boldsymbol{v}} = \dot{\boldsymbol{u}} = T\boldsymbol{f}(\boldsymbol{u})$. Moreover, for the following theorem, one might choose $\boldsymbol{v}$ sufficiently close to such $\boldsymbol{u}$ (most importantly when $\boldsymbol{v}$ is a reference period-1 function, for example, the solution in the previous step of continuation). Lastly, the replacement of $T\boldsymbol{f}(\boldsymbol{u})$ by $\boldsymbol{f}(\boldsymbol{u})$ does not change the fundamental properties of the operator $\mathsf{L}_{\boldsymbol{u},T}$.

Theorem 3.1. *If $(\boldsymbol{u}(\cdot), T)$ corresponds to a simple cycle, then the operator*

$$\mathsf{L}_{\boldsymbol{u},T} = \begin{bmatrix} \nabla - T\nabla \boldsymbol{f}(\boldsymbol{u}) & -\boldsymbol{f}(\boldsymbol{u}) \\ \eta_0 - \eta_1 & 0 \\ \mathrm{Int}_{\boldsymbol{f}(\boldsymbol{u})} & 0 \end{bmatrix},$$

from $C^1([0,1], \mathbb{R}^n) \times \mathbb{R}$ into $C^1([0,1], \mathbb{R}^n) \times \mathbb{R}^n \times \mathbb{R}$ is one-to-one and onto.

Proof. See [70], page 36. □

By Theorem 3.1 we know (3.11) can be collectively rewritten as

$$\mathsf{L}_{\boldsymbol{u},T} \begin{pmatrix} \boldsymbol{w} \\ S \end{pmatrix} = \boldsymbol{A}_{\boldsymbol{u},T}, \tag{3.12}$$

where $\boldsymbol{A}_{\boldsymbol{u},T}$ captures the right-hand side of (3.11), and that $\mathsf{L}_{\boldsymbol{u},T}$ is regular. Thus, an appropriate correction $(\boldsymbol{w}, S)$ can indeed be found by Newton's method, see Section 1.3.5 (assuming we can find a suitable discretization, as (3.12) is infinite-dimensional).

This approach can be extended to the **continuation** of a **limit cycle branch** of a system

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \alpha), \ \boldsymbol{x} \in \mathbb{R}^n, \ \alpha \in \mathbb{R}, \tag{3.13}$$

with respect to (w.r.t.) parameter $\alpha \in \mathbb{R}$ as the solution to the following BVP ($\boldsymbol{u}_{\mathrm{old}}$ denotes the period-1 cycle for the previous value of α):

$$\left.\begin{aligned} \dot{\boldsymbol{u}}(s) - T\boldsymbol{f}(\boldsymbol{u}(s), \alpha) &= \boldsymbol{0}, \ s \in [0,1], \\ \boldsymbol{u}(0) - \boldsymbol{u}(1) &= \boldsymbol{0}, \\ \int_0^1 \langle \boldsymbol{u}(\sigma), \dot{\boldsymbol{u}}_{\mathrm{old}}(\sigma) \rangle \, \mathrm{d}\sigma &= 0. \end{aligned}\right\} \tag{3.14}$$

Furthermore, from Theorem 3.1 and the implicit function theorem, we obtain that a simple cycle has *locally unique* continuation w.r.t. parameter α. Again, we can define a corresponding operator to (3.14),

$$\begin{bmatrix} \nabla_{\boldsymbol{u}} - T\nabla_{\boldsymbol{u}}\boldsymbol{f}(\boldsymbol{u},\alpha) & -\boldsymbol{f}(\boldsymbol{u},\alpha) & -T\frac{\partial \boldsymbol{f}}{\partial \alpha}(\boldsymbol{u},\alpha) \\ \eta_0 - \eta_1 & 0 & 0 \\ \mathrm{Int}_{\boldsymbol{u}_{\mathrm{old}}} & 0 & 0 \end{bmatrix}, \tag{3.15}$$

which then has a one-dimensional null space for a simple cycle $\boldsymbol{u}$.

To solve (3.7) (or (3.14)) numerically, we must reduce it to a finite-dimensional problem (as opposed to searching in the infinite-dimensional space of periodic functions $\boldsymbol{u}$), that is, we need to choose a *discretization*. Although shooting, multiple shooting, and finite differences are sometimes used, the most common and most capable is the method of *orthogonal collocation*.

Consider for simplicity the BVP (3.7). We shall introduce a partitioning of the interval $[0,1]$ by $N-1$ mesh points, i.e.,

$$0 = s_0 < s_1 < \cdots < s_N = 1.$$

The primary goal of the orthogonal collocation is to approximate the true solution $\boldsymbol{u}$ by piecewise-differentiable continuous function, which is defined as a *vector polynomial* $\boldsymbol{u}^{(j)}(s)$ of maximal degree m within each subinterval $[s_j, s_{j+1}]$, $j = 0, 1, \ldots, N-1$. Such polynomials can be specified by m *collocation* points

$$s_j < \zeta_{j,1} < \zeta_{j,2} < \cdots < \zeta_{j,m} < s_{j+1}$$

belonging to each subinterval, where the approximate solution must satisfy the time-normalized ODE system (3.3) exactly. That is, we require

$$\left.\frac{\mathrm{d}\boldsymbol{u}^{(j)}}{\mathrm{d}s}\right|_{s=\zeta_{j,i}} = T\boldsymbol{f}(\boldsymbol{u}^{(j)}(\zeta_{j,i})) \tag{3.16}$$

for $i = 1, \ldots, m$, $j = 0, 1, \ldots, N-1$. To put it another way, we are trying to find N polynomial splines of degree m to approximate the true limit cycle. As the theory of splines tells us, it is advantageous to represent a given vector polynomial $\boldsymbol{u}^{(j)}$ by vectors of (unknown) solutions

$$\boldsymbol{u}^{j,k} = \boldsymbol{u}^{(j)}(s_{j,k}) \in \mathbb{R}^n, \ k = 0, 1, \ldots, m,$$

where $s_{j,k}$ are *representation points*[6], i.e., nodes of equidistant partitioning of the interval

$$s_j = s_{j,0} < s_{j,1} < \cdots < s_{j,m-1} < s_{j,m} = s_{j+1},$$

[6]Throughout the discussion of orthogonal collocation, we will use both *collocation* and *representation* points, which do not (in general) coincide.

implying $\boldsymbol{u}^{j,m} = \boldsymbol{u}^{j+1,0}$, given by

$$s_{j,k} = s_j + \frac{k}{m}(s_{j+1} - s_k),\ j = 0, 1, \dots, N-1, k = 0, 1, \dots, m.$$

This yields the following formulation of $\boldsymbol{u}^{(j)}(s)$ as interpolation between values at representation points,

$$\boldsymbol{u}^{(j)}(s) = \sum_{i=0}^{n} \boldsymbol{u}^{j,i} l_{j,i}(s), \tag{3.17}$$

where $l_{j,i}(s)$ are the *Lagrange basis polynomials* on $[s_j, s_{j+1}]$,

$$l_{j,i}(s) = \prod_{k=0, k\neq i}^{m} \frac{s - s_{j,k}}{s_{j,i} - s_{j,k}} \implies l_{j,i}(s_{j,k}) = \mathbb{I}_{i=k} = \begin{cases} 1, & i = k, \\ 0, & i \neq k. \end{cases}$$

Using (3.17), we can translate the problem of finding a vector polynomial (3.16) into a problem of determining the coefficients $\boldsymbol{u}^{j,i}$. Similarly, we can discretize the periodicity and phase conditions (3.4), (3.6), yielding respectively

$$\boldsymbol{u}^{0,0} = \boldsymbol{u}^{N-1,m} \tag{3.18}$$

and

$$\sum_{j=0}^{N-1} \sum_{i=0}^{m-1} \omega_{j,i} \left\langle \boldsymbol{u}^{j,i}, \dot{\boldsymbol{v}}^{j,i} \right\rangle + \omega_{N,0} \left\langle \boldsymbol{u}^{N,0}, \dot{\boldsymbol{v}}^{N,0} \right\rangle = 0, \tag{3.19}$$

where $\dot{\boldsymbol{v}}^{j,i}$ are the values of the derivative of the reference period-1 solution at representation points $s_{j,i}$ and $\omega_{j,i}$ are the *Gauss-Legendre quadrature coefficients*[7]. By combining (3.16), (3.17), (3.18), and (3.19) we get a system (3.20) of $nmN + n + 1$ scalar equations for the unknown values $\boldsymbol{u}^{j,i} \in \mathbb{R}^n$ at representation points $s_{j,i}$ (there are $(mN + 1)$ values $\boldsymbol{u}^{j,i}$) and for the unknown period T,

$$\left.\begin{aligned} \sum_{i=0}^{m} \boldsymbol{u}^{j,i} l'_{j,i}(\zeta_{j,k}) - T\boldsymbol{f}\left(\sum_{i=0}^{m} \boldsymbol{u}^{j,i} l_{j,i}(\zeta_{j,k})\right) &= \boldsymbol{0}_n, \\ j = 0, 1, \dots, N-1,\ k = 1, \dots, m, & \\ \boldsymbol{u}^{0,0} - \boldsymbol{u}^{N-1,m} &= \boldsymbol{0}_n, \\ \sum_{j=0}^{N-1} \sum_{i=0}^{m-1} \omega_{j,i} \left\langle \boldsymbol{u}^{j,i}, \dot{\boldsymbol{v}}^{j,i} \right\rangle + \omega_{N,0} \left\langle \boldsymbol{u}^{N,0}, \dot{\boldsymbol{v}}^{N,0} \right\rangle &= 0. \end{aligned}\right\} \tag{3.20}$$

The resulting finite-dimensional system (3.20) is nonlinear, thus we solve it by Newton's method (or some modification exploiting the block structure of Jacobian,

[7] Gauss-Legendre quadrature is a method of approximating a definite integral $\int_{-1}^{1} f(x)\,\mathrm{d}x \approx \sum_{i=1}^{n} \omega_i f(x_i)$. The optimal value of x_i and ω_i for a given value of n can be computed analytically. For more information, see [72]

which we will discuss later) — this problem is well-posed by Theorem 3.1 and the discussion afterward.

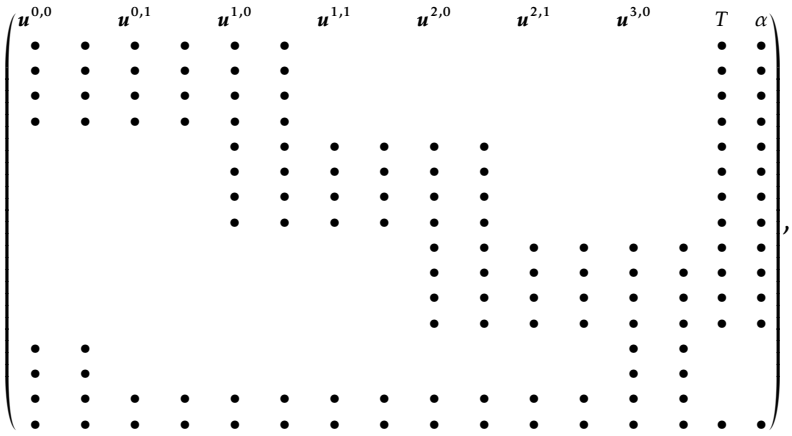
So far, we have translated (3.7) into a finite-dimensional problem, but we have neglected a discussion of a crucial choice of the collocation points $\{\zeta_{j,i}\}$. The distribution of collocation points affects the approximation, so a natural goal is to minimize its error. It can be shown that the optimal choice is to position them at so-called *Gauss points*, which are the roots of m-th degree *Legendre polynomial* corresponding to the subinterval $[s_j, s_{j+1}]$. For more information, we refer the reader to [72]. These roots originally belong to the interval $[-1, 1]$, but can be easily transformed to each $[s_j, s_{j+1}]$. Furthermore, the Legendre polynomials form an orthogonal system on the interval $[-1, 1]$ (or its transformation), which lends the name to the orthogonal collocation method.

Let us note the collocation at Gauss points leads to a very highly accurate approximation of a smooth solution of (3.7) by (3.20) at *mesh* points, namely

$$\left\| \boldsymbol{u}(s_j) - \boldsymbol{u}^{j,0} \right\| = O(h^{2m}),$$

where $h = \max_{0 \le j \le N-1}(s_{j+1} - s_j)$.

Let us turn our attention back to the discretized BVP system (3.20) and consider its continuation variant (i.e., the discretization of (3.14)), which can be written as $\boldsymbol{H}(\boldsymbol{X}) = \boldsymbol{0}$ for appropriate $\boldsymbol{H}$ and $\boldsymbol{X} = \left(\left\{\boldsymbol{u}^{j,k}\right\}, T, \alpha\right) \in \mathbb{R}^{mnN+n+2}$ corresponding to discretized $\boldsymbol{u}$, period T and the continuation parameter α. Its Jacobian matrix $\nabla \boldsymbol{H}$ reads (for $n = 2$, $N = 3$, and $m = 2$)[8]

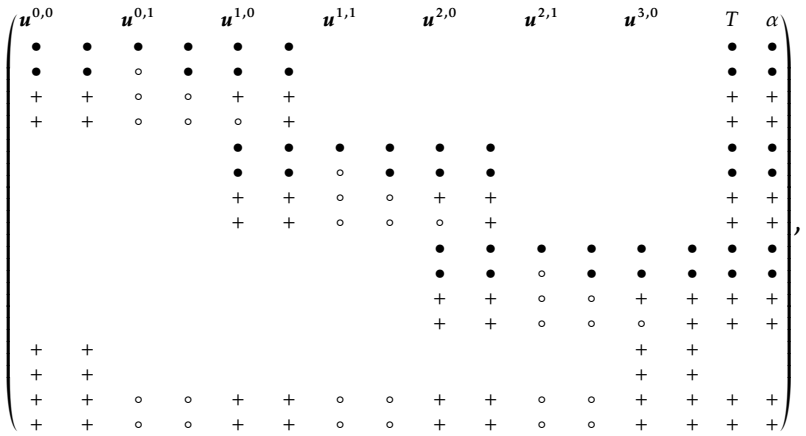


(3.21)

where • denotes a generally non-zero element. Thus $\nabla \boldsymbol{H}$ is sparse, and because it corresponds to the linear operator (3.15) it also has a one-dimensional null-space satisfying $\boldsymbol{H}(\boldsymbol{X}) = \boldsymbol{0}$ (at generic points). Let us note that by later also adding a constraint equation stemming from the chosen *continuation* method, see Section 3.2, we will get a fully determined system.

[8] The last row of $\nabla \boldsymbol{H}$ represents the continuation equation, which we have not yet discussed (and therefore, we assume, it can, in theory, depend on all variables).

The Jacobian matrix $\nabla \boldsymbol{H}$ is crucial for Newton's method, and its properties deeply affect the computation. Indeed, assuming the residual formulation of Newton's method (1.57), we need to be able to effectively solve the system $\nabla \boldsymbol{H} \cdot \boldsymbol{X} = \boldsymbol{R}$. Fortunately, in the ODE case, we can perform so-called *condensation of parameters*, see [73] and [74], eliminating the dependence on *intermediate variables* (i.e., $\boldsymbol{u}^{j,k}$ for $k = 1, \dots, m-1$) utilizing Gaussian elimination performed on rows. This yields ($\circ$'s mark eliminated elements)



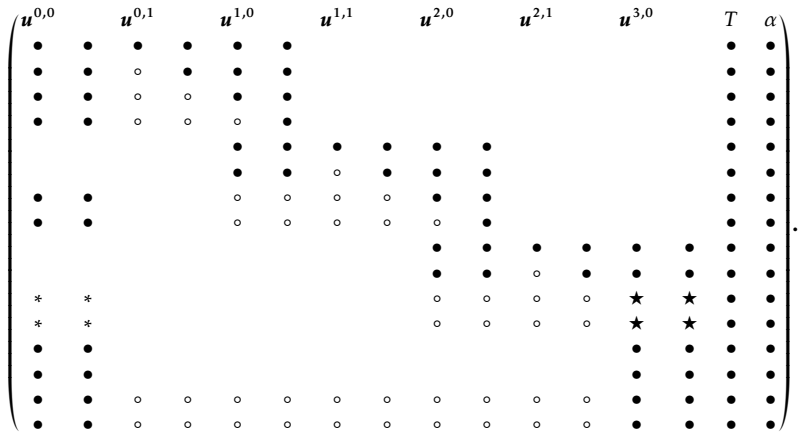
(3.22)

which has a *decoupled subsystem* for + signs. Hence, we can first solve the subsystem given by +'s, and follow up by solving for the intermediate variables afterward, see [74]. This significantly reduces the dimensionality and thus also memory requirements.

Assume that we have corrected (in, possibly, multiple steps of the Newton-Raphson method) $\boldsymbol{X} = \left(\left\{\boldsymbol{u}^{j,k}\right\}, T, \alpha\right)$ converging onto a new point

$$\boldsymbol{X}_c = \left(\left\{\boldsymbol{u}_c^{j,k}\right\}, T_c, \alpha_c\right)$$

on the branch of the limit cycle. Then $\boldsymbol{X}_c$ approximates a solution to (3.14). Moreover, we can further perform Gaussian elimination on rows of $\nabla \boldsymbol{H}(\boldsymbol{X}_c)$, see (3.22) and [70], to obtain



(3.23)

Furthermore, we denote by $\boldsymbol{P}_0$ the matrix block marked by $*$ sings and by $\boldsymbol{P}_1$ the block of $\bigstar$'s. Let $\boldsymbol{u}(0) = \boldsymbol{u}^{0,0}$ and $\boldsymbol{u}(1) = \boldsymbol{u}^{N,0}$, then the following holds

$$\nabla \boldsymbol{H}(\boldsymbol{X}_c) \cdot \left(\left\{\boldsymbol{u}^{j,k}\right\}, 0, 0\right)^\top = \boldsymbol{0} \implies \boldsymbol{P}_0\boldsymbol{u}(0) + \boldsymbol{P}_1\boldsymbol{u}(1) = \boldsymbol{0}.$$

A careful reader might notice that $\boldsymbol{P}_0$ and $\boldsymbol{P}_1$ belong to the linearized equation (3.16), i.e., by (3.8) the equality

$$\boldsymbol{u}(1) = -\boldsymbol{P}_1^{-1}\boldsymbol{P}_0\boldsymbol{u}(0) \tag{3.24}$$

approximates

$$\boldsymbol{u}(1) = \boldsymbol{\Gamma}(1)\boldsymbol{\Gamma}(0)^{-1}\boldsymbol{u}(0).$$

Thus, using $\boldsymbol{\Gamma}(0) = \boldsymbol{I}_n$ from (3.8), $-\boldsymbol{P}_1^{-1}\boldsymbol{P}_0 \approx \boldsymbol{\Gamma}(1) = \boldsymbol{M}(T_0)$ for the monodromy matrix $\boldsymbol{M}(T_0)$ corresponding to the cycle L_0 of (1.4). Computing the eigenvalues of $-\boldsymbol{P}_1^{-1}\boldsymbol{P}_0$ then estimates the multipliers of the cycle L_0, which determine its stability.

3.1.2 Semidynamical Systems

Now we shall turn our attention to semidynamical systems induced by DDEs. Throughout this section, we will assume the DDE of the form (1.24),

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{x}(t-\tau_1), \dots, \boldsymbol{x}(t-\tau_H)),$$

together with $0 = \tau_0 < \tau_1 < \cdots < \tau_H = \tau$. Although for the purposes of continuation, a dependence of $\boldsymbol{f}$ on some parameter $\alpha \in \mathbb{R}$ is assumed, here we consider α fixed and omit it from the notation. Let us note this section is primarily based on [75] and [69].

3.1.2.1 Equilibrium Localization

The techniques of equilibrium localization in DDEs do not differ much from the ODE case. Indeed, assume we are trying to find a stable equilibrium. By time integration, see Sections 1.2.2 and 1.3.2, we can reach a point $\boldsymbol{x}_t$ in a close neighborhood of the steady state solution. Then, we can apply the Newton-Raphson method, see Section 1.3.5, to the n-dimensional system

$$\boldsymbol{f}(\boldsymbol{x}^i, \dots, \boldsymbol{x}^i) = \boldsymbol{0}, \tag{3.25}$$

where we use $\boldsymbol{x}^0 = \boldsymbol{x}_t(0)$ as the initial approximation. The iterations of the Newton-Raphson method $\boldsymbol{x}^i$ converge to the steady state $\boldsymbol{x}^*$ for $\boldsymbol{x}^0$ sufficiently close to $\boldsymbol{x}^*$. Naturally, this process can be performed with free parameter α to obtain the continuation curve of the equilibrium by considering (3.25), along with a constraint given by the continuation method, see Section 3.2.

Suppose again we have found $\boldsymbol{x}^*$ with the desired accuracy and that our goal is now to determine its stability. It can be shown that the stability of $\boldsymbol{x}^*$ follows the stability of (the zero solutions of) the linearized equation, see (1.26),

$$\dot{\boldsymbol{x}}(t) = \sum_{i=0}^{H} \nabla_{\tau_i} \boldsymbol{f}^* \boldsymbol{x}(t-\tau_i), \tag{3.26}$$

where $\nabla_{\tau_i} \boldsymbol{f}^* = \nabla_{\tau_i} \boldsymbol{f}(\boldsymbol{x}^*, \dots, \boldsymbol{x}^*)$. Moreover, the linearized equation (3.26) is *asymptotically* stable if all roots $\lambda \in \mathbb{C}$ of the corresponding *characteristic equation*

$$\det\left(\lambda \boldsymbol{I} - \nabla_{\tau_0} \boldsymbol{f}^* - \sum_{i=1}^{H} \nabla_{\tau_i} \boldsymbol{f}^*\right) = 0 \tag{3.27}$$

lie in the open left half-plane, i.e., $\Re(\lambda) < 0$, see [27]. Unfortunately, the equation (3.27) has *infinitely many* roots. However, only *finitely many* roots have a real part larger than a given threshold. Thus, to study the stability of $\boldsymbol{x}^*$ it suffices to calculate only *stability-determining* roots $\lambda \in \mathbb{C}$, which are roots satisfying $\Re(\lambda) \geq r$ for a given threshold $r < 0$ close to zero.

Although analytical conditions for stability do exist (and are derived by techniques from complex analysis), we shall omit their theoretical derivation to maintain a focused and concise scope. Recently, there has been a development in numerical methods to approximate the stability-determining (i.e., right-most) characteristic roots of (3.27) using either a discretized *solution* operator $\mathsf{S}(t)$ of (3.26), see Section 1.2.3.2 and Definition 1.27, or a discretized *infinitesimal* generator A of the semigroup of the solution operator of (3.26), see Note 10.

The solution operator $\mathsf{S}(t)$ has eigenvalues $\mu(t)$, which are related to the roots of the characteristic equation by $\mu(t) = e^{\lambda t}$, see [75]. Clearly, to determine the stability of $\boldsymbol{x}^*$, we need to look at its dominant eigenvalues. Moreover, it follows that for t large, the eigenvalues $\mu(t)$ are well separated, which we can exploit in the computation. Below, we describe an algorithm to compute the dominant eigenvalues of $\mathsf{S}(\Delta t)$, where Δt is the time-step of a linear multistep (LMS) method.

> **Note 10: Infinitesimal generator**
>
> In Section 1.2.3.2, we have discussed that $\mathsf{S}(t)$ is a one-parameter *strongly continuous semigroup*, see [75]. One can thus define the *infinitesimal generator* A of $\{\mathsf{S}(t)\}$, see [31], by
>
> $$\mathsf{A}\boldsymbol{\phi} = \lim_{t \to 0^+} \frac{\mathsf{S}(t)\boldsymbol{\phi} - \boldsymbol{\phi}}{t}.$$
>
> In particular, for the linearized equation (3.26), the infinitesimal generator becomes
>
> $$\mathsf{A}\boldsymbol{x}(\theta) = \dot{\boldsymbol{x}}(\theta),$$
>
> $$\boldsymbol{x} \in \operatorname{dom} \mathsf{A} := \left\{ \boldsymbol{x} \in \mathbb{X}_C \mid \dot{\boldsymbol{x}} \in \mathbb{X}_C \;\&\; \dot{\boldsymbol{x}}(0) = \sum_{i=0}^{H} \nabla_{\tau_i} \boldsymbol{f}^* \boldsymbol{x}(-\tau_i) \right\}.$$
>
> Note that some algorithms for the computation of the stability of an equilibrium use a discretized infinitesimal generator instead of the solution operator.

When discretized, either of the discussed operators results in some matrix, for which we need to compute its eigenvalues. Therefore, it is our key concern that

this matrix is small or its dominant eigenvalues can be computed efficiently by using an iterative scheme, such as subspace iteration.

A straightforward way to approximate the solution operator is to consider the matrix form of the numerical integration of the linearized equation. Such an approach has been developed by Engelborghs et al., see [76], [75], and [69], and relies on the linear multistep method, see Section 1.3.2 and Algorithm 3.

Let us discretize the delay interval $[-\tau, 0]$ (typically extended slightly to both the left and right, as we shall discuss below) by an equidistant mesh of constant step-length h to approximate the solution operator $\mathsf{S}(h)$. Then, the solution $\boldsymbol{x}$ can be represented by a finite discrete set of points $\boldsymbol{x}^i := \boldsymbol{x}(t_i)$ with $t_i = ih$. A k-step LMS method for (3.26) with step-length h may be written as

$$\sum_{i=0}^{k} \alpha_i \boldsymbol{x}^i = h \sum_{i=0}^{k} \beta_i \cdot \left(\nabla_{\tau_0} \boldsymbol{f}^* \boldsymbol{x}^i + \sum_{j=1}^{H} \nabla_{\tau_i} \boldsymbol{f}^* \tilde{\boldsymbol{x}}(t_i - \tau_j) \right), \tag{3.28}$$

where $\alpha_i, \beta_i \in \mathbb{R}$ are parameters and $\tilde{\boldsymbol{x}}(t_i - \tau_j)$ are computed by polynomial interpolation with $s_- \in \mathbb{N}$ and $s_+ \in \mathbb{N}$ points to the left and right, respectively, when $(t_i - \tau_j)$ does not "align" with t_k from the discretization. To be precise, DDE-BIFTOOL uses so-called Nordsieck interpolation, see [69].

Clearly, the discretization of the operator $\mathsf{S}(h)$ itself is given by the linear map from $[\boldsymbol{x}^l, \ldots, \boldsymbol{x}^{k-1}]^\top$ to $[\boldsymbol{x}^{l+1}, \ldots, \boldsymbol{x}^k]^\top$, where $l = -s_- - \lceil \tau/h \rceil$ and where the mapping is prescribed by (3.28). This procedure results in a $N \times N$ matrix approximation of the operator $\mathsf{S}(h)$, where

$$N = n(-l + k) \approx n\tau/h. \tag{3.29}$$

Unfortunately, due to our choice of approximation and discretization of the solution operator $\mathsf{S}(h)$ at the step-length h, which is typically rather small, the eigenvalues $\mu(h)$ of $\mathsf{S}(h)$ are not well separated. Nonetheless, we can compute them using the QR method with computational cost $O(N^3)$, from which the roots of the characteristic equation can be calculated. Moreover, there have been derived conditions on the step-length h, such that all characteristic roots λ with $\Re(\lambda) \geq r$ for some threshold $r < 0$ are approximated sufficiently accurately, see [76].

> **Note 11: Runge-Kutta discretization**
>
> Let us note a discretization induced by the Runge-Kutta numerical integrator can also be used, albeit it is less frequent in practice. We refer the interested reader to [77].

3.1.2.2 Periodic Orbit Localization

The localization of periodic orbits in DDEs proceeds similarly to the ODE case, see Section 3.1.1.2. Indeed, the main differences lie in the discretization and

normalization to unit period. Let us remark this subsection is based on [78], [69] and [75], but also [79], [80], and [81].

So far, our discretization scheme, prescribed by *representation* points,

$$0 = s_0 = s_{0,0} < s_{0,1} < \cdots < s_{0,m-1} < s_{0,m} = s_1 = s_{1,0} < \cdots < s_N = T,$$

has been designed for a single period interval $[0, T]$ (or $[0, 1]$ after period-1 normalization). As such, we can periodically extend it to cover the interval $[-\tau, T]$. There are now two possible, yet fundamentally different approaches.

The first approach, used in DDE-BIFTOOL, see [69] or [75], normalize the system in time to a unit period for the given cycle (analogously as we did for ODEs). Afterward, we can wrap around the delays into interval $[0, 1]$, which yields the following BVP[9]

$$\left.\begin{aligned} \dot{\boldsymbol{u}}(\zeta_{j,i}) - T\boldsymbol{f}\left(\boldsymbol{u}(\zeta_{j,i}), \boldsymbol{u}\left(\left(\zeta_{j,i} - \frac{\tau_1}{T}\right) \bmod [0,1]\right), \dots, \boldsymbol{u}\left(\left(\zeta_{j,i} - \frac{\tau_H}{T}\right) \bmod [0,1]\right)\right) &= \boldsymbol{0}, \\ j = 0, 1, \dots, N-1,\ k = 1, \dots, m, & \\ \boldsymbol{u}(0) - \boldsymbol{u}(1) &= \boldsymbol{0}, \\ \int_0^1 \langle \boldsymbol{u}(\sigma), \dot{\boldsymbol{u}}_{\mathrm{old}}(\sigma) \rangle \, \mathrm{d}\sigma &= 0, \end{aligned}\right\} \tag{3.30}$$

where $\zeta_{j,i}$ are again the *collocation* points, and $\boldsymbol{u}_0$ and $\boldsymbol{u}_1$ are solution segments on $[-\tau, 0]$ and $[1 - \tau/T, 1]$, respectively. The period T is an unknown variable. Unfortunately, the resulting linearized system for the BVP (3.30) has a banded structure, which cannot be easily exploited to aid the used linear solver. Moreover, the transformation via the modulo operator complicates the computation of Floquet multipliers, because it distorts the included matrix corresponding to the discretized time-integration operator.

The alternative method, see [78] and [75], relies on the construction of the orthogonal collocation BVP *without* subsequent wrapping of the delays. This produces a larger BVP, as collocation of the delayed segment is also required,

$$\left.\begin{aligned} \dot{\boldsymbol{u}}(\zeta_{j,i}) - T\boldsymbol{f}\left(\boldsymbol{u}(\zeta_{j,i}), \boldsymbol{u}\left(\zeta_{j,i} - \frac{\tau_1}{T}\right), \dots, \boldsymbol{u}\left(\zeta_{j,i} - \frac{\tau_H}{T}\right)\right) &= \boldsymbol{H}^{\mathrm{dde}}(\boldsymbol{u}, T) = \boldsymbol{0}, \\ j = 0, 1, \dots, N-1,\ k = 1, \dots, m, & \\ \boldsymbol{u}_0 - \boldsymbol{u}_1 = \boldsymbol{H}^{\mathrm{per}}(\boldsymbol{u}) &= \boldsymbol{0}, \\ \int_0^1 \langle \boldsymbol{u}(\sigma), \dot{\boldsymbol{u}}_{\mathrm{old}}(\sigma) \rangle \, \mathrm{d}\sigma = \boldsymbol{\Phi}[\boldsymbol{u}] &= 0 \end{aligned}\right\} = \boldsymbol{H}(\boldsymbol{u}, T) = \boldsymbol{0}. \tag{3.31}$$

Note that the trajectory $\boldsymbol{u}(s)$ of the periodic orbit for $s \in [-\tau/T, 1]$ may be split into 2 parts, $\boldsymbol{u}_0$ corresponding to the initial point, which captures the delay, and

[9] The notation $t \bmod [0,1]$ refers to $t - \max k \in \mathbb{Z} \mid k \le t$.

$\boldsymbol{u}_t(s)$ for $s \in [0,1]$ representing the cycle itself. To solve BVP (3.31), one can again employ the Newton-Raphson method in its residual form, see Section 1.3.5, where its linearization may be written as

$$\left.\begin{aligned} \nabla_{\boldsymbol{u}_0}\boldsymbol{H}^{\mathrm{dde}}\Delta\boldsymbol{u}_0 + \nabla_{\boldsymbol{u}_t}\boldsymbol{H}^{\mathrm{dde}}\Delta\boldsymbol{u}_t + \nabla_T\boldsymbol{H}^{\mathrm{dde}}\Delta T &= -\boldsymbol{H}^{\mathrm{dde}}, \\ \Delta\boldsymbol{u}_0 - \Delta\boldsymbol{u}_1 &= -\boldsymbol{H}^{\mathrm{per}}, \\ \nabla_{\boldsymbol{u}_0}\boldsymbol{\Phi}\Delta\boldsymbol{u}_0 + \nabla_{\boldsymbol{u}_t}\boldsymbol{\Phi}\Delta\boldsymbol{u}_t + \nabla_T\boldsymbol{\Phi}\Delta T &= \boldsymbol{\Phi}. \end{aligned}\right\} \tag{3.32}$$

All functions and their partial derivatives are assumed to be evaluated for a given $(\boldsymbol{u}, T)$, such that $\Delta\boldsymbol{u}_0, \Delta\boldsymbol{u}_t, \Delta T$ are considered to be correction variables for (3.32) in the context of the Newton-Raphson method (this notation was introduced in Section 1.3.5).

> **Tip 7: Similarities with ODEs**
>
> By omitting terms corresponding to partial derivatives with respect to $\boldsymbol{u}_0$, we get a similar expression to the linearized BVP for the ODE case (3.11).

In [78], Verheyden and Lust proposed a method of condensation of parameters for (3.32), such that the BVP is reduced to the following form

$$\begin{pmatrix} \boldsymbol{\Gamma}(1) - \boldsymbol{I} & * \\ * & * \end{pmatrix}\begin{pmatrix} \Delta\boldsymbol{u}_0 \\ \Delta T \end{pmatrix} = -\boldsymbol{R}, \tag{3.33}$$

where $\boldsymbol{\Gamma}(1)$ is the monodromy matrix, see (3.8), which can be obtained from the time-integration matrix $\boldsymbol{\Gamma}_t$ for variational equations[10],

$$\boldsymbol{\Gamma}_t = -\left(\nabla_{\boldsymbol{u}_t}\boldsymbol{H}^{\mathrm{dde}}\right)^{-1}\nabla_{\boldsymbol{u}_0}\boldsymbol{H}^{\mathrm{dde}}, \tag{3.34}$$

similarly to the ODE case, see (3.24). The exact form of (3.33) can be found in [78]. The Newton-Picard method, see [82], [83] or [84], can be used to efficiently solve (3.33). Afterward, $\Delta\boldsymbol{u}_t$ may be obtained from the first equation of (3.32). Lastly, Floquet multipliers of the periodic orbit can be computed as the dominant eigenvalues of the discretized monodromy matrix $\boldsymbol{\Gamma}(1)$.

3.2 Continuation

Let us now explore the *continuation* process itself (for references, see [13] and [85]). In general, a (finite-dimensional) *continuation problem* is the task of finding a curve in $\mathbb{R}^{n+1}$ prescribed by system

$$\boldsymbol{H}(\boldsymbol{X}) = \boldsymbol{0}, \ \boldsymbol{H} : \mathbb{R}^{n+1} \to \mathbb{R}^n. \tag{3.35}$$

[10] The variational equations for a given cycle were introduced in Definition 1.21 and correspond to the linearization of the orthogonal collocation BVP.

As an example, one might consider the *equilibrium curve*, which is given as the α-parametrized solution of $\boldsymbol{f}(\boldsymbol{x}, \alpha) = \boldsymbol{0}$ for a (semi)dynamical system

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \alpha).$$

All of the continuation problems we shall be interested in arise from parametrized *localization problems*, see the previous section 3.1.

From the *implicit function theorem* follows, that the system (3.35) locally defines a smooth one-dimensional manifold (*curve*) $\mathscr{M}$ passing through a point $\boldsymbol{X}^0$, if $\operatorname{rank} \mathrm{J}_{\boldsymbol{H}} = n$, where $\mathrm{J}_{\boldsymbol{H}} = \nabla \boldsymbol{H}(\boldsymbol{X}^0)$. By a numerical solution to (3.35) is understood a sequence of points $\boldsymbol{X}^0, \boldsymbol{X}^1, \ldots$ approximating the curve $\mathscr{M}$ with the desired accuracy. The continuation algorithms we shall present here will be based on the *prediction–correction* process.

For the prediction part, we will solely use the *tangent* prediction

$$\tilde{\boldsymbol{X}}^i_{(0)} = \boldsymbol{X}^i + h\boldsymbol{v}^i, \tag{3.36}$$

where $\boldsymbol{v}^i \in \mathbb{R}^{n+1}$ is the normalized tangent vector to the curve $\mathscr{M}$ at the (regular) point $\boldsymbol{X}^i$ and $h \in \mathbb{R}$ is the stepsize. Using the arclength parametrization of $\boldsymbol{X}(s)$ of the curve near $\boldsymbol{X}^i$ (i.e., $\boldsymbol{X}(0) = \boldsymbol{X}^i$), we can substitute it into (3.35) and differentiate with respect to s, which yields

$$\nabla \boldsymbol{H}(\boldsymbol{X}^i) \overbrace{\dot{\boldsymbol{X}}(s)\big|_{s=0}}^{\boldsymbol{v}^i} = \boldsymbol{0}.$$

In other words, $\boldsymbol{v}^i$ must lie in the null space of the matrix of local linearization of $\mathscr{M}$ around $\boldsymbol{X}^i$. Moreover, since $\boldsymbol{X}^i$ is presumed to be *regular*, i.e., $\operatorname{rank} \nabla \boldsymbol{H}(\boldsymbol{X}^i) = n$, then $\boldsymbol{v}^i$ is determined uniquely up to a constant multiple.

> **Tip 8: Secant prediction**
>
> Although we have introduced only the tangent prediction, another popular method is the *secant prediction*. For more information, one can see [13].

Currently, we have predicted a point $\tilde{\boldsymbol{X}}^i_{(0)}$ assumed to be close to the curve, which we need to "*correct*" such that it lies on $\mathscr{M}$ within a certain specified degree of accuracy. This, in general, requires the use of a Newton-like method, see Section 1.3.5, as (3.35) might be highly nonlinear. However, Newton-like methods can only be applied to fully determined systems, i.e., the number of equations must correspond to the number of unknown variables. Therefore, we add an extra scalar (remember $\boldsymbol{H} : \mathbb{R}^{n+1} \to \mathbb{R}^n$) condition

$$g^i(\boldsymbol{X}) = 0$$

to (3.35), producing

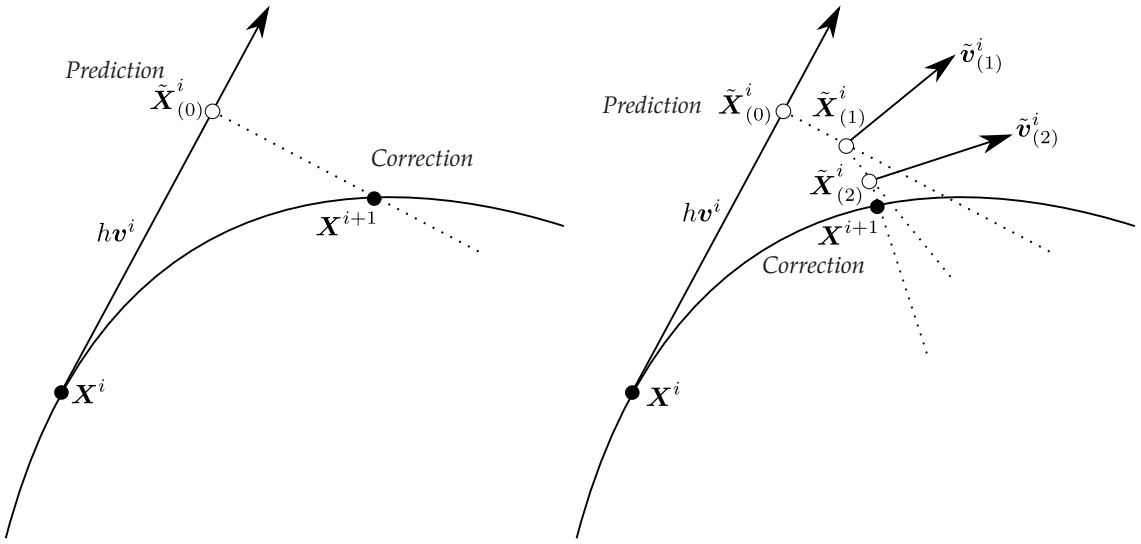
$$\left.\begin{aligned} \boldsymbol{H}(\boldsymbol{X}) &= \boldsymbol{0}, \\ g^i(\boldsymbol{X}) &= 0. \end{aligned}\right\} \tag{3.37}$$

Finally, Newton's method can be used on (3.37).

One possibility is to select a hyper-plane P^i passing through the point $\tilde{\boldsymbol{X}}^i_{(0)}$, such $\boldsymbol{v}^i$ is its normal vector. Then, we can find the corrected point $\boldsymbol{X}^{i+1}$ as the intersection of P^i and the curve $\mathscr{M}$. Further, any point $\boldsymbol{X}$ on P^i must satisfy

$$\overbrace{\left\langle \boldsymbol{X} - \tilde{\boldsymbol{X}}^i_{(0)}, \boldsymbol{v}^i \right\rangle}^{g^i(\boldsymbol{X})} = 0, \tag{3.38}$$

which defines $g^i : \mathbb{R}^{n+1} \to \mathbb{R}$. When combined with the tangent prediction, this method is called the *pseudo-arclength continuation*, see Figure 3.2a.



(a) Pseudo-arclength continuation (b) Moore-Penrose continuation

Figure 3.2: Overview of different prediction–correction methods for continuation (adapted from [85]).

Another possible approach is to adapt $g^i(\boldsymbol{X})$ throughout the correction process, i.e., in the course of the Newton iterations. This produces a sequence of *scalar continuation constraints* $\left\{g^i_{(k)}\right\}_{k=0}^{N_i}$, with $g^i_{(0)} \equiv g^i$, such that after the k-th iteration of the Newton's method, producing $\tilde{\boldsymbol{X}}^i_{(k)}$, we update the restricting hyper-plane to $P^i_{(k)}$ defined as

$$g^i_{(k+1)}(\boldsymbol{X}) = \left\langle \boldsymbol{X} - \tilde{\boldsymbol{X}}^i_{(k)}, \tilde{\boldsymbol{v}}^i_{(k)} \right\rangle = 0,$$

where $\tilde{\boldsymbol{v}}^i_{(k)} \neq \boldsymbol{0}$ lies in the null-space of $\nabla \boldsymbol{H}(\tilde{\boldsymbol{X}}^i_{(k)})$, see (3.38). This method is commonly referred to as the *Moore-Penrose* continuation method, see Figure 3.2b for illustration.

3.3 Cycle Continuation in Coupled Oscillators

As we have discussed in Section 2.1 on the problem formulation, we shall again consider the problem of computing phase shifts between 2 weakly coupled oscil-

lators with possible delay in the coupling. Let us disclose this section is heavily based on the article by Záthurecký et al. [56].

For the soundness of the grid-based approach to the computation of phase shift between the oscillators of (2.1), we have simply used the continuous dependence on parameters and initial conditions. Unfortunately, this argument is insufficient for the cycle-continuation-based approach, as nothing a priori guarantees the (local) existence of a suitable (stable) cycle, which could be used for continuation at an arbitrary coupling strength $\lambda > 0$.

Notably, Záthurecký et al. [56] have shown that there must locally exist a cycle manifold $\mathscr{M}$ (with respect to coupling strength and an arbitrary parameter affecting the frequency of one oscillator) near zero coupling between the two nearly identical oscillators. In particular, we shall focus primarily on the anti-phase synchronization, which may be used to explain the emergence of frequencies higher than physiological, see [5] and [4]. This will be performed mainly by studying the phase shifts between the two oscillators.

Recall the studied system (2.1),

$$\begin{aligned}\dot{\boldsymbol{x}}_1(t) &= \boldsymbol{f}(\boldsymbol{x}_1(t);\omega_1) + \lambda \boldsymbol{K}(\boldsymbol{x}_1(t), \boldsymbol{x}_2(t-\tau), \sigma),\\ \dot{\boldsymbol{x}}_2(t) &= \boldsymbol{f}(\boldsymbol{x}_2(t);\omega_2) + \lambda \boldsymbol{K}(\boldsymbol{x}_1(t-\tau), \boldsymbol{x}_2(t), \sigma),\end{aligned}$$

describing λ-weakly coupled oscillators $\boldsymbol{x}_1(t), \boldsymbol{x}_2(t)$. In particular, we shall again concern ourselves with the case of two weakly coupled interneuron models, see (1.63).

This methodology now enables us to study Arnold tongues corresponding to different shifts. In particular, the continuation process computes the entire cross-section of manifold $\mathscr{M}$ at a given λ_i. This can be represented as a curve $\mathscr{C}_{\lambda_i}$, for instance, in 3-dimensional space (λ, C_2, T). Projecting parts of $\mathscr{C}_{\lambda_i}$, where the corresponding cycle is stable, to the (λ, C_2) parameter plane yields a slice (with fixed λ_i) of all Arnold tongues of $\mathscr{M}$ as present in $\mathscr{C}_{\lambda_i}$. This very fact is obscured by the simulation approach. Indeed, if two submanifolds of $\mathscr{M}$ corresponding to stable underlying cycles overlap when projected onto the parameter space, i.e., they form 2 overlapping Arnold tongues, the simulation approach will highlight only the one in the basin of attraction of the cycle with a given shift (if both are stable).

Let us begin with $\lambda = 0$, i.e., decoupled nearly-identical oscillators. Then any shift may occur between them (as there is effectively no connection) and from the theoretical contributions of [56], we know that an in-phase and anti-phase synchrony must exist for cycles on $\mathscr{M}$ close to 0- and $T/2$-shifts for small enough coupling (i.e., the distortion from the coupling will not prevent phase synchronization). Therefore, we choose a fixed discretization $\Lambda = \{\lambda_i\}_{i=0}^{N_\Lambda}$ for λ.

For each $\lambda_i \in \Lambda$, we let the system converge to a stable cycle from a fixed initial condition $\boldsymbol{\phi}$. After successful convergence, we then continue it with respect to C_2, which yields a slice of the unknown cycle manifold $\mathscr{M}$.

> **Note 12: Discretization of λ**
>
> Note that currently, we have "blindly" discretized λ, and then for each element $\lambda_i \in \Lambda$ of the discretization, we have attempted to converge onto a stable periodic orbit. Only afterward could we compute the continuation with respect to C_2. Another approach would be first to continue the cycle with respect to λ, producing a curve $\mathscr{C}_\lambda$ in the desired manifold $\mathscr{M}$. Then, we could perform the continuation with respect to C_2 for each point of $\mathscr{C}_\lambda$, which is already discretized due to the nature of numerical continuation.

Moreover, for each cycle on the C_2-continuation curve, we know its corresponding period T. This allows us to use shift searching methods, see Section 2.2.2, with much better accuracy (compared to only estimated period). In particular, we partition the interval $[0,T]$, and for each of the disjunct subintervals $Q_i \subseteq [0,T]$, we run the shift searching method. Finally, we select the shift with the least error.

In addition, when there is no delay in the coupling $\boldsymbol{K}$ (i.e., the ODE case), one can also continue the LPC curves with respect to λ and C_2. Note that these curves do not always originate from the zero coupling, as can be seen in Figures 3.3 and 3.4 (parts (C) and (D)). Unfortunately, DDE-BIFTOOL does not support the continuation of LPC curves. Nevertheless, we may attempt to estimate these curves from the boundaries of stable and unstable limit cycles on the manifold $\mathscr{M}$.

Now, we can show the relation of λ and C_2 to both the period T and the shift β, see Figures 3.3 and 3.4 (parts (A) and (B)). Furthermore, we can project the cycle manifold onto the parameter plane (C_2, λ), where it draws out desired tongues. Let us remark tongues corresponding to different stable sub-manifolds of $\mathscr{M}$ can overlap.

Figures 3.3 and 3.4 show this methodology applied to two weakly coupled interneuron models, see (1.63). The continuation was performed using MatCont, see [67] and [68], in the ODE case and DDE-BIFTOOL, see [69], in the DDE case. In particular, we calculated the *optimal shift* in the first coordinate of both oscillators over a $3T$-long trajectory. The region around zero-delay is numerically unstable for the continuation, see Section 3.1.2 — for very small delay (notice that the delay affects only the coupling term), the DDE system approaches the behavior of an ODE, resulting in a bad condition of the Jacobian matrix for the orthogonal collocation/continuation. For implementation details, see the Git repository for [56], potentially also the Git repository associated with this thesis for specifics of plotting computed data.

The runtimes necessary to generate data for Figures 3.3 and 3.4 (parts (A) and (B)) are approximately 43 *seconds* for the ODE case and 26 *minutes* for the DDE version, respectively. Note that although parts (C) and (D) of the same figures require a finer discretization Λ, in practice one would probably not need to use it. Indeed, there is no inherent increase in error when choosing a sparser discretization, whereas, in the simulation-based approach, it would affect the entire result. As

there is no dependence on results from other slices throughout the calculation, the parallelization is much easier and more efficient[11]. The required runtime in MATLAB, see [37], in the DDE case, is heavily dependent on the version of the dde23 solver, as discussed in Tip 3.

While there exists a Julia package DDEBifurcationKit.jl for continuation in DDEs, see [86], it currently does not support the computation of Floquet multipliers for periodic orbits, see this GitHub issue for more details. For completeness' sake, we provide a reference single-threaded unoptimized implementation using this package as well, which can be found in the Git repository for [56]. The data generation process described above takes about 3 minutes for an ODE case (on 4 threads).

Finally, Figure 3.5 depicts the comparison of both approaches. Specifically, we plot the heatmaps of shifts β computed on MetaCentrum using the simulation-on-grid approach. Moreover, black curves render the boundaries of anti-phase Arnold tongues calculated with the bifurcation-based approach. In total, one can see that *in general* these two methods give comparable results. However, the simulation-based approach suffers from a rather high artifact rate, even though we used a very dense underlying grid.

[11] The implementation of the simulation-on-grid relies on using clever iteration order and multi-threading locks, which results in delays when a thread is waiting to write to or read from the results matrices.

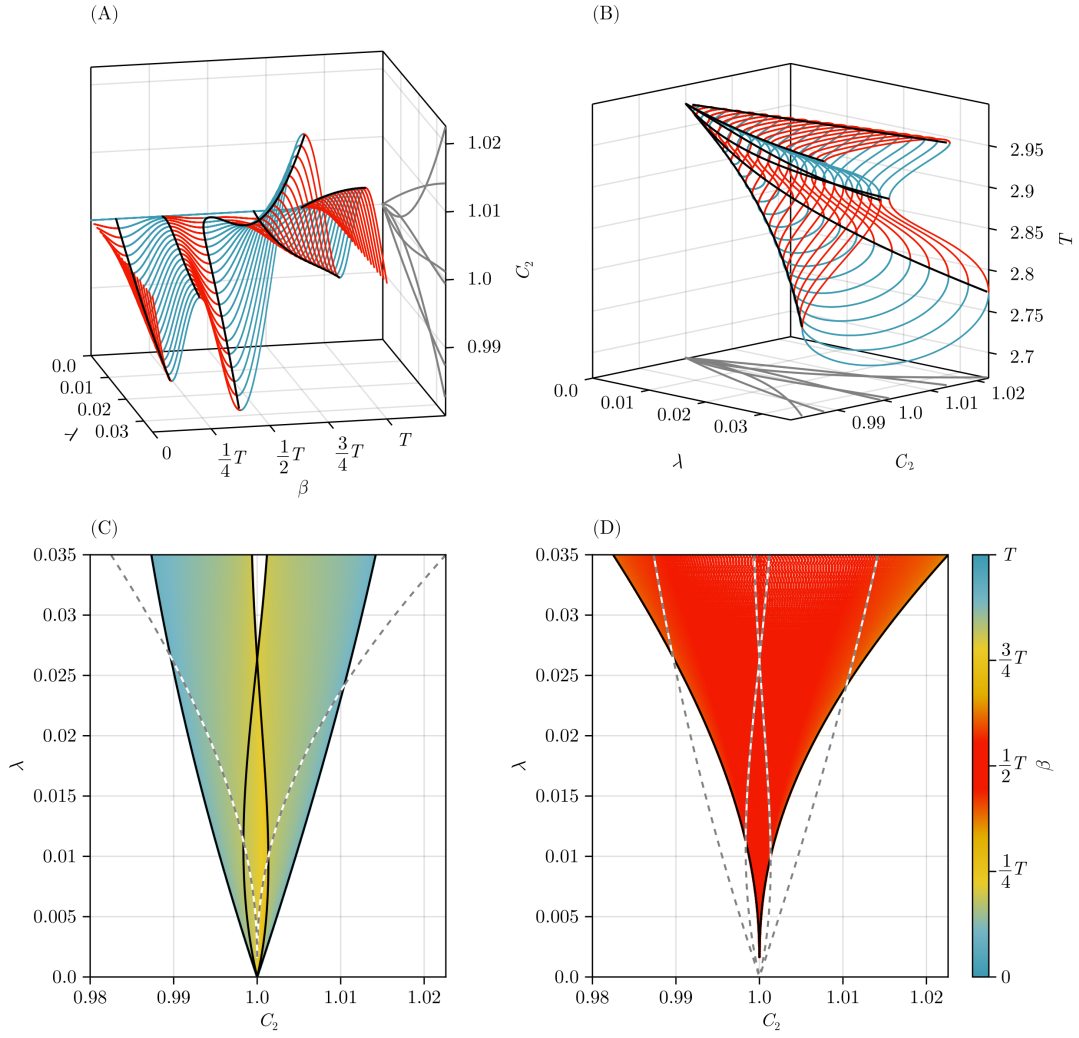


Figure 3.3: Numerical continuation of the cycle manifold $\mathscr{M}$ of the system (1.63) without delay for $C_1 = 1$ and free parameter C_2 using MatCont is shown in (A); (B) plots period T with respect to λ and C_2. The colors of the continued section curves indicate stability (blue) or instability (red) of the limit cycles. The grey curves in panels (A) and (B) represent the projections of the black LPC curves onto (C_2, λ) delineating the Arnold tongues. The heatmaps of (C) the in-phase and (D) the anti-phase Arnold tongues illustrate the corresponding phase shift β. Adapted from [56] (redrawn in Julia using data from the article).

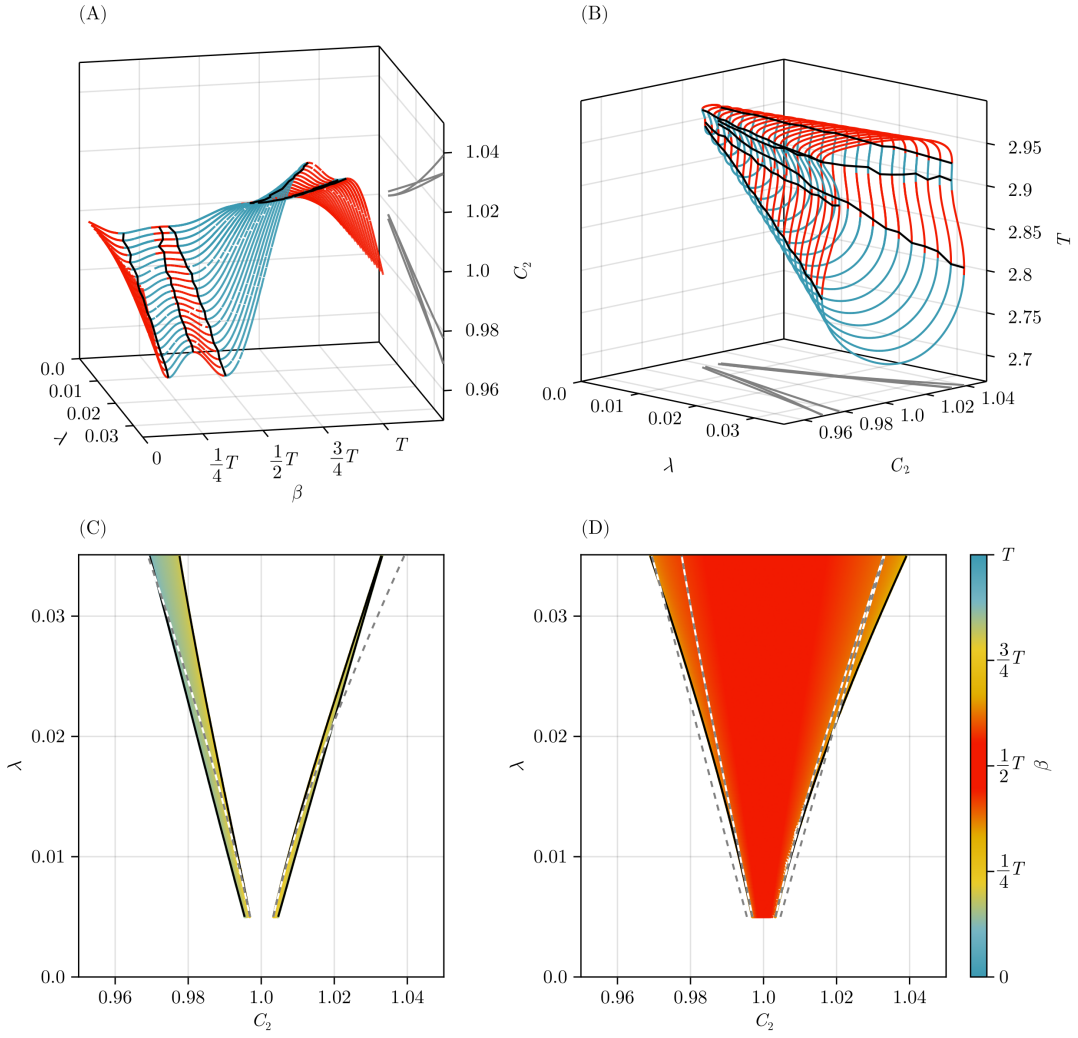
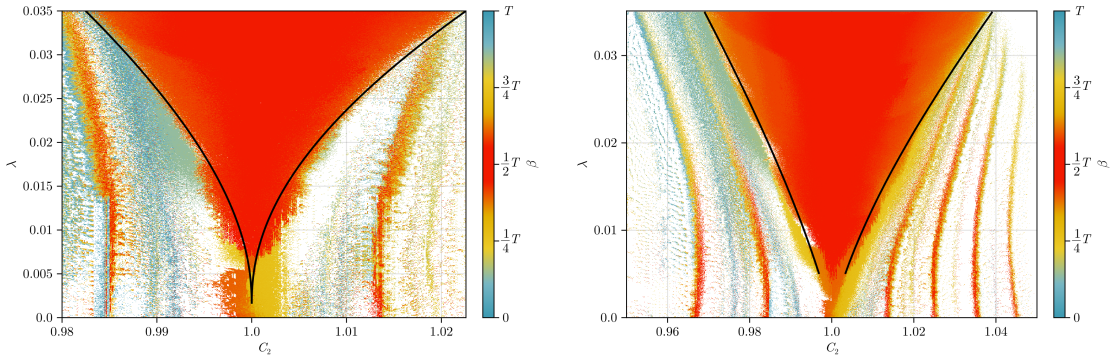


Figure 3.4: Numerical continuation of the cycle manifold $\mathscr{M}$ of the system (1.63) with delay $\tau = 0.025$ for $C_1 = 1$ and free parameter C_2 using DDE-BIFTOOL is shown in (A); (B) plots period T with respect to λ and C_2. The colors of the continued section curves indicate stability (blue) or instability (red) of the limit cycles. The grey curves in panels (A) and (B) represent the projections of the black estimated LPC curves onto (C_2, λ) delineating the Arnold tongues. The heatmaps of (C) the in-phase and (D) the anti-phase Arnold tongues illustrate the corresponding phase shift β. Adapted from [56] (redrawn in Julia using data from the article).



(a) $\tau = 0$ (b) $\tau = 0.025$

Figure 3.5: Comparison of heatmaps of phase shifts β computed using the simulation-based approach, see Chapter 2 and Figure 2.8, and borders of anti-phase Arnold tongues obtained from the continuation of the cycle manifold $\mathscr{M}$, see above.

Conclusions

Phase-locked synchronization is of utmost importance for neuroscience. Specifically, in-phase synchronization refers to the phase relationship between two coupled neurons firing simultaneously, whereas anti-phase synchronization refers to the phase relationship between two coupled neurons that maximizes the frequency of their combined signal. However, the problem of determining suitable parameter values, which would result in synchronization, and the phase shift thereof, is a rather complicated task.

After building up the necessary theoretical background in the first chapter, we have delved into phase-shift detection using a grid simulation approach. Iteratively, we built and improved the methodology for the problem at hand, introducing various concepts along the way. Moreover, at each step, we applied the algorithm to a system of two weakly coupled interneuron models. Finally, the computation was executed on the MetaCentrum grid infrastructure.

In the third chapter, we discussed the localization of equilibria and limit cycles for both ODE and DDE systems. Then, we applied the showcased numerical methods to the system of interest, obtaining the cycle manifold corresponding to an Arnold tongue for a given synchronization. We were then able to compute phase shifts (between the two oscillators) of periodic orbits on the manifold. By projection onto the parameter plane, we obtained a comparable result to the more traditional grid simulation.

It is rather complicated to say which method is "better", as each takes a very different route on the path to the same goal. The simulation-based approach more closely mimics (under the assumption of an appropriate *iterator*) a continuous change of parameters, which may or may not have the time to converge to the equilibrium. This renders the choice of transient interesting for applications because the method can compute synchronization modes the system can reach by continuously changing its parameters. Moreover, its strengths lie in the fact that much less theoretical background is needed, making it more accessible to a wider (academic) community.

Nevertheless, at its core, it is still a heuristic algorithm, which we have attempted to improve in various ways. Its biggest drawback probably lies in the dependence on nearby already computed trajectories and phase shifts for a given element of the

grid. This very idea, which we have used to improve the Arnold tongue detection, complicates multi-threading execution and necessitates the use of a dense grid to allow us to draw reasonable conclusions. Lastly, while the initial draft of the method was rather easy to implement, each subsequent iteration makes it more complicated and introduces possible parameters and choices of methods/norms/etc. For instance, in the current version of *GridWalker.jl*, there are approximately 40 parameters/functions strictly related to the computation itself to be selected.

On the other hand, the approach based on bifurcation theory streamlines many parts of the process at the cost of much higher theoretical complexity of the underlying methods. Indeed, save for the shift computation, most of the algorithm uses standard continuation methods available in bifurcation toolkits, such as MATCONT or DDE-BIFTOOL. This also significantly reduces the number of parameters or selectable functions. Furthermore, the ability to easily and efficiently parallelize the computation compensates for the higher time complexity of the calculation at each individual point. Finally, the theoretical background and independence of single slice computation from the previous results guarantee that once we have computed a phase shift for a given point (C_2^i, λ^i), it must hold for any discretization Λ such that $\lambda^i \in \Lambda$. This crucial property is missing in the simulation-based approach.

In total, we believe the bifurcation-based approach should be more prevalent in applied fields, as it offers clear computational benefits. Nevertheless, some capabilities of the grid simulation approach, for instance, stochastic phenomena or the effect of transient, cannot be replicated by using the continuation-based approach. Thus, researchers should strive to pick the best methodology for their problem of interest.

Bibliography

[1] Willms, A. R., Kitanov, P. M. and Langford, W. F. (2017). Huygens' clocks revisited. *Royal Society Open Science* **4** 170777.

[2] Brázdil, M., Worrell, G. A., Trávníček, V., Pail, M., Roman, R., Plešinger, F., Klimeš, P., Cimbálník, J., Stacey, W. and Jurák, P. (2023). Ultra fast oscillations in the human brain and their functional significance.

[3] Cimbalnik, J., Brinkmann, B., Kremen, V., Jurak, P., Berry, B., Gompel, J. V., Stead, M. and Worrell, G. (2018). Physiological and pathological high frequency oscillations in focal epilepsy. *Annals of Clinical and Translational Neurology* **5** 1062–76.

[4] Ševčík, J. and Přibylová, L. (2025). Cycle multistability and synchronization mechanisms in coupled interneurons: In-phase and anti-phase dynamics under current stimuli. *accepted for publication in Applied Mathematics and Computation*.

[5] Přibylová, L., Ševčík, J., Eclerová, V., Klimeš, P., Brázdil, M. and Meijer, H. G. E. (2024). Weak coupling of neurons enables very high-frequency and ultra-fast oscillations through the interplay of synchronized phase shifts. *Network Neuroscience* **8** 293–318.

[6] Přibylová, L., Ševčík, J., Halmazňa, T., Husa, Š., Malárik, P., Kajanová, L., Polách, M., Zapadlo, Š. and Eclerová, V. (2025). Chaos links dendritic calcium to bursting in hippocampal pyramidal cells. *Chaos, Solitons and Fractals*.

[7] Jiruska, P., Alvarado-Rojas, C., Schevon, C. A., Staba, R., Stacey, W., Wendling, F. and Avoli, M. (2017). Update on the mechanisms and roles of high-frequency oscillations in seizures and epileptic disorders. *Epilepsia* **58** 1330–9.

[8] Pikovsky, A., Rosenblum, M. and Kurths, J. (2001). *Synchronization: A universal concept in nonlinear sciences*. Cambridge University Press.

[9] Halaštová, B. (2025). *Analýza signálů v neurovědách*. Master's thesis, Masarykova univerzita, Přírodovědecká fakulta, Brno.

[10] Allaire, J. J., Teague, C., Scheidegger, C., Xie, Y., Dervieux, C. and Woodhull, G. (2025). Quarto.

[11] Bezanson, J., Edelman, A., Karpinski, S. and Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review* **59** 65–98.

[12] Danisch, S. and Krumbiegel, J. (2021). Makie.jl: Flexible high-performance data visualization for Julia. *Journal of Open Source Software* **6** 3349.

[13] Kuznetsov, Y. A. (2023). *Elements of applied bifurcation theory*. Springer International Publishing.

[14] Přibylová, L. (2021). *Teorie bifurkací, chaos a fraktály*. Masarykova univerzita.

[15] Teschl, G. (2012). *Ordinary differential equations and dynamical systems*. American Mathematical Society, Providence, RI.

[16] Guo, S. and Wu, J. (2013). *Bifurcation theory of functional differential equations*. Springer New York.

[17] Saperstone, S. H. (1981). *Semidynamical systems in infinite dimensional spaces*. Springer New York.

[18] Smith, H. (2010). *An introduction to delay differential equations with applications to the life sciences*. Springer New York.

[19] Ševčík, J. (2021). *Synchronizace*. Master's thesis, Masarykova univerzita, Přírodovědecká fakulta, Brno.

[20] Wikipedia contributors. (2024). Grothendieck group — Wikipedia, the free encyclopedia.

[21] Hartman, P. (2002). *Ordinary differential equations*. Society for Industrial; Applied Mathematics.

[22] Chicone, C. (2006). *Ordinary differential equations with applications*. Springer, New York, NY.

[23] Perko, L. (2001). *Differential equations and dynamical systems*. Springer New York.

[24] Elaydi, S. (2005). *An introduction to difference equations*. Springer Science+Business Media.

[25] Kloeden, P. E. and Platen, E. (1992). *Numerical solution of stochastic differential equations*. Springer Berlin Heidelberg.

[26] Hale, J. K. (1977). *Theory of functional differential equations*. Springer New York.

[27] Hale, J. K. and Lunel, S. M. V. (1993). *Introduction to functional differential equations*. Springer New York.

[28] Kolmanovskii, V. and Myshkis, A. (1992). *Applied theory of functional differential equations*. Springer Netherlands.

[29] Diekmann, O., Verduyn Lunel, S. M., Gils, S. A. van and Walther, H.-O. (1995). *Delay equations*. Springer New York.

[30] Bellen, A. and Zennaro, M. (2003). *Numerical methods for delay differential equations*. Oxford University Press.

[31] Curtain, R. F. and Zwart, H. (1995). *An introduction to infinite-dimensional linear systems theory*. Springer New York.

[32] Hutchinson, G. E. (1948). Circular causal systems in ecology. *Annals of the New York Academy of Sciences* **50** 221–46.

[33] Butcher, J. C. (2016). *Numerical methods for ordinary differential equations*. Wiley.

[34] Hairer, E., Nørsett, S. P. and Wanner, G. (2008). *Solving ordinary differential equations I*. Springer, Berlin, Germany.

[35] Trefethen, L. N. (1994). *Finite difference and spectral methods for ordinary and partial differential equations*.

[36] Rackauckas, C. and Nie, Q. (2017). DifferentialEquations.jl–a performant and feature-rich ecosystem for solving differential equations in Julia. *Journal of Open Research Software* **5**.

[37] Inc., T. M. (2023). MATLAB version: 23.2.0.2365128 (R2023b).

[38] Widmann, D. and Rackauckas, C. (2022). DelayDiffEq: Generating delay differential equation solvers via recursive embedding of ordinary differential equation solvers. *arXiv preprint arXiv:2208.12879*.

[39] Shampine, L. F. and Thompson, S. (2001). Solving DDEs in matlab. *Applied Numerical Mathematics* **37** 441–58.

[40] Scheutzow, M. (2018). Stochastic differential equations.

[41] Mohammed, S. E. A. (1986). Nonlinear flows of stochastic linear delay equations. *Stochastics* **17** 207–13.

[42] Pinsky, P. F. and Rinzel, J. (1994). Intrinsic and network rhythmogenesis in a reduced traub model for CA3 neurons. *Journal of Computational Neuroscience* **1** 39–60.

[43] Atherton, L. A., Prince, L. Y. and Tsaneva-Atanasova, K. (2016). Bifurcation analysis of a two-compartment hippocampal pyramidal cell model. *Journal of Computational Neuroscience* **41** 91–106.

[44] Kochenderfer, M. J. and Wheeler, T. A. (2019). *Algorithms for optimization*. MIT Press, London, England.

[45] Zemánek, P. (2021). *Optimalizace aneb když méně je více*.

[46] Lacerda de Orio, R. (2010). *Electromigration modeling and simulation*. PhD thesis, echnische Universität Wien.

[47] Keener, J. and Sneyd, J. (2009). *Mathematical physiology*. Springer New York.

[48] Murray, J. D. (2002). *Mathematical biology: I. An introduction*. Springer New York.

[49] Jekl, J. (2018). *Matematické modely neuronu*. Master's thesis, Masarykova univerzita, Přírodovědecká fakulta, Brno.

[50] Skinner, F. K. (2006). Conductance-based models. *Scholarpedia* **1** 1408.

[51] Hodgkin, A. L. and Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology* **117** 500–44.

[52] Morris, C. and Lecar, H. (1981). Voltage oscillations in the barnacle giant muscle fiber. *Biophysical Journal* **35** 193–213.

[53] Lecar, H. (2007). Morris-Lecar model. *Scholarpedia* **2** 1333.

[54] White, J. A., Chow, C. C., Rit, J., Soto-Treviño, C. and Kopell, N. (1998). *Journal of Computational Neuroscience* **5** 5–16.

[55] Ermentrout, G. B. and Terman, D. H. (2010). *Mathematical foundations of neuroscience*. Springer New York.

[56] Záthurecký, J., Eclerová, V., Ševčík, J., Zapadlo, Š. and Přibylová, L. (2025). Phase shifts inside arnold tongues of weakly coupled oscillators. *Communications in Nonlinear Science and Numerical Simulation* 108729.

[57] Izhikevich, E. M. (2006). *Dynamical systems in neuroscience: The geometry of excitability and bursting*. The MIT Press.

[58] Song, H., Mylvaganam, S. M., Wang, J., Mylvaganam, S. M. K., Wu, C., Carlen, P. L., Eubanks, J. H., Feng, J. and Zhang, L. (2018). Contributions of the hippocampal CA3 circuitry to acute seizures and hyperexcitability responses in mouse models of brain ischemia. *Frontiers in Cellular Neuroscience* **12**.

[59] Jiruska, P., Curtis, M. de, Jefferys, J. G. R., Schevon, C. A., Schiff, S. J. and Schindler, K. (2013). Synchronization and desynchronization in epilepsy: Controversies and hypotheses. *The Journal of Physiology* **591** 787–97.

[60] Jacobs, J., LeVan, P., Chander, R., Hall, J., Dubeau, F. and Gotman, J. (2008). Interictal high-frequency oscillations (80–500 hz) are an indicator of seizure onset areas independent of spikes in the human epileptic brain. *Epilepsia* **49** 1893–907.

[61] Jacobs, J., Staba, R., Asano, E., Otsubo, H., Wu, J. Y., Zijlmans, M., Mohamed, I., Kahane, P., Dubeau, F., Navarro, V. and Gotman, J. (2012). High-frequency oscillations (HFOs) in clinical epilepsy. *Progress in Neurobiology* **98** 302–15.

[62] Worrell, G. and Gotman, J. (2011). High-frequency oscillations and other electrophysiological biomarkers of epilepsy: Clinical studies. *Biomarkers in Medicine* **5** 557–66.

[63] Staba, R. J. and Bragin, A. (2011). High-frequency oscillations and other electrophysiological biomarkers of epilepsy: Underlying mechanisms. *Biomarkers in Medicine* **5** 545–56.

[64] Brázdil, M., Pail, M., Halámek, J., Plešinger, F., Cimbálník, J., Roman, R., Klimeš, P., Daniel, P., Chrastina, J., Brichtová, E., Rektor, I., Worrell, G. A. and Jurák, P. (2017). Very high-frequency oscillations: Novel biomarkers of the epileptogenic zone. *Annals of Neurology* **82** 299–310.

[65] Cimbalnik, J., Pail, M., Klimes, P., Travnicek, V., Roman, R., Vajcner, A. and Brazdil, M. (2020). Cognitive processing impacts high frequency intracranial EEG activity of human hippocampus in patients with pharmacoresistant focal epilepsy. *Frontiers in Neurology* **11**.

[66] Wiggins, S. (2003). *Introduction to applied nonlinear dynamical systems and chaos*. Springer, New York, NY.

[67] Dhooge, A., Govaerts, W. and A. Kuznetsov, Yu. (2003). MatCont: A MATLAB package for numerical bifurcation analysis of ODEs. *ACM Transactions on Mathematical Software* **29** 141–64.

[68] Dhooge, A., Govaerts, W., A. Kuznetsov, Yu., Meijer, H. G. E. and Sautois, B. (2008). New features of the software MatCont for bifurcation analysis of dynamical systems. *Mathematical and Computer Modelling of Dynamical Systems* **14** 147–75.

[69] Sieber, J., Engelborghs, K., Luzyanina, T., Samaey, G. and Roose, D. (2014). DDE-BIFTOOL manual - bifurcation analysis of delay differential equations.

[70] Kuznetsov, Y. Introduction to numerical bifurcation analysis: Location and continuation of limit cycles.

[71] Wikipedia contributors. (2025). Jordan curve theorem — Wikipedia, the free encyclopedia.

[72] Brin, L. Q. (2020). *Tea time numerical analysis*.

[73] Govaerts, W., Kuznetsov, Y. A. and Dhooge, A. (2005). Numerical continuation of bifurcations of limit cycles in MATLAB. *SIAM Journal on Scientific Computing* **27** 231–52.

[74] Doedel, E., Keller, H. B. and Kernevez, J. P. (1991). NUMERICAL ANALYSIS AND CONTROL OF BIFURCATION PROBLEMS (II): BIFURCATION IN INFINITE DIMENSIONS. *International Journal of Bifurcation and Chaos* **01** 745–72.

[75] Roose, D. and Szalai, R. (2007). Continuation and bifurcation analysis of delay differential equations. In *Numerical continuation methods for dynamical systems* pp 359–99. Springer Netherlands.

[76] Engelborghs, K. and Roose, D. (2002). On stability of LMS methods and characteristic roots of delay differential equations. *SIAM Journal on Numerical Analysis* **40** 629–50.

[77] Breda, D. (2006). Solution operator approximations for characteristic roots of delay differential equations. *Applied Numerical Mathematics* **56** 305–17.

[78] Verheyden, K. and Lust, K. (2005). A newton-picard collocation method for periodic solutions of delay differential equations. *BIT Numerical Mathematics* **45** 605–25.

[79] Engelborghs, K., Luzyanina, T., Hout, K. J. I. 't. and Roose, D. (2001). Collocation methods for the computation of periodic solutions of delay differential equations. *SIAM Journal on Scientific Computing* **22** 1593–609.

[80] Fairgrieve, T. F. and Jepson, A. D. (1991). O. K. Floquet multipliers. *SIAM Journal on Numerical Analysis* **28** 1446–62.

[81] Breda, D., Maset, S. and Vermiglio, R. (2006). NUMERICAL COMPUTATION OF CHARACTERISTIC MULTIPLIERS FOR LINEAR TIME PERIODIC COEFFICIENTS DELAY DIFFERENTIAL EQUATIONS. *IFAC Proceedings Volumes* **39** 163–8.

[82] Lust, K. and Roose, D. (1998). An adaptive newton–picard algorithm with subspace iteration for computing periodic solutions. *SIAM Journal on Scientific Computing* **19** 1188–209.

[83] Linge, S. and Langtangen, H. P. (2017). Nonlinear problems. In *Finite difference computing with PDEs* pp 353–407. Springer International Publishing.

[84] Hellevik, L. R. (2020). Numerical methods for engineers.Available at https://leifh.folk.ntnu.no/teaching/tkt4140/.

[85] Přibylová, L. (2021). *Nelineární dynamika*. Masarykova univerzita.

[86] Veltz, R. (2020). BifurcationKit.jl.

Appendix A

Algorithms

Algorithm 6: Golden section method

Input: Optimization problem (1.46), initial interval $I = [a, b]$, number of evaluations $N \in \mathbb{N}$
Output: Approximation of minimum $\bar{x}_k$
begin
set $[a_0, b_0] \leftarrow [a, b]$, $l_0 \leftarrow b_0 - a_0$;
calculate $x_2^- \leftarrow a_0 + \frac{1}{\tau^2} l_0$ & $x_2^+ \leftarrow a_0 + \frac{1}{\tau} l_0$;
evaluate $f_2^- \leftarrow f(x_2^-)$, $f_2^+ \leftarrow f(x_2^+)$
if $f_2^- < f_2^+$ **do**
set $[a_2, b_2] \leftarrow [a_0, x_2^+]$;
set Increasing $\leftarrow 1$;
else do
set $[a_2, b_2] \leftarrow [x_2^-, b_0]$;
set Increasing $\leftarrow 0$;
end
set $l_2 \leftarrow b_2 - a_2$
for $i = 3$ **to** N **do**
if Increasing equals 1 **do**
set $x_i^+ \leftarrow x_{i-1}^-$, $f_i^+ = f_{i-1}^-$;
calculate $x_i^- \leftarrow a_{i-1} + \frac{1}{\tau^2} l_{i-1}$;
evaluate $f_i^- \leftarrow f(x_i^-)$
else do
set $x_i^- \leftarrow x_{i-1}^+$, $f_i^- = f_{i-1}^+$;
calculate $x_i^+ \leftarrow a_{i-1} + \frac{1}{\tau} l_{i-1}$;
evaluate $f_i^+ \leftarrow f(x_i^+)$
end
if $f_i^- < f_i^+$ **do**
set $[a_i, b_i] \leftarrow [a_{i-1}, x_i^+]$;
set Increasing $\leftarrow 1$;

```
        else do
            set [a_i, b_i] ← [x_i^-, b_{i-1}];
            set Increasing ← 0;
        end
        set l_i ← b_i − a_i;
        set x̄_i ← (a_i + b_i)/2
    end
end
```

Algorithm 7: Fibonacci method

Input: Optimization problem (1.46), initial interval $I = [a, b]$, number of evaluations $N \in \mathbb{N}$, tolerance $\varepsilon > 0$ small
Output: Approximation of minimum $\bar{x}$

```
begin
    set A ← a, B ← b, n ← N;
    define s := (1−√5)/(1+√5);
    let ρ ← (1/τ)(1−s^n)/(1−s^{n+1}), r ← (1 − ρ)A + ρB;
    set f^r ← f(r) and n ← n − 1
    while n > 0 do
        if n > 1 do
            set l ← ρA + (1 − ρ)B
        else do
            set l ← εA + (1 − ε)r
        end
        update ρ ← (1/τ)(1−s^n)/(1−s^{n+1});
        set f^l ← f(l) and n ← n − 1
        if f^l < f^r do
            set [r, B, f^r] ← [l, r, f^l]
        else do
            set [A, B] ← [B, l]
        end
    end
    set x̄ = (A+B)/2
end
```