On the Combination of Symbolic Dynamics and Dynamic Programming for Dynamic Systems

Julius Rückin

Master's thesis

On the Combination of Symbolic Dynamics and Dynamic Programming for Dynamic Systems

Julius Rückin

November 21, 2020

Chair of Data Processing
Technische Universität München

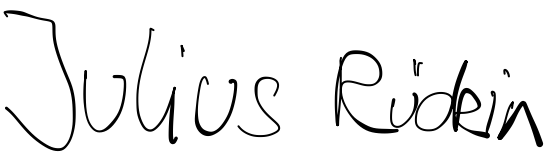
Julius Rückin. *On the Combination of Symbolic Dynamics and Dynamic Programming for Dynamic Systems.* Master's thesis, Technische Universität München, Munich, Germany, 2020.

Supervised by Prof. Dr.-Ing. Klaus Diepold and Martin Gottwald; submitted on November 21, 2020 to the Department of Electrical and Computer Engineering of the Technische Universität München.

Chair of Data Processing, Technische Universität München, 80290 München, Germany, `http://www.ldv.ei.tum.de/`.

I assure the single handed composition of this master's thesis only supported by declared resources.

Munich, November 21, 2020,



Abstract

Investigation and understanding of nonlinear dynamic systems are crucial in various applications and fields of research as describing chaotic systems in physics [34], comprehend bio-molecular processes for designing new medial drugs [3] or in engineering as for developing new high-performance materials [53]. Simultaneously, such systems' global dynamics are notoriously hard to represent, although differential equations to describe these systems are well-known in some cases. A standard tool for analyzing dynamic behavior over time are dynamic simulations [21]. However, these techniques often suffer from a very high dimensional data space resulting in enormous effort to create suitable dimensionality reduction methods [33] [50].

Hence, we investigate a practical and efficient tool to describe and analyze complex dynamic systems, namely symbolic dynamics [44]. We derive symbolic dynamic systems by representing the phase spaces as partitions endowed with the Markov property, called Markov partitions. Such phase space discretizations facilitate analyzing and computing global dynamics. Further, they make dynamic systems accessible to optimal sequential decision-making algorithms. Thus, we extend the work of [24] to bridge a substantial gap in research work by developing algorithms to construct Markov partitions for various dynamic systems automatically. Further, we formalize and implement a framework to build Markov decision processes for dynamic systems based on Markov partitions. We show how to fuse our work with approximate dynamic programming and apply this fused framework in experiments executed in dynamic system environments.

Moreover, our experiments with Markov partitions provide evidence for superior convergence performance of iterative Monte Carlo based policy evaluation algorithms compared to regular grid-like discretizations. However, we also experience mathematically inherent limits while constructing Markov partitions for a broad class of dynamic systems. Additionally, it is not clear how to entirely transport Markov partitions' topological and measure-theoretic properties beyond the application to some fixed policy evaluation step.

Contents

1 Introduction

In the last decades, we have encountered significant progress in characterizing dynamic systems. The theory of dynamic systems is well-studied and establishes a solid foundation for powerful tools investigating an enormous spectrum of systems [28] [4]. Advances in compute power and engineering methods paved the path to bring the theory of dynamic systems into real-world applications, e.g. bio-molecular processes [3], engineering [53], chaos theory [34] or encryption algorithms [6].

The expressiveness of dynamic systems often comes at the cost of complexity. In particular, continuous nonlinear dynamic systems are notoriously hard to analyze. Unfortunately, most real-world environments are modeled by such nonlinear systems. An important method to examine the dynamic behavior of these systems is dynamic simulations [21]. However, dynamic simulations usually lack applicability since they result in high-dimensional state spaces for which classical dimensionality reduction techniques fail to capture global features of such systems' long-term behavior. As a remedy, application-specific dimensionality reduction methods were introduced. Nevertheless, these approaches require tedious engineering work, domain expertise, and tuning of various parameters [37] [50].

Simultaneously, in recent years emerging from the field of reinforcement learning and in particular deep reinforcement learning, optimal sequential decision-making algorithms were proposed that show partially human-like performance in various environments [47] [48]. These advances caused substantial research work dedicated to applying reinforcement learning techniques to system control and robotics tasks. Such real-world environments are endowed with physical behavior captured by the underlying continuous dynamic systems [40] [43].

A unique challenge of a dynamic system is its continuous phase space. It does not allow to apply dynamic programming techniques immediately, but approximating the optimal cost function is required. Therefore, work on state space representations for reinforcement learning problems [58], approximate dynamic programming (ADP) [7] and neuro-dynamic programming [9] emerged to circumvent such problems. However, such methods quickly evolved to highly complicated and resource-demanding frameworks. To some extent, they lack mathematical justification [31] or, especially in the case of continuous control settings, also lack reproducibility [36].

A simple yet frequently utilized method to produce sophisticated cost function approximations is to partition the continuous state space into distinct subsets. Then, one defines linear or quadratic approximations of each subset's cost function. Thus, we ensure piecewise linear or quadratic behavior of the cost function, leading to beneficial optimization effects [9]. Although this approach appears to be a reasonable solution to deal with continuous state spaces, one must pay attention to the technique used to partition the state space. In most applications, a regular grid-like partition is performed to discretize the state space [22]. However, this approach is error-prone since each subset should contain states which behave approximately equal. A primitive, universally applied regular grid-like partition without paying attention to the underlying environment's or agent's dynamics is neither a mathematically convenient technique to aggregate such states nor can one at least approximately guarantee its effectiveness. Thus, the long-known but still relevant question is if there are more sophisticated methods to partition a continuous state space [9].

To this end, we investigate efficient methods to analyze dynamic system behavior and transfer them to ADP. In particular, we aim to build mathematically convenient representations of continuous state spaces. Most controllable real-world systems are well-studied and described by the language of dynamic systems [23] [42]. Thus, exploiting this information about such environments' behavior to assemble reasonable state space partitions feels natural.

Therefore, we introduce symbolic dynamics as a part of the dynamic systems theory. At its heart, symbolic dynamics facilitates the analysis of space- and time-continuous dynamic systems by discretizing the phase space and time. Thus, a system is only observable at some fixed points in time and over finitely many aggregated subsets of its phase space, each associated with a unique symbol. Then, the iterative application of its one-time dynamics are examined. These studies form precise and efficient to analyze mathematical objects, called shift spaces [44].

The critical point in applying symbolic dynamics to space- and time-continuous dynamic systems is to bridge the gap between the continuous and symbolic system description. Therefore, so-called Markov partitions are utilized to discretize the continuous phase space. Based on these Markov partitions, shifts of finite type are derived. On the one hand, shifts of finite type facilitate computing the global properties of a system with simple linear algebra. On the other hand, the created Markov partitions are reusable to discretize a state space while respecting the environment's priorly known dynamics. Moreover, Markov partitions are proven to guarantee that the system's transition from one state into another merely depends on the current system's state. Thus, the Markov property of a Markov decision process (MDP) is guaranteed simply by the design of the state space [8].

Overall, we fuse the concept of Markov partitions with the ADP framework. We propose an algorithmic and automized procedure that creates an MDP for a given dynamic system based on its Markov partition. To achieve this goal, we formalize and implement the fusion of Markov partitions, MDPs, and classical ADP algorithms.

To build such MDPs at scale, we must construct Markov partitions automatically. Nevertheless, most research work in creating Markov partitions involves many arithmetics and manually performed calculations [44]. Further, most semi-automated construction work is neither well-formalized nor implemented so that it does not suit the claim of a rigorously applicable procedure [51]. Thus, at its core, this work also presents an extensive study of existing construction work. Furthermore, we extend the proposed work by [24] to a well-formalized algorithm. The extended implementation does not rely on user interaction and visual feedback. This framework enables us to fuse Markov partitions and ADP. Besides, we deliver a substantially facilitated application of symbolic dynamics to various dynamic systems.

Although our work leads us to a fusion of symbolic dynamics and ADP, there are many open questions. Our theoretical analysis shows that Markov partitions pose strong assumptions on the system, e.g., locally split hyperbolicity of the phase space [27]. Additionally, Markov partitions could become too complex for higher-dimensional systems to be represented in a computer [17]. Moreover, Markov partitions retain their properties only for some fixed policy evaluation step, as we will see while formalizing the fusion of Markov partitions and policy iteration algorithms. Nevertheless, experiments show that Markov partitions could lead to superior convergence behavior of iterative Monte Carlo based policy evaluation algorithms compared to regular grid-like discretizations.

Our main contributions in this thesis are summarized as follows:

- We fuse symbolic dynamics and Markov partitions with ADP algorithms to produce mathematically more advanced state space approximations. We show their practical advantages and inherent limits in ADP applications.
- We perform an extensive analysis of theoretical and practical limits in Markov partitions' existence and construction. We thoroughly examine existing work on constructing Markov partitions.
- We propose an algorithmic and automized procedure to build Markov partitions for locally split hyperbolic systems by extending the work of [24]. Further, we show how to estimate an MDP for such Markov partitions.
- We apply and compare our implemented Markov partition-based MDP framework in policy evaluation experiments with multiple (nonlinear) systems and baseline state space partitions.

In chapter 2, we introduce the central notions of symbolic dynamics and Markov partitions. In chapter 3, we examine the topological idea of locally split hyperbolic systems to examine Markov partitions' strengths and limits. Next, in chapter 4, we propose an algorithmic procedure to create Markov partitions automatically. In chapter 5, we fill the missing parts to define an MDP based on Markov partitions. Further, in chapter 6, we perform experiments to empirically evaluate the effectiveness of Markov partitions applied to iterative Monte Carlo based policy evaluation algorithms. Last, in chapter 7, we highlight the most important related work from symbolic dynamics, general theory of dynamic systems, and ADP. Finally, in chapter 8, we critically summarize and evaluate our contributions and future work.

2 On Symbolic Dynamics for Representations of Dynamic Systems

This chapter introduces the primary concepts and theorems of symbolic dynamics. It presents the foundation to transform continuous dynamics systems into equivalent discrete symbolic systems. The content is mainly based on [44] and gives a selection of the most important results. For an in-depth introduction to symbolic dynamics, we refer to [44]. In the last section, we finally bridge the gap between symbolic and continuous systems. Therefore, we formalize the notion of Markov partitions and state simple constructive examples. These investigations outline the starting point in our work to fuse symbolic dynamics and dynamic programming.

2.1 Shift Spaces

We introduce shift spaces to represent dynamic systems symbolically. Further, we explain how to construct and compare shift spaces by (iso)morphisms in the category of shift spaces. Isomorphisms support us to make precise connections between results from classical dynamic systems theory and symbolic dynamics.

We call a finte set $\mathcal{A}$ of symbols an alphabet. Its elements are called letters. We deal with bi-infinite sequences of symbols, $x = (x_i)_{i\in\mathbb{Z}}$, such that

$$x = ...x_{-2}x_{-1}x_0x_1x_2...,$$

where $\forall i \in \mathbb{Z} : x_i \in \mathcal{A}$ and x_i is the i-th coordinate of x.

Definition Full Shift [44]. Let $\mathcal{A}$ be a finite alphabet. Then the full $\mathcal{A}$-shift is the collection of all sequences x, generated by $\mathcal{A}$. It is denoted by

$$\mathcal{A}^{\mathbb{Z}} = \{x = (x_i)_{i\in\mathbb{Z}} : \forall i \in \mathbb{Z} : x_i \in \mathcal{A}\}.$$

$\mathcal{A}^{\mathbb{Z}}$ is the set of all functions $f : \mathbb{Z} \to \mathcal{A}$ producing these bi-infinite x, which are points in the full shift $\mathcal{A}^{\mathbb{Z}}$.

Blocks (also called words) are a finite number of consecutive symbols generated by some $f : \mathbb{Z} \to \mathcal{A}$, where we restrict to a finite subsequence. The length of a block u is $|u| = k$, where $k \in \mathbb{N}$ is the number of symbols in u. ϵ is the empty block

with $|\epsilon| = 0$. We can select blocks from x by fixing $i, j \in \mathbb{Z}$ with $i \leq j$. Further, $x_{[-k,k]}$ is the central $(2k+1)$-block of x.

The index i in x can be interpreted as representing a certain timestamp. For example, at time step 0, x holds the value $f(0) = x_0 \in \mathcal{A}$. This closely relates to the definition of system dynamics over time. Hence, we intuitively define transformations of a full shift $\mathcal{A}^{\mathbb{Z}}$ to itself.

Definition Shift Map [44]. Let $\sigma : \mathcal{A}^{\mathbb{Z}} \to \mathcal{A}^{\mathbb{Z}}$ be a function, mapping $x \in \mathcal{A}^{\mathbb{Z}}$ to $y = \sigma(x) \in \mathcal{A}^{\mathbb{Z}}$, such that $y_i = x_{i+1}$.

A shift map motivates the terminology *full shift* since σ allows to construct any sequence of symbols without constraints. Later, more restrictive types of shifts are highlighted. This gives rise to the idea of codes $\phi : \mathcal{A}^{\mathbb{Z}} \to \mathcal{A}^{\mathbb{Z}}$. ϕ transforms sequences x into other sequences y. If sequences behave identically at any index, they are independent of time. Then, ϕ is called stationary, and it holds

$$\forall x \in \mathcal{A}^{\mathbb{Z}} : \sigma(\phi(x)) = \phi(\sigma(x)).$$

ϕ is a morphism, particularly an endomorphism on $\mathcal{A}^{\mathbb{Z}}$. Further, periodic points are interesting objects, since they pose an invariant between dynamic systems.

Definition Periodic Points [44]. Let $\mathcal{A}$ be an alphabet and $\mathcal{A}^{\mathbb{Z}}$ its full shift. $x \in \mathcal{A}^{\mathbb{Z}}$ is periodic for the shift map σ, if there is a $n \in \mathbb{N}_+$, such that $\sigma^n(x) = x$, then x has period n under σ. The smallest such n is the least period of x. Further, if $\sigma(x) = x$, then x is a fixed point. Fixed points are crucial for the construction of Markov partitions in chapter 4.

Dynamic systems impose fixed constraints to orbits captured by *shift spaces*. Let $\mathcal{A}$ be an alphabet and $\mathcal{F}$ be a collection of forbidden blocks. Then, $X_{\mathcal{F}} \subseteq \mathcal{A}^{\mathbb{Z}}$ is the set of sequences that do not contain any block $u \in \mathcal{F}$.

Definition Shift Spaces [44]. $X \subseteq \mathcal{A}^{\mathbb{Z}}$ is a shift space with $X = X_{\mathcal{F}}$ for some (countable) $\mathcal{F}$ containing forbidden blocks over $\mathcal{A}$.

Definition Languages [44]. Let $X \subseteq \mathcal{A}^{\mathbb{Z}}$. $\mathcal{B}_n(X)$ denotes the set of all allowed n-blocks, that is

$$\forall n \in \mathbb{N} : \forall b_n \in \mathcal{B}_n(X) : \exists x \in X : b_n \subseteq x,$$

where b_n is a n-block. Then the language of X is the collection

$$\mathcal{B}(\mathcal{X}) = \cup_{n=0}^{\infty} \mathcal{B}_n(X).$$

Proposition 1.3.4. in [44]. Let $X \subseteq \mathcal{A}^{\mathbb{Z}}$ be a shift space with language $\mathcal{L} = \mathcal{B}(X)$.

1. If $w \in \mathcal{L}$, then
   (a) Every subblock of w belongs to $\mathcal{L}$, and
   (b) There are nonempty blocks $u, v \in \mathcal{L}$, such that $uwv \in \mathcal{L}$.
2. The languages of shift spaces are characterized by 1. If $\mathcal{L}$ is a collection of blocks over $\mathcal{A}$, then $\mathcal{L} = \mathcal{B}(X)$ for some X if and only if $\mathcal{L}$ satisfies 1.
3. $\mathcal{L} = \mathcal{B}(X)$ determines the shift space X by $X = X_{\mathcal{B}(X)^c}$. Thus, two shift spaces are equal if and only if they have the same language.

The proposition clearly shows the connection between languages and shifts. Namely, $X \subseteq \mathcal{A}^{\mathbb{Z}}$ is a shift space if and only if for all $x \in \mathcal{A}^{\mathbb{Z}}$ and $i < j$, it holds $x[i, j] \in \mathcal{B}(X)$. Next, we create more advanced alphabets based on an alphabet $\mathcal{A}$. We receive equivalent but more expressive representations of a shift space $X \subseteq \mathcal{A}^{\mathbb{Z}}$. These representations are called higher block codes.

Let X be a shift space over $\mathcal{A}$ and $\mathcal{A}_X^{[N]} = \mathcal{B}_N(X)$. Then $\mathcal{A}_X^{[N]}$ forms the full shifts $(\mathcal{A}_X^{[N]})^{\mathbb{Z}}$. The Nth higher block code $\beta_N : X \to (\mathcal{A}_X^{[N]})^{\mathbb{Z}}$ is defined by

$$(\beta_N(x))_{[i]} = x_{[i,i+N-1]}.$$

Definition Higher Block Shift [44]. Let X be a shift space. Then the Nth higher block shift $X^{[N]}$ or higher block presentation of X is the image $X^{[N]} = \beta_N(X) \subseteq (\mathcal{A}_X^{[N]})^{\mathbb{Z}}$.

$X^{[N]}$ is again a shift space. Hence, previous results still apply. Analogous to higher block shifts that depend on overlapping blocks, higher power codes are defined as a sequence of non-overlapping blocks, such that the Nth higher power code $\gamma_N : X \to (\mathcal{A}_N^{[N]})^{\mathbb{Z}}$ is given by

$$(\gamma_N(x))_{[i]} = x_{[iN,iN+N-1]},$$

where γ_n splits x in consecutive blocks of length N.

Definition Higher Power Shift [44]. Let X be a shift space. The Nth higher power shift X^N of X is the image $X^N = \gamma_N(X)$ of $X \subseteq (\mathcal{A}_X^{[N]})^{\mathbb{Z}}$. Higher power shifts X^N of a shift space X are again shift spaces.

Last, we extend our tool box by so-called *sliding block codes*. Let X be a shift space over an alphabet $\mathcal{A}$. Sliding block codes help transforming a shift space into

another shift space Y over an alphabet $\mathcal{U}$. Fix $m, n \in \mathbb{Z}$ with $-m \leq n$. Then, construct a fixed block map, transforming blocks of length $m+n+1$ in $x \in X$ into symbols in $\mathcal{U}$. The block map $\Phi : \mathcal{B}_{m+n+1} \to \mathcal{U}$ is induced by

$$y_i = \Phi(x_{i-m}...x_{i+n}) = \Phi(x_{[i-m,i+n]}),$$

where x, y are points in the shift spaces X and Y respectively.

Definition Sliding Block Codes [44]. Let $X \subseteq \mathcal{A}^{\mathbb{Z}}$, $Y \subseteq \mathcal{U}^{\mathbb{Z}}$ be shift spaces and $\Phi : \mathcal{B}_{m+n+1} \to \mathcal{U}$ be a block map. Then, $\phi : X \to \mathcal{U}^{\mathbb{Z}}$ with $y = \phi(x)$ with $y_i = \Phi(x_{[i-m,i+n]})$ is called sliding block code with memory m and anticipation n induced by ϕ. Hence, $\phi = \Phi_{\infty}^{[-m,n]}$. If $\phi(X) \subseteq Y$, we write $\phi : X \to Y$.

A sliding block code $\phi : X \to Y$ acts as a homomorphism between X and Y together with the shift maps σ_x and σ_Y. Thus, $\phi \circ \sigma_x = \sigma_Y \circ \phi$. Consequently, ϕ makes statements about (X, σ_X) equally applicable to (Y, σ_Y). Also, $\phi(X)$ is again a shift space. Further, ϕ is a sliding block code if and only if ϕ is a homomorphism between X and Y, and if there is a $N \geq 0$, such that $\phi(x)_0$ is a function of $x_{[-N,N]}$. Additionally, if ϕ is surjective, then ϕ is called a factor map, and Y is a factor of X. If ϕ is injective, then ϕ is called embedding of X into Y.

Definition Conjugacy [44]. Let $\phi : X \to Y$ be a sliding block code. If ϕ has an inverse $\psi : Y \to X$, then ϕ is a conjugacy from X to Y and $X \cong Y$ are conjugate. A conjugacy is an isomorphism in the category of shift spaces.

Conjugacies are crucial for defining two shift spaces as structurally the same. Thus, there are also some invariant properties of shift spaces under conjugacy. For example, the least period of $x \in X$ is invariant under embeddings and hence under conjugacies. If an invariance does not hold, two shift spaces are not conjugate.

2.2 Shifts of Finite Type

Shift spaces $X \subseteq \mathcal{A}^{\mathbb{Z}}$ of finite type are characterized by a finite set $\mathcal{F}$ of forbidden blocks. Shifts of finite type are crucial to represent continuous dynamic systems, which is our primary goal in studying symbolic dynamics. Moreover, shifts of finite type are described by finite and directed graphs. Hence, their analysis effectively breaks down to results from graph theory and linear algebra.

Definition Shifts of Finite Type [44]. A shift of finite type is a shift space that can be described by a finite set of forbidden blocks. Hence, it is a shift space of the already seen form $X_{\mathcal{F}}$ for some finite set $\mathcal{F}$.

However, some $\mathcal{F}'$ could be infinite as well. Shifts of finite type merely require the existence of one finite $\mathcal{F}$ that identically characterizes X. Hence, only shifts that cannot be characterized by any finite $\mathcal{F}$ are not of finite type. Thus, a full shift $X = \mathcal{A}^{\mathbb{Z}}$ is a shift of finite type with $\mathcal{F} = \emptyset$.

M-Step Shifts of Finite Type [44]. A shift of finite type is M-step, if it can be described by a collection of forbidden blocks all of the length $M + 1$. This definition is motivated by the observation that a M-step shift has only M steps of memory to read a forbidden block. Hence, a M-step shift is also K-step for $K \geq M$.

Theorem 2.1.8. in [44]. A shift space X is an M-step shift of finite type if and only if whenever $uv, vw \in \mathcal{B}(X)$ and $|v| \geq M$ then $uvw \in \mathcal{B}(X)$.

Theorem 2.1.10. in [44]. A shift space that is conjugate to a shift of finite type is itself a shift of finite type. Hence, shifts of finite type are closed under conjugacy.

Next, finite and directed graphs are proposed as a convenient tool to represent shifts of finite type. Execute bi-infinite walks over the graph's edges to produce a so-called *edge shift*. Moreover, we gain computational advantages since we can apply linear algebra to compute already defined properties.

Definition Graphs [44]. A graph $G(\mathcal{V}, \varepsilon)$ consists of a finite set $\mathcal{V} = \mathcal{V}(G)$ of vertices and a finite set $\varepsilon = \varepsilon(G)$ of edges. Each edge $e = (i(e), t(e)) \in \varepsilon$ starts at a $i(e) \in \mathcal{V}$ and terminates at a $t(e) \in \mathcal{V}$.

Self-loops are allowed, i.e. $i(e) = t(e)$. For a state I, $\varepsilon_I = \varepsilon_I(G)$ is the set of outgoing edges and $\varepsilon^I = \varepsilon^I(G)$ is the set of incoming edges. $|\varepsilon_I|$ is called out-degree of I and $|\varepsilon^I|$ is called in-degree of I. One can interpret vertices as states and edges as allowed state transitions. Hence, the transition probability $p(t(e)|i(e))$ depends on the existence of $e = (i(e), t(e)) \in \varepsilon_{i(e)}$.

Graph Homomorphisms [44]. Let G, H be graphs. A graph homomorphism from G to H consists of a pair of maps, $\partial\Phi : \mathcal{V}(G) \to \mathcal{V}(H)$ and $\Phi : \varepsilon(G) \to \varepsilon(H)$, such that $i(\Phi(e)) = \partial\Phi(i(e))$ and $t(\Phi(e)) = \partial\Phi(t(e))$ for all $e \in \varepsilon(G)$. Then, $(\partial\Phi, \Phi) : G \to H$. If $\partial\Phi$ and Φ are injective, then $(\partial\Phi, \Phi)$ is a graph embedding. If $\partial\Phi$ and Φ are also surjective, then $(\partial\Phi, \Phi)$ is a graph isomorphism. Then, G and H are identical up to the renaming of vertices and edges. We write $G \cong H$.

Adjacency Matrices [44]. Let G be a graph with vertices $\mathcal{V}$ and edges ε. For $I, J \in \mathcal{V}$, let $A_{I,J}$ be the number of edges $e = (I, J) \in \varepsilon$. Then, the adjacency matrix of G is defined as $A = A(G) = A_G = (A)_{I,J}$, where $I, J \in \mathcal{V}$.

Recall, if there is an adjacency matrix A' for G, then there also exists a permutation matrix P, such that $A' = PAP^{-1}$. Thus, A_G characterizes G up to graph isomorphisms. We write $G = G(A) = G_A$.

Edge shifts [44]. Let $G = (\mathcal{V}, \varepsilon)$ be a graph with adjacency matrix G. The edge shift X_G or X_A is the shift space over the alphabet $\mathcal{A} = \varepsilon$ defined by

$$X_G = X_A = \{\varepsilon = (\varepsilon)_{i\in\mathbb{Z}} \in \varepsilon^Z : t(\varepsilon_i) = i(\varepsilon_{i+1}) \; \forall i \in \mathbb{Z}\}.$$

The edge shift map is denoted by σ_G or σ_A. An edge shift space X_G is the set of all possible bi-infinite walks on G. Note that if $G \cong H$, then $X_G \cong X_H$, since the bi-infinite walks in X_G and X_H are identical up to renaming of edges.

Proposition 2.2.6 in [44]. If G is a graph with adjacency matrix A, then $X_G = X_A$ is a 1-step shift of finite type. Hence, X_G carries the Markov property.

Proof of Proposition 2.2.6. in [44]. Let $G = (\mathcal{V}, \varepsilon)$ be a graph. Let $\mathcal{A} = \varepsilon$ be the alphabet of the corresponding edge shift X_G. Consider

$$\mathcal{F} = \{ef : e, f \in \mathcal{A}, t(e) \neq i(f)\}.$$

$\mathcal{F}$ is obviously finite, since ε is finite by definition, and consists of 2-blocks over $\mathcal{A}$. By definition of an edge shift, $\varepsilon \in \mathcal{A}^{\mathbb{Z}}$ is in X_G if and only if there is no $x \in \mathcal{F}$, such that $x \in \varepsilon$. Hence, $X_G = X_{\mathcal{F}}$. Thus, X_G is of finite type. Further, since all blocks in $\mathcal{F}$ are of length 2, $X_{\mathcal{F}}$ and X_G is 1-step. □

Therefore, the shift space X_G has only a 1-step memory. By interpreting vertices as states, we conclude that the state transition from one state I to another state J or equivalently its corresponding edge $e = (I, J)$ is only influenced by the current state I. Hence, the edge shift X_G holds the Markov property, making this type of shift space suitable to build Markov decision processes. Further, examples for the computational efficiency of edge shifts are stated.

Proposition 2.2.12. in [44]. Let G be a graph with adjacency matrix A, let $m \geq 0$. Then, it holds

1. The number of paths of length m from I to J is $(A^m)_{I,J}$.
2. The number of cycles of length m in G is $tr(A^m)$, which is the number of points in X_G with period m and thus represents a periodic orbit of the dynamic system.

Theorem 2.3.2. in [44]. If X is an M-step shift of finite type, then there is a graph G, such that $X^{[M+1]} = X_G$.

Hence, a graph G for any M-step shift of finite type can be constructed. Thus, edge shifts are by no means only a special shift of finite type but characterize any shift of finite type. Therefore, any shift of finite type could be recoded to a shift that fulfills the Markov property.

Definition Nth higher edge graphs [44]. Let G be a graph and $N \geq 2$. The Nth higher edge graph $G^{[N]}$ has a vertex set containing all paths of length $N-1$ in G and an edge set containing exactly one edge from $e_1...e_{N-1}$ to $f_1...f_{N-1}$ if and only if $e_2...e_{N-1} = f_1...f_{N-2}$, named $e_2...e_{N-1}f_{N-1} = e_1f_1...f_{N-2}$. $G^{[1]} = G$.

Obviously, $(X_G)^{[N]} = X_{G^{[N]}}$. Moreover, for $N \geq 2$, the adjacency matrix of $G^{[N]}$ contains only 0's and 1's. Hence, we can represent a walk on $G^{[N]}$ by the visited vertices.

Definition Vertex Shifts [44]. Let $B \in \{0,1\}^{r\times r}$ be the adjacency matrix of a graph G. The vertex shift $\hat{X}_B = \hat{X}_G$ is the shift space over $\mathcal{A} = \{1,...,r\}$ defined by

$$\hat{X}_B = \hat{X}_G = \{x = (x_i)_{i\in\mathbb{Z}} \in \mathcal{A}^{\mathbb{Z}} : B_{x_ix_{i+1}} = 1 \; \forall i \in \mathbb{Z}\}.$$

Note that up to renaming of symbols, 1-step shifts of finite type and vertex shifts are identical. Further, up to the renaming of symbols, every edge shift is a vertex shift on a different graph.

Definition Nth higher power graph [44]. Let $G = (\mathcal{V}, \varepsilon)$ be a graph and $N \geq 1$. The Nth higher power graph G^N is defined as $\mathcal{V}(G^N) = \mathcal{V}(G)$ and with exactly one edge from I to J for each path in G of length N from I to J. Note that $A_{G^N} = (A_G)^N$ by applying *Proposition 2.2.12*. Further, $X_{G^N} = (X_G)^N$.

Definition Markov Chains [44]. Let S be a finite set. A Markov chain μ on S is an assignment of initial state probabilities $\mu(I) \geq 0$ and state transition probabilities $\mu(J \mid I) \geq 0$ for $I, J \in S$, such that

$$\sum_{I\in S} \mu(I) = 1 \; \forall I \in S$$

and

$$\sum_{J\in S} \mu(J \mid I) = 1 \; \forall I \in S.$$

μ fulfills the Markov property, such that for a sequence of states it holds

$$\mu(I_0I_1...I_n) = \mu(I_0) \cdot \mu(I_1 \mid I_0) \cdot ... \cdot \mu(I_n \mid I_{n-1}).$$

Further, for an already occurred sequence $I_0...I_{n-1}$ it holds

$$\mu(I_n \mid I_0...I_{n-1}) = \mu(I_n \mid I_{n-1}) = \frac{\mu(I_0...I_n)}{\mu(I_0...I_{n-1})}.$$

Let $G = (\mathcal{V}, \varepsilon)$ with $\mathcal{V} = \{I \in S : \mu(I) > 0\}$ and $\varepsilon = \{(I, J) : \mu(J \mid I) > 0\}$. There exists at most one edge between each pair of vertices. Hence, the state sequences with a probability larger 0 are the state sequences of paths in G. Equivalently, these are the blocks appearing in $B(\hat{X}_G)$ of the vertex shift $\hat{X}_G$. Thus, vertex shifts hold the set of allowed state transitions. However, we cannot make explicit statements about occurrence probabilities of such transitions.

Note that a Markov chain is also an assignment of probabilities $\mu(I) > 0$ for $I \in \mathcal{V}$ and $\mu(e \mid i(e))$ for $e \in \varepsilon$, such that

$$\sum_{I \in \mathcal{V}} \mu(I) = 1 \forall I \in \mathcal{V}$$

and

$$\sum_{e \in \varepsilon_I} \mu(e \mid I) = 1 \forall I \in \mathcal{V}.$$

The initial state probabilities $\mu(I)$ for all $I \in \mathcal{V}$ are summarized in an initial state distribution vector $\boldsymbol{p}$, such that $\boldsymbol{p}_I = \mu(I)$. Further, the conditional probabilities $\mu(e \mid I)$ are summarized in a matrix $P \in \mathbb{R}_{\geq 0}^{|\mathcal{V}| \times |\mathcal{V}|}$, such that $P_{I,J} = \sum_{e \in \varepsilon_I^J} \mu(e \mid I)$. Hence, $\sum_{J \in \mathcal{V}} \sum_{e \in \varepsilon_I^J} = 1$ for a fixed $I \in \mathcal{V}$. Further, $(P^m \boldsymbol{p})_I$ is the probability that the system is in state $I \in \mathcal{V}$ after following a random path of length m randomly starting in some $J \in \mathcal{V}$ with respect to the initial distribution $\boldsymbol{p}$.

2.3 From Dynamic Systems to Shifts of Finite Type

This section clarifies how symbolic dynamics fit into the theory of dynamic systems. We connect continuous dynamic systems with discrete symbolic systems. Further, we show that objects studied in symbolic dynamics will naturally emerge in dynamic systems theory. Therefore, we introduce shift spaces as metric spaces and define non-discrete notions as continuity and convergence for shift spaces. Additionally, we investigate invariants for symbolic and continuous systems. Last, we transfer continuous systems into shifts of finite type. This transformation requires a discretization of the system's phase space by Markov partitions. We examine the Markov partition construction process using simple examples and motivate the fusion of Markov partitions and dynamic programming.

Intuitively, closeness between two points in a shift space is given, if large central blocks of their coordinates agree. Let $M = \mathcal{A}^{\mathbb{Z}}$. For $x, y \in M$, the metric $\rho : M \times M \to \mathbb{R}_{\geq 0}$ is defined by

$$\rho(x, y) = \begin{cases} 2^{-k} & \text{if } x \neq y \text{ and } k \text{ is maximal w.r.t. } x_{[-k,k]} = y_{[-k,k]}, \\ 0 & \text{if } x = y. \end{cases}$$

Obviously, $\rho(x, y) = 0$ if and only if $x = y$ and $\rho(x, y) = \rho(y, x)$. Further, we check the triangle inequality. Let $x, y, z \in M$. If $\rho(x, y) = 2^{-k}$, then $x_{[-k,k]} = y_{[-k,k]}$. Similar, if $\rho(y, z) = 2^{-l}$, then $y_{[-l,l]} = z_{[-l,l]}$. Set $m = min\{k, l\}$. Then, $x_{[-m,m]} = z_{[-m,m]}$, and thus

$$\rho(x, z) \leq 2^{-m} \leq 2^{-k} + 2^{-l} = \rho(x, y) + \rho(y, z).$$

Therefore, ρ is indeed a metric over a shift space M. (M, ρ) is a metric space.

Definition Diameters [44]. Let (M, ρ) be a metric space and $E \subseteq M$. The diameter of E is defined by

$$diam(E) = \sup \rho(x, y) : x, y \in E.$$

Convergence of Sequences [44]. Let (M, ρ) be a metric space. A sequence $\{x_n\}_{n=1}^{\infty} \subseteq M$ converges to some $x \in M$, if $\rho(x_n, x) \to 0$ as $n \to \infty$. We write $x_n \to x$ as $n \to \infty$ or equivalently $\lim_{n\to\infty} x_n = x$. Note that if $x_n \to x$ and $x_n \to y$ as $n \to \infty$, then $x = y$. So, if a sequence $x_n \in M$ converges, it converges to exactly one point $x \in M$.

Continuity in Metric Spaces [44]. Let M, N be metric spaces and $\phi : M \to N$. ϕ is continuous, if for all $x_n \to x \in M$, then $\phi(x_n) \to \phi(x) \in N$. If ϕ is surjective, injective and a continuous ϕ^{-1} exists, then ϕ is a homeomorphism.

Compactness in Metric Spaces [44]. Let M be a metric space. M is compact, if every sequence in M has a convergent subsequence.

Theorem 6.1.21. in [44]. $X \subseteq \mathcal{A}^{\mathbb{Z}}$ is a shift space if and only if X is shift-invariant and compact.

Definition Cylinder Sets [44]. Let X be a shift space and fix $u \in \mathcal{B}(X)$ and $k \in \mathbb{Z}$. The cylinder set for block u at coordinate k is defined as follows:

$$C_k(u) = C_k^X(U) = \{x \in X : x_{[k,k+|u|-1]} = u\}.$$

Further definitions and results to work with *Markov partitions* of dynamic systems are required. On the one hand, Markov partitions are an essential tool for bridging the gap between continuous and symbolic systems. On the other hand, they are crucial to discretize the state space required to apply various dynamic programming techniques.

Definition Dense Sets [44]. Let $E \subseteq F$. E is called dense in F, if $F \subseteq \bar{E}$.

Proposition 6.1.23 in [44]. Let M be a metric space and let $E_1, ..., E_n$ be dense and open in M. Then, $\cap_{k=1}^{n} E_k$ is open and dense in M.

Theorem 6.1.24. (Baire Category Theorem) in [44]. Let M be a compact metric space and let $E_1, E_2, ...$ be open and dense in M. Then, $\cap_{k=1}^{\infty} E_k$ is dense in M.

Definition Dynamic Systems [44]. A dynamic system (M, ϕ) is a compact metric space M together with a continuous map $\phi : M \to M$. If ϕ is a homeomorphism, then (M, ϕ) is called invertible dynamic system. If $M = X$ is a shift space and $\phi = \sigma_X$ is the shift map of x, (M, ϕ) is called a shift dynamic system.

If some $x, y \in X$ are close to each other, they agree on a large central block of length $2k + 1$. After applying σ to x and y, they still agree on a large central block that is at most slightly shorter. Hence, σ is continuous. Together with the established compactness of X, (X, σ) is a dynamic system.

Definition Topological Conjugacy [44]. Let (M, ϕ), (N, ψ) be dynamic systems and $\theta : (M, \phi) \to (N, \psi)$ be a homomorphism. If θ is a monomorphism, then θ is an embedding. If θ is an epimorphism, then θ is a factor map. If θ is bijective and a continuous θ^{-1} exists, then θ is a topological conjugacy and $(M, \phi) \cong (N, \psi)$.

If M and N are compact metric spaces, then θ^{-1} is continuous. Hence, in the category of shift spaces, if θ is bijective, it is guaranteed to be a topological conjugacy.

(Curtis-Lyndon-Hedlund) Theorem 6.2.9. in [44]. Let (X, σ_X), (Y, σ_Y) be shift dynamic systems. Suppose that there exists a function $\theta : X \to Y$. Then, θ is a sliding block code if and only if it is a homomorphism.

Hence, if two dynamic systems are represented as shift dynamic systems, establishing a topological conjugacy breaks down to finding a conjugacy between both symbolic systems.

Definition Topological Entropy [44]. Let (M, ϕ) be a dynamic system and $\epsilon > 0$. Some $E \subseteq M$ is (n, ϵ)-spanning for ϕ, if the following holds true:

$$\forall x \in M : \exists y \in E : \forall 0 \leq j < n : \rho(\phi^j(x), \rho^j(y)) \leq \epsilon.$$

Since ϕ is continuous and M is compact, E is finite for all $\epsilon > 0$ and $n \geq 1$. Denote the size of E by $r_n(\phi, \epsilon)$. Define

$$r(\phi, \epsilon) = \lim_{n \to \infty} \sup \frac{1}{n} \log r_n(\phi, \epsilon).$$

as the growth rate of $r_n(\phi, \epsilon)$ as $n \to \infty$. Then,

$$h(\phi) = \lim_{\epsilon \to 0} r(\phi, \epsilon)$$

is the topological entropy of the dynamics ϕ. $h(\phi)$ is an invariant. Hence, if the topological entropies of two dynamic systems differ, they cannot be conjugate.

Definition Periodic Points [44]. Let (M, ϕ) be a dynamic system and let $n \geq 1$. The number of period points of period n is defined as

$$p_n(\phi) = \mid \{x \in M : \phi^n(x) = x\} \mid$$

and represents another invariant of dynamic systems.

Definition Zeta Function [44]. Let (M, ϕ) be a dynamic system with $p_n(\phi) < \infty$ for all $n \geq 1$. The zeta function is defined as

$$\zeta_\phi(t) = \exp\left(\sum_{n=1}^{\infty} \frac{p_n(\phi)}{n} t^n\right).$$

Definition Generating Function [44]. Let (M, ϕ) be a dynamic system with $p_n(\phi) < \infty$ for all $n \geq 1$. The generating function is defined as

$$g_\phi(t) = \sum_{n=1}^{\infty} p_n(\phi) t^n.$$

For a shift of finite type X with underlying graph G, the zeta function is particularly easy to calculate. Let $A \in \mathbb{N}^{r \times r}$ be the adjacency matrix of G. Then,

$$\zeta_{\sigma_A}(t) = \frac{1}{\det(I - tA)}.$$

Now, we create shift spaces from a given dynamic system (M, ϕ). Suppose that ϕ^{-1} exists and let $\{\phi^n(y) : n \in \mathbb{Z}\}$ be the orbits of points $y \in M$. We split M in a finite partition $E_0, ..., E_{r-1}$. An orbit of some $y \in M$ can be represented by $x = ...x_{-1}x_0x_1... \in \{0, 1, ..., r-1\}^{\mathbb{Z}} = X_{[r]}$ and is defined as

$$\phi^n(y) \in E_{x_n} \text{for } n \in \mathbb{Z}.$$

Hence, for each $y \in M$, we have a symbolic point x in the full r-shift. By definition, $\phi(y)$ corresponds to $\sigma(x)$. The correspondence is not necessarily one-to-one, since $\phi(y)$ could be on the boundary of some E_i and E_j for $i \neq j$. Thus, the set of all points in $X_{[r]}$ is perhaps not compact. Hence, they do not form a shift space.

Definition Topological Partition [44]. A topological partition of a metric space M is a finite collection $\mathcal{P} = \{P_0, P_1, ..., P_{r-1}\}$ of disjoint open sets with $\bar{P}_j$ for all $j \in [r-1]$ cover M, such that $M = \bar{P}_0 \cup ... \cup \bar{P}_{r-1}$.

Let $\mathcal{A} = \{0, ..., r-1\}$ and let $w = a_1...a_n$ be a word. w is allowed for $\mathcal{P}$, ϕ, if $\cap_{j=1}^{n} \phi^{-j}(P_{a_j}) \neq \emptyset$. Then, $\mathcal{L}_{\mathcal{P},\phi}$ is the collection of allowed words. If ϕ is invertible, $X_{\mathcal{P},\phi}$ is called symbolic dynamic system depending on $\mathcal{P}$, ϕ.

Definition Markov Partitions [44]. Let (M, ϕ) be an invertible dynamic system and $\mathcal{P} = \{P_0, ..., P_{r-1}\}$ be a topological partition of M. For each $x \in X_{\mathcal{P},\phi}$ and $n \geq 0$, there exists a non-empty and open set

$$D_n(x) = \cap_{k=-n}^{n} \phi^{-k}(P_{x_k}) \subseteq M.$$

$\bar{D}_n(x)$ are compact and $\bar{D}_0(x) \supseteq \bar{D}_1(x) \supseteq ....$ Hence, $\cap_{n=0}^{\infty} \bar{D}_n(x) \neq \emptyset$. $\mathcal{P}$ induces a symbolic representation of (M, ϕ), if for all $x \in X_{\mathcal{P},\phi}$ it holds that

$$| \cap_{n=0}^{\infty} \bar{D}_n(x) | = 1.$$

$\mathcal{P}$ is called a Markov partition for (M, ϕ), if $\mathcal{P}$ induces a symbolic representation for (M, ϕ). Then, $X_{\mathcal{P},\phi}$ is a shift of finite type.

Let $\mathcal{P}$ be a symbolic representation of an invertible (M, ϕ). Then, there exists a well-defined map $\pi : X_{\mathcal{P},\phi} \to M$, $\pi : x = ...x_{-1}x_0x_1... \mapsto \pi(x)$, where x is the symbolic representaion of a unique $\pi(x)$ which is the respective orbit in the continuous dynamic system. Note, that

$$D_{n+1}(\sigma(x)) \subseteq \phi(D_n(x)) \subseteq D_{n-1}(\sigma(x)).$$

Proposition 6.5.8. in [44]. Let $\mathcal{P}$ be a symbolic representation for an invertible (M, ϕ). Let $\pi : X_{\mathcal{P},\phi} \to M$ be defined as above. Then, π is a factor map from $(X_{\mathcal{P},\phi}) \to (M, \phi)$. Hence, $\phi \circ \pi = \pi \circ \sigma$ commutes and π is an epimorphism between $X_{\mathcal{P},\phi}$ and (M, ϕ).

Thus, if we can construct a Markov partition $\mathcal{P}$ for some invertible (M, ϕ) , a homomorphism π between the induced shift of finite type and (M, ϕ) itself exists. Hence, both systems are structurally identical. Analysis performed on $X_{\mathcal{P},\phi}$ remains valid in (M, ϕ). Thus, we construct $X_{\mathcal{P},\phi}$ to facilitate analyzing (M, ϕ).

The main problem is how to construct such a Markov partition. Are there limits for which systems one can build Markov partitions? Is there an algorithmic construction procedure? Hence, we examine different dynamic systems classes. For an in-depth investigation of these questions, see chapter 3.

Last, we highlight an example of a Markov partition construction inspired by [44]. Such a simple example already leads to tedious arithmetic efforts while creating such a partition. This encourages subsequent research in finding an automated method to build Markov partition. However, the example at hand already introduces the natural properties of Markov partitions.

Let (M, ϕ) be defined by $M = \mathbb{T}^2$ and $\phi : \mathbb{T}^2 \to \mathbb{T}^2$ with $\phi(s,t) = (s+t, s)\ (mod\ 1)$. Note that $\mathbb{T}^2 = \mathbb{S}^1 \times \mathbb{S}^1 = [0,1] \times [0,1]$ with $(0,t) = (1,t)$ for all $t \in [0,1]$ and $(s,0) = (s,1)$ for all $s \in [0,1]$. $\mathbb{T}^2$ is a group under coordinate-wise addition $mod\ 1$. Since $\phi^{-1}(s,t) = (t, s-t)\ (mod\ 1)$ is the inverse of ϕ, $(\mathbb{T}^2, \phi)$ is an invertible dynamic system. ϕ is called a toral automorphism.

We characterize $\mathbb{T}^2 = \mathbb{R}^2 \setminus \mathbb{Z}^2$ by identifying $(s,t) \in \mathbb{T}^2$ with the coset $(s,t) + \mathbb{Z}^2 = \{(s,t) + (a,b) : a \in \mathbb{Z}, b \in \mathbb{Z}\}$. Hence, $(s,t) = (s',t')$, if $(s,t) - (s',t') \in \mathbb{Z}^2$. Thus, coordinate-wise addition over $\mathbb{T}^2$ is equivalent to addition of cosets in $\mathbb{R}^2/\mathbb{Z}^2$. Therefore, the quotient mapping $q : \mathbb{R}^2 \to \mathbb{T}^2$ is given by $q((s,t)) = (s,t) + \mathbb{Z}^2$.

Next, for $\phi(s,t) = (s+t, s)$, there is a $A \in \mathbb{Z}^2$, $A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$, such that $A(s,t)^T = (s+t, s)^T$. Hence, $q \circ A = \phi \circ q$ commutes. Thus, we can analyze the dynamic system over the torus or in the euclidean plane.

Further, there is an eigenvector L_1 with eigenvalue $|\lambda_1| > 1$ and an eigenvector L_2 of A with eigenvalue $|\lambda_2| < 1$. Hence, A is expanding in direction of L_1 and contracting in direction of L_2 by definition of eigenvalues. We call L_1 *expanding segment* and L_2 *contracting segment*. The construction of a Markov partition $\mathcal{P} = \{P_1, P_2, P_3\}$ in $\mathbb{R}^2$, such that $\cup_{i=1}^3 q(\bar{P}_i) \subseteq \mathbb{T}^2$ is visualized in Figure 2.1.

One proves that the partition $\mathcal{P}$ is indeed Markov by showing:

1. $\mathcal{P}$ is a symbolic representation for $(\mathbb{T}^2, \phi)$, and
2. $X_{\mathcal{P},\phi}$ is a shift of finite type.

The primary tool to see that both statements hold is the intersection property.

Definition Intersection Property [44]. If $\phi(P_i) \cap P_j \neq \emptyset$ for some i, j, then $\phi(P_i) = \phi(P_i) \cap P_j$ is a single connected strip parallel to the extending segment L_1 that cuts completely through P_j in L_1. Further, it does not contain an expanding boundary segment of P_j. For visual interpretation, see Figure 2.2.

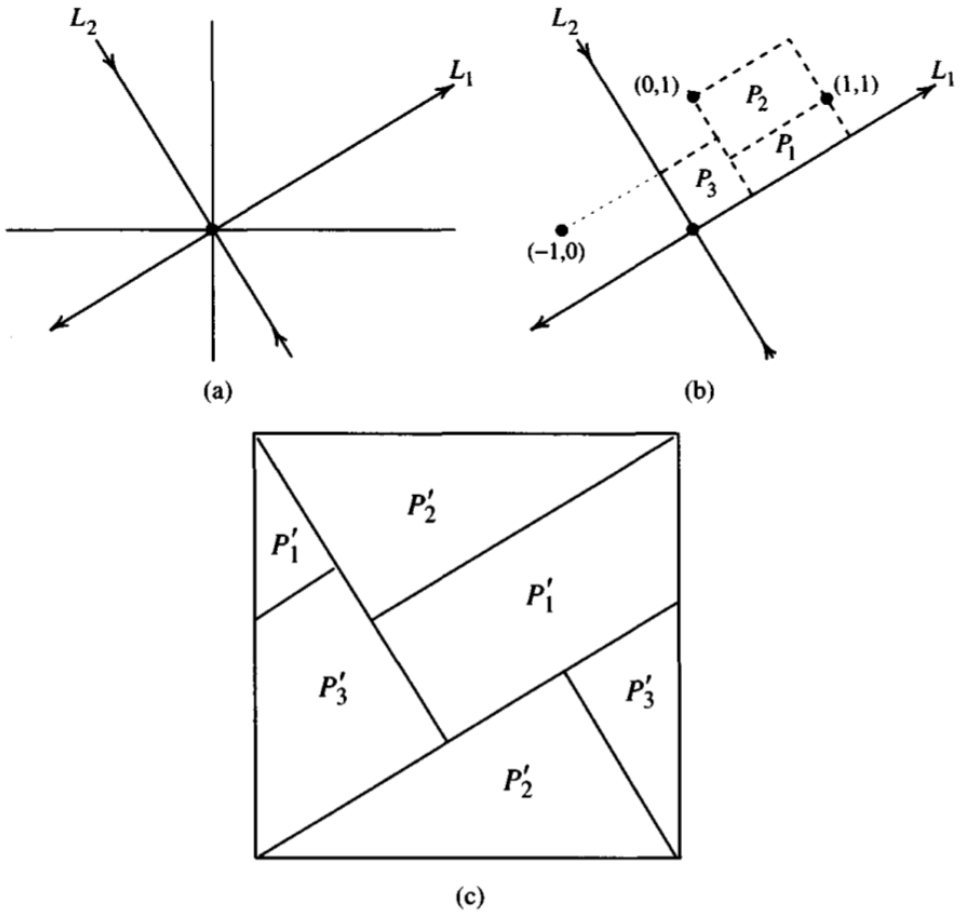


Figure 2.1: (a) shows expanding and contracting eigenspaces. (b) shows rectangles of partitions aligned with expanding and contracting segments. (c) shows the q-images of the P_i. The visualization is taken from [44].

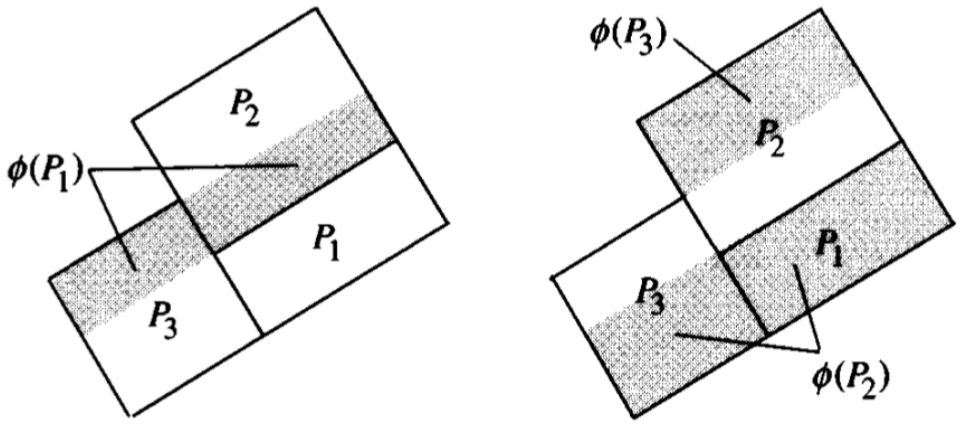


Figure 2.2: Verify that the gray $\phi(P_i)$ indeed fulfill the intersection property. The P_i are also called Markov rectangles. The visualization is taken from [44].

The union rectangle boundaries is the image under q of the union of the expanding segment E in L_1 and the contracting segment C in L_2. Hence, $\phi(E) = A(E) \supset E$ and $A(C) \subset C$. The intersection $\phi(P_i) \cap P_j \neq \emptyset$ has to be a strip that cuts through P_j in extending direction, otherwise $A(C) \subset C$ would be contradicted. Symmetrically, if $\phi(P_i) \cap P_j$ would contain a boundary of P_j that is aligned in expanding direction, it would contradict $E \subset A(E)$.

It is not trivial to guarantee that $\phi(P_i) \cap P_j$ is a single connected strip. $\phi(P_i)$ could be expanded such that it completely wraps around the torus. Then, $\phi(P_i) \cap P_j$ would cut P_j at least twice. Thus, the rectangles must be sufficiently small. Therefore, it is not sufficient to align these Markov rectangles according to contracting and expanding segments. In general, the precise rectangle shapes are highly dependent on the dynamic system. Figure 2.2 verifies the intersection property for $\mathcal{P}$.

The intersection property is elementary to prove that $\mathcal{P}$ is a Markov partition. Let (a_i, b_i) be the dimension of P_i. Since $\phi(P_i) \cap P_j$ fulfills the intersection property, if $\phi(P_i) \cap P_j \neq \emptyset$ for all i, j, then $\phi(P_i) \cap P_j$ is of dimension $(a_j, \lambda_1^{-1} b_i)$, because A uniformly contracts by $|\lambda_2| = \lambda_1^{-1}$ in L_2 direction. Repeatedly applying this argument n times gives an intersection strip of dimension $(a_k, \lambda_1^{-n} b_i)$. Its reversed application with ϕ^{-1} has dimensions $(\lambda_1^{-n} a_k, b_i)$. Hence, by extending this argument to $n \to \infty$, if $x \in X_{\mathcal{P},\phi}$, then $\cap_{k=0}^{n} \phi^{-k}(P_{n_k})$ is a single strip in P_{x_0} with dimensions $(\lambda_1^{-n} a_{x_n}, b_{x_0})$. Symmetrically, for ϕ^{-1} we have $\cap_{k=-n}^{0} \phi^{-k}(P_{n_k})$ is a single strip in P_{x_0} with dimensions $(a_{x_0}, \lambda_1^{-n} b_{x_{-n}})$ [44].

All together, $D_n(x) = \cap_{k=-n}^{n} \phi^{-k}(P_{x_k})$ is of dimension $(\lambda_1^{-n} a_{x_n}, \lambda_1^{-n} b_{x_{-n}})$. Hence, $\bar{D}_0(x) \supseteq \bar{D}_1(x) \supseteq ...$ since $diam(D_n(x)) = diam(\bar{D}_n(x))$. This decreasing sequence of compact sets shrinks to zero in diameter, such that $|\cap_{n=0}^{\infty} \bar{D}_n(x)| = 1$. Thus, $\mathcal{P}$ is indeed a symbolic representation [44].

Since for $\mathcal{P}, \phi$ the intersection property holds, if $\phi(P_i) \cap P_j \neq \emptyset$ and $\phi(P_j) \cap P_k \neq \emptyset$, then $\phi^2(P_i) \cap \phi(P_j) \cap P_k \neq \emptyset$. Hence, if $x_i x_j \in \mathcal{B}(X_{\mathcal{P},\phi})$ and $x_j x_k \in \mathcal{B}(X_{\mathcal{P},\phi})$, then $x_i x_j x_k \in \mathcal{B}(X_{\mathcal{P},\phi})$. This argument applies to arbitrary intersections. Thus, $\mathcal{B}(X_{\mathcal{P},\phi}) = \mathcal{L}_{\mathcal{P},\phi}$ is the set of allowed words over $\mathcal{P}$ consisting of blocks of length two in the vertex shift $\hat{X}_B = X_{\mathcal{P},\phi}$. The adjacency matrix $B \in \{0,1\}^{|\mathcal{P}|\times|\mathcal{P}|}$ of $\hat{X}_B$ is defined as follows:

$$B_{i,j} = \begin{cases} 1 & \text{if } \phi(P_i) \cap P_j \neq \emptyset \\ 0 & \text{if } \phi(P_i) \cap P_j = \emptyset \end{cases}$$

B characterizes the graph of $\hat{X}_B$. Hence, $X_{\mathcal{P},\phi}$ is a shift of finite type. Thus, $\mathcal{P}$ is a Markov partition [44]. □

Observe that we constructed a Markov partition $\mathcal{P}$ given a dynamic system $(\mathbb{T}^2, \phi)$. Thus, we can derive a vertex shift $\hat{X}_B$. Based on $\hat{X}_B$, we can again apply the results of symbolic dynamics introduced above. This analysis then directly carries over to the continuous dynamic system $(\mathbb{T}^2, \phi)$.

3 From Dynamic Systems to Markov Partitions and Symbolic Dynamics

This chapter introduces the main topological and geometrical results in dynamic systems to understand the abilities and limits of constructing Markov partitions introduced in section 2.3. We bridge the gap between these abstractly stated results and the dynamic systems' combinatorial representation by symbolic dynamics. This bridge is built upon the idea of Markov partitions, motivated from a topological point of view. Constructive proofs about their existence are highlighted.

During the development of various theorems, we reflect on their expressiveness for achieving our primary goal of constructing Markov partitions for a broad class of dynamic systems. Most of the stated observations are based on [27]. We give intuitions for the relevance and only state proofs if they support the understanding how to build a general construction procedure for Markov partitions. For a more extensive introduction to hyperbolic dynamics and its roots within the theory of differential geometry and algebraic topology, we refer to [27].

3.1 Topological Universality of Toral Endomorphisms

The most relevant dynamic systems for our work are hyperbolic systems. They impose a local state space structure that allows constructing Markov partitions. Hence, the notion of hyperbolic systems is defined. It is also examined why transforming dynamic systems into a system over the torus is a powerful algebraic concept. Additionally, it is elaborated on how to guarantee the system's structural equivalence, such that one can work with both systems. Namely, the topological universality theorem presented in [27] is investigated.

Contracting, Expanding, Hyperbolic [27]. Let (Y, T) be a dynamic system over the manifold Y with system dynamics $T : Y \to Y$. A linear transformation T of a normed space Y is eventually contracting, if

$$\exists k > 0 \in \mathbb{Z}, C < 1 \in \mathbb{R} : \forall y \neq 0 \in Y : \left\| T^k(y) \right\| \leq C \left\| y \right\| .$$

T is expanding, if it is invertible and T^{-1} is evantually contracting.

T is hyperbolic, if there are T-invariant splittings,

$$Y = Y_{contr} \oplus Y_{exp}, T = T_{contr} \oplus T_{exp},$$

where $T_{contr} : Y_{contr} \to Y_{contr}$ is a contraction mapping and T_{exp} is an expanding mapping respectively. Y_{contr} is T-invariant, if for all $y_{contr} \in Y_{contr}$, then $T(y_{contr}) \in Y_{contr}$ and analogously for Y_{exp}.

For linear transformations T, deciding if T is hyperbolic breaks down to the computation of eigenvalues λ_i of T. If all $\mid \lambda_i \mid \neq 1$, then T is hyperbolic. Note that if $|\lambda_i| < 1$, then all points along the respective eigenvector v_i will contract to the origin. Thus, these points shrink in norm under iterative application of T. Symmetrically, the same observation applies to $|\lambda_i| > 1$ for iterative application of T^{-1} by the definition of eigenvalues.

Topological Universality Theorem [27]. Let V be a compact locally contractible space (e.g. compact manifold) and let $A = A(V)$ denote the homological Abel-Jacobi torus of V,

$$A = H_1(V; \mathbb{R}) \to (H_1(V; \mathbb{Z})/torsion),$$

which is isomorphic to $\mathbb{T}^n$, where n is the first betti number $\beta_1 = rank H_1(V)$. Further, let

$$Ab : H_1(V; \mathbb{R}) \to H_1(A; \mathbb{R})$$

be the Abel's isomorphism.There is a unique homotopy class of continuous Abel maps $q : V \to A$ for which the induced homomorphisms $q_* : H_1(V; \mathbb{R}) \to H_1(A; \mathbb{R})$ are equal Ab. Since all q are homotopic, all induced $q_* = Ab$. Hence, Ab isomorphically characterizes the first homology groups of V and A. Thus, the fundamental groups of a manifold V and its Abel-Jacobi torus A are closely related. This is a first connection to transforming the toy example in section 2.3 to the torus.

Let $f : V \to V$ be continuous and $f_* : H_1(V; \mathbb{R}) \to H_1(V; \mathbb{R})$, $\underline{f}_* : A \to A$ be induced by f. The linear f_* and the toral endomorphism $\underline{f}_*$ determine each other.

Let either f_* be expanding or f be a homeomorphism and f_* hyperbolic. Then there is a continuous map $\alpha : V \to A$, such that

1. the induced homomorphism $\alpha_* : H_1(V; \mathbb{R}) \to H_1(A; \mathbb{R})$ equals Ab, hence the induced $\alpha : V \to A$ is in the Abel's homotopy class of maps q.
2. α is a morphism in the category of spaces with $\mathbb{Z}_+$-actions (i.e. shift spaces), called semi-conjugacy in symbolic dynamics. Hence, α is an f-morphism for

   $$(\alpha \circ f)(v) = (\underline{f}_* \circ \alpha)(v) \; \forall v \in V.$$

   Thus, α acts as a translator q in the sense of section 2.3.

3. α is unique up to group translations of A, which preserve the fixed points of $\underline{f_*}$ acting on the torus A.

The topological universality theorem shows the expressiveness of a dynamic system (V, f) with hyperbolic system dynamics f_* over its first homology group. The existence of $\alpha : V \to A$ ensures structural equivalence of results on the torus since α is an (almost unique) morphism or semi-conjugacy between the systems (V, f) and $(A, \underline{f_*})$. However, the theorem only states the existence of such an α. Further, it is not clear how to verify if f_* is hyperbolic. This involves calculating the first homology group for general manifolds V, which is not straightforward.

Additionally, the main lemma of hyperbolic systems, which paves the path to the existence of such an α is stated. Also, it highlights the essential property of global structural stability of hyperbolic systems.

Shadowing Lemma [27]. Let X and Y be $\mathbb{Z}$-spaces with $S : X \to X$ and $T : Y \to Y$ respectively. Let Y be a metric space. Then, $Q : X \to Y$ is called a $\mathbb{Z}$-quasimorphism if the function

$$x \mapsto dist_Y(Q \circ S(x), T \circ Q(x))$$

is bounded on the S-orbits $\{...S^{-2}(x), S^{-1}(x), x, S(x), S^2(x)...\} \subset X$ for all $x \in X$.

If Y is a Banach space and T is hyperbolic, then every $\mathbb{Z}$-quasimorphsim Q_0 is shadowed by a unique morphism $M : X \to Y$, that is

$$M \circ S = T \circ M$$

and where shadowed means that

$$x \mapsto dist_Y(Q_0(x), M(x)) = \|Q_0(x) - M(x)\|$$

is bounded on all orbits in X.

Hence, if (Y, T) is a hyperbolic system, then the orbit of any particle $x \in X$ approximately transformed by a quasimorphism Q_0 into an orbit in Y is uniformly close to the actual trajectory of the orbit of $x \in X$ transformed into the hyperbolic system by the unique true morphism M. In the particular case of $X = Y$ and $S = T$, each orbit in a hyperbolic system can indeed be approximated (e.g., by discretized numerical computations in a computer) and still qualitatively behaves globally very similar to its actual orbit's trajectory. Further, this implies that all approximately close orbits in some fixed orbit neighborhood have a similar qualitative long-term behavior. Thus, a hyperbolic system is said to be structurally stable even on a global scale.

3.2 Locally Split Hyperbolic and Anosov Systems

This section extends the notion of hyperbolicity to locally split hyperbolic systems. Moreover, Anosov actions, foliations, and diffeomorphisms resulting in a general interpretation of Markov partition boundaries for various systems are introduced. Further, it reflects on the applicability and construction of such Anosov foliations, and discusses Anosov diffeomorphisms' universality.

Diffeomorphism. A diffeomorphism is a bijective, continuously differentiable function $f : X \to Y$ with a continuously differentiable inverse $f^{-1} : Y \to X$ between differentiable manifolds X and Y. Also, f is a homeomorphism between X and Y. Hence, a diffeomorphism is an isomorphism between differentiable manifolds.

Anosov actions [27]. A C^1-smooth action of the group $\mathbb{Z}$ on a smooth Riemannian manifold X is called Anosov action, if $Df_1 : T(X) \to T(X)$ of the diffeomorphism $f_1 : X \to X$ that corresponds to $1 \in \mathbb{Z}$ is hyperbolic.

Dynamic Expansiveness [27]. Uniform expansiveness of $Df_1 : T(X) \to T(X)$ implies expansiveness of Anosov $\mathbb{Z}$-actions, where a family of transformations $f_i : X \to X$, $i \in I$, of a metric space (X, d) is called expansive, if

$$\sup_{i \in I} d(f_i(x), f_i(y)) \geq \epsilon$$

for all x, y with $x \neq y$ and some $\epsilon = \epsilon(x) > 0$.

If X is a compact space and $f_i = f^i, i \in I = \mathbb{Z}$, are powers of a homeomorphism $f : X \to X$, then expansiveness of $\{f_i\}$ is equivalent to exponential expansiveness. Further, uniform expansiveness of $Df : T(X) \to T(X)$ implies exponential expansiveness of f.

Hyperbolicity of Diffeomorphisms [27]. Let $f : X \to X$ be a diffeomorphism on a continuously differentiable manifold X. Hyperbolicity of f is defined by corresponding properties of $Df : T(X) \to T(X)$. Let $L \to X$ be a vector bundle with fibers denoted by $L_x \in L$, $x \in X$. Let $F : L \to L$ be a continuous fiberwise linear map with vector spaces $F^{-1}(x) = L_x$ corresponding to $x \in X$. Let the self map induced by F be $\underline{F} : X \to X$. Let $F = \{F_x : L_x \to L_{\underline{F}(x)}\}_{x \in X}$.

Let the linear space L_x be endowed with norms. F is hyperbolic, if there is an $\epsilon > 0$, constants $c, c' > 0$ and a splitting

$$L = L_{contr} \oplus L_{exp},$$

where L_{contr} and L_{exp} are F- invariant subbundles in L, such that $\forall i \in \mathbb{N}$:

$$\left\|F^i(l)\right\| \leq c(1-\epsilon)^i \left\|l\right\| \forall l \in L_{contr}$$

and

$$\left\|F^i(l)\right\| \geq c'(1+\epsilon)^i \left\|l\right\| \forall l \in L_{exp}.$$

Hyperbolicity of F implies uniform expansiveness of F. Hence, $\exists N \in \mathbb{Z}$ with

$$max(\left\|F^N(l)\right\|, \left\|F^{-N}(l)\right\|) \geq 2 \left\|l\right\| \forall l \in L.$$

If X is compact and L_x are finite dimensional, then hyperbolicity does not depend on the norms in L_x. Further, small perturbations $F_\epsilon : L \to L$ remain hyperbolic. Compare these observations to the shadowing lemma above. Additionally, notice the extension to locally split hyperbolic structures of the state space X given by splitting each tangent space L_x into contracting and expanding subspaces. Thus, it generalizes the linear toy example in section 2.3 to a broader class of systems.

Anosov foliations [27]. The Anosov sub-bundles $T_{contr}, T_{exp} \in T(X)$ are integrable. Hence, they are tangent bundles of foliations S_{contr}, S_{exp} that are partitions of X into C^1-smooth submanifolds, called contracting (stable) and expanding (unstable) leaves of dimensionality equals to the ranks of the bundles T_{contr}, T_{exp}.

Thus, S_{contr}, S_{exp} could serve as boundaries of Markov rectangles. Their general definition expands the construction of Markov partitions to systems characterized by Anosov diffeomorphisms. However, there are open questions regarding the universality of Anosov foliations. Assuming X to be locally split hyperbolic everywhere is a rather strict condition. It is unclear if this is a universal property for various real-world examples of dynamic systems (X, f).

First, there are differentiable manifolds X for which there are no Anosov diffeomorphisms. A simple example is an n-dimensional sphere $\mathbb{S}^n$. However, some manifolds provide a well-behaved basis for Anosov diffeomorphisms. The most prominent one is that of n-dimensional tori $\mathbb{T}^n$ [27].

Second, in general, the classification of manifolds that admit Anosov diffeomorphisms is not yet rigorously solved. Only infranil manifolds are known to admit Anosov diffeomorphisms. Infranil manifolds are equivalent to almost flat manifolds (Gromov–Ruh theorem). Moreover, it is conjectured that this class of infranil manifolds is indeed the only class of manifolds admitting Anosov diffeomorphisms [27]. If this conjecture holds, Ansosv diffeomorphisms' existence could restrict to a limited subset of differentiable manifolds.

Although Anosov foliations could serve as boundaries of Markov partitions, one has to analyze how to arrange these boundaries. See example 6.5.10. in [44], here two foliations (one contracting and one expanding eigenvector direction) exist. It is not obvious how to determine the rectangles' dimensions. It only appears to be evident that the rectangle boundaries are in the direction of the (locally) expanding and contracting leaves. However, this observation does not provide enough information to create rectangles that fulfill the intersection property. Further, it is not clear how to algorithmically find such Anosov foliations.

Construction of Anosov foliations [27]. Ansosv foliations can be characterized in two ways. Let X be endowed with a metric $dist$.

1. The contracting leaves are defined by the equivalence relation:

$$x \sim y \Leftrightarrow \lim_{i\to\infty} dist(f_{\mathbf{1}}^i(x), f_{\mathbf{1}}^i(y)) \to 0.$$

The expanding leaves are defined by the equivalence relation:

$$x \sim y \Leftrightarrow \lim_{i\to\infty} dist(f_{\text{-}\mathbf{1}}^i(x), f_{\text{-}\mathbf{1}}^i(y)) \to 0.$$

2. Let the map $f_{[\delta]}$ act on the families of subsets

$$\{A_x \subset X\}_{x\in X},$$

where the subsets $A_x \subseteq B_x(\delta)$. Further, let

$$f_{[\delta]}(A_x) = f(A_x) \cap B_{f(x)}(\delta).$$

If $f = f_{\mathbf{1}}$, $1 \in \mathbb{Z}$, is an Asonov diffeomorphism and X a compact manifold, then for all small $\delta > 0$, $0 < \epsilon \leq \delta$, the families of subsets in X,

$$\{f_{[\delta]}^i(B_x(\epsilon)) \subset X\}_{x\in X}$$

converge for $i \to \infty$ to the connected components of the expanding leaves through the points $x \in X$ within the balls $B_x(\delta)$. Equivalently, the same construction, but with $f^{-1} = f_{\mathbf{1}}^{-1}$, converges for $i \to \infty$ to the connected components of the contracting leaves through the points $x \in X$ within the balls $B_x(\delta)$.

1. gives a general equivalence relation characterizing Anosov foliations. Only evaluating $dist(f_{\mathbf{1}}^i(x), f_{\mathbf{1}}^i(y))$ for some $x, y \in X$ at $i \to \infty$ is required. However, the convergence speed is unknown. Furthermore, one cannot quantify the error by the approximation of $x \sim y$. Last, for continuous X with infinitely many pairs $x, y \in X$, this computation is intractable.

In **2.**, it is a priori unclear which $0 < \epsilon \leq \delta$ is sufficiently small to converge to connected components of contracting and expanding leaves. Also computing $f(A_x)$ is extremely inefficient in a computer, if f is non-linear and A_x non-convex. In general, this approach is topologically motivated, such that set operations are potentially infeasible to implement in a computer for complex geometries.

Structural Stability of smooth locally split hyperbolic systems [27]. Let X and Y be $\mathbb{Z}$-spaces (the group $\mathbb{Z}$ continuously acts on X and Y), where Y is endowed with a metric. $Q : X \to Y$ is an ϵ-quasi-morphism, if the generators of the $\mathbb{Z}$-actions in X and Y, given by $f : X \to X$ and $g : Y \to Y$, satisfy

$$dist_Y(Q \circ f(x), g \circ Q(x)) \leq \epsilon \; \forall x \in X.$$

Let f and the $\mathbb{Z}$-action generated by f (given by f^i for some $i \in \mathbb{Z}$) admit a unique local shadowing, if there is a $\epsilon_0 = \epsilon_0(Y, f) > 0$ and $\delta = \delta(\epsilon) \to 0$, such that every ϵ-quasi-morphism for $\epsilon \leq \epsilon_0$ is δ-close to a unqiue $\mathbb{Z}$-morphism $M : X \to Y$ with $M \circ f = g \circ M$.

Compare to the shadowing lemma for $\mathbb{Z}$-quasimorphisms on X and Y with $T : Y \to Y$ being hyperbolic. Such an ϵ-quasimorphism essentially captures the same notion of local stability. It only lacks global stability since the boundedness of distances is tight by $\epsilon < \epsilon_0$ and $\delta(\epsilon) \to \infty$.

If Y is a complete metric space and g a homeomorphism that has invariant splittings,

$$Y = Y_{contr} \times Y_{exp} \; and \; g = g_{contr} \times g_{exp},$$

then g satisfies the global shadowing property for all arbitrarily large $\epsilon > 0$, which is a significant dependency reduction of ϵ and δ on Y and f. Also, this structural stability is now ensured on a global scale of the systems (X, f) and (Y, g).

If all small neighborhoods $U \subset Y$ have invariant $U_{contr} \times U_{exp}$ splittings, then the local shadowing property for (Y, g) is ensured which matches the definition of locally split hyperbolic systems.

Anosov Shadowing Lemma [27]. Anosov $\mathbb{Z}$-actions admit unique local shadowings.

Hence, small $U \subset Y$ split into products of contracting and expanding leaves. However, this is again a qualitative statement without precisely knowing how to compute such leaves. Nevertheless, the lemma introduces the main theorem about the structural stability of systems described by Anosov diffeomorphisms.

Structural stability of locally split hyperbolic systems - Theorem [27]. Let Y be a compact C^1-smooth manifold, $g : Y \to Y$ be an Anosov C^1-diffeomorphism and $f : Y \to Y$ be a C^1-diffeomorphism (here $X = Y$). If f is sufficiently close to g in C^1-topology, then there is a homeomorphism $M : Y \to Y$ that is C^0-close to g, such that $f \circ M = M \circ g$.

This theorem provides global structural stability of Anosov systems in the same fashion, as seen in the shadowing lemma. To ensure g and f inducing the same global dynamics characterized by the true morphism M, one only requires g to be sufficiently close to f.

3.3 From Differentiable to Combinatorial Systems

This section introduces Cayley graphs to transform continuous dynamic systems into combinatorial systems, also known as shifts of finite type. It is clarified how the global structural stability of Anosov systems maps to combinatorial systems. Further, these results are to symbolic dynamics.

Definition Cayley graph [27]. Let G be a group with a generating set S. The Cayley graph $\Gamma = \Gamma(G, S)$ is a colored, directed graph, s.t.

- $\forall g \in G : \exists! v \in V(\Gamma) : v = g$, where $V(\Gamma)$ is the vertext set of Γ,
- $\forall s \in S$: there is a color c_s assigned to s,
- $\forall g \in G, s \in S : \exists! e \in E(\Gamma) : e = (g, g \cdot s)$ with color c_s, where $E(\Gamma)$ is the edge set of Γ.

Definition Markov Shifts [27]. Let G be a Cayley graph of the semigroup $(\mathbb{N}, +)$ generated by $S = \{1\}$, $1 \in \mathbb{N}$, or by the group $(\mathbb{Z}, +)$, $1 \in \mathbb{Z}$. Let the spaces of combinatorial graph maps be $G^{\rightsquigarrow} = G^{\mathbb{N}}$ and $G^{\leftrightsquigarrow} = G^{\mathbb{Z}}$ that are represented by spaces of marked edge-paths in G, which are one sided future directed and two sided double infinite respectively.

In the case of shifts of finite type, one assumes G to be finite. Hence $G^{\rightsquigarrow}$ and $G^{\leftrightsquigarrow}$ are compact spaces since the number of combinatorial graph maps over a finite graph is finite again. This shift space is realized as a space of maps $F^{\mathbb{Z}}$ from $\mathbb{Z}$ to a finite set F. Since Cayley graphs are utilized to characterize shifts of finite type, one takes the sets of edges in G for F:

$$G^{\leftrightsquigarrow} \subset E(G)^{\mathbb{Z}}.$$

The finiteness of these types of shifts relates to the finiteness of the number of conditions encoded by the combinatorial maps of G. Shifts of finite type are defined

by open and closed $U \subset F^{\mathbb{Z}}$. $Y \subset F^{\mathbb{Z}}$ is a shift of finite type, if

$$Y = \cap_{g \in \mathbb{Z}} g(U)$$

for some $U \subset F^{\mathbb{Z}}$, which is the pullback of a $S \subset F^I$ under the restriction map $F^{\mathbb{Z}} \to F^I$ for some finite $I \subset \mathbb{Z}$. Since $U \subset F^{\mathbb{Z}}$ is open and closed, Y is dynamically isolated. Thus, shifts of finite type can be defined as dynamically isolated closed invariant subsets in $F^{\mathbb{Z}}$. Since the diagonals $G^{\leftrightsquigarrow} \times G^{\leftrightsquigarrow}$ are also of finite type, shifts of finite type are expansive.

Local Shadowing [27]. Let X be a topological space, $f : X \to X$ a continuous map and $\mathcal{U} = \{U\}_{U \subset Y}$ be a covering of Y, where U are open. Let $G = G(\mathcal{U}, f)$ be the graph on the vertex set $\mathcal{U}$, where U_1 is joint with U_2 by an edge (U_1, U_2), whenever $f(U_1) \cap f(U_2) \neq \emptyset$. Note that the orbits of the $\mathbb{Z}$-action on $G(\mathcal{U}, f)^{\leftrightsquigarrow}$ are $\mathbb{Z}$-families $\{U_i \in \mathcal{U}\}_{i \in \mathbb{Z}}$, such that $U_i \cap f^{-1}(U_{i+1}) \neq \emptyset \; \forall i \in \mathbb{Z}$.

Let Y be compact and f is a homeomorphism, which generates a $\mathbb{Z}$-action on Y. The local shadowing property of f can now be stated as follows. Given an open cover $\mathcal{V}$ of X, there is a finite open cover $\mathcal{U}$ of X and a surjective $\mathbb{Z}$-morphism

$$\phi : G(\mathcal{U}, f)^{\leftrightsquigarrow} \twoheadrightarrow (X, f),$$

such that the orbits $\{U_i\}$ of the $\mathbb{Z}$-action are $\mathcal{V}$-close to the corresponding orbits $x_i = f^i(x_0) \in X$ of the $\mathbb{Z}$-action on X. Hence, $\forall i \in \mathbb{Z}, U_i \in \{U_i\} : \exists ! : V \in \mathcal{V} : x_i \in V \wedge U_i \subset V$, where $x_i \in Y$ and $U_i \subset X$, since they are $\mathcal{V}$-close to each other. Next, we bridge the gap between the local shadowing lemma for Anosov systems and shifts of finite type.

Combinatorial Shadowing in Markov Shifts [27]. Let $\phi_i : \{i, i+1, ..., i+k\} \to G, i = ..., -2, -1, 0, 1, 2, ...$ be a sequence of combinatorial maps that are paths in G of length k. Then, there is a unique conbinatorial map $\Phi : \mathbb{Z} \to G$, such that the restriction of Φ to $\{i+3, ..., i+k-2\} \subset \mathbb{Z}$ is equal to the restriction of ϕ_i to these subsets for all $i = ..., -2, -1, 0, 1, 2, ...$ This property induces the following remarkable observations.

Observation 1 [27]. Compact manifolds with Anosov $\mathbb{Z}$-actions and compact spaces with locally split hyperbolic homeomorphisms are both representable as quotients of shifts of finite type.

Observation 2 [27]. Compact spaces X with locally split hyperbolic homeomorphisms $f : X \to X$ admit arbitrary finite open covers $\mathcal{U}$, such that the surjective morphism $\phi : G(\mathcal{U}, f)^{\leftrightsquigarrow} \to (X, f)$ is finite-to-one.

These two observations show that Anosov systems and locally split hyperbolic systems can be transformed into shifts of finite type as introduced in chapter 2. Further, such systems are topological conjugate by the existence of the homeomorphism ϕ. A symbolic representation exist for all such systems. By a suitable choice for the cover $\mathcal{V}$, one guarantees that these shifts of finite type are unique up to finitely many alternatives. These results precisely match the observations for shifts of finite type in chapter 2. Moreover, locally split hyperbolic systems are crucial in constructing Markov partitions and shift spaces. Besides the assumption of local split hyperbolicity, the partition $\mathcal{V}$ must be sufficiently fine. Thus, one still lacks a Markov partition construction procedure.

3.4 On Markov Partitions to Construct Symbolic Systems

In this section, Markov partitions are introduced in a rigorous topological fashion based on the work of [27] and in line with previous results from section 2.3. Further, Markov partition examples of expanding and hyperbolic systems are presented. Last, a topologically motivated Markov partition construction method is stated. However, this construction work is not feasible for implementation on a computer. Nevertheless, it deepens the understanding of Markov partitions.

Definition Partition [27]. A partition is a covering of a topological space X by closed subsets V_i, which are fat. V_i is fat, if $int(V_i) = V_i \setminus \partial V_i$ is dense in V_i and $\forall V_i \neq V_j, U_i = int(V_i), U_j = int(V_j) : \ U_i \cap U_j = \emptyset$. The definition is identical with topological partitions in section 2.3.

Definition Markov partition [27]. A partition $\{V'_j\}_{j\in J}$ refines $\{V_i\}_{i\in I}$, if there is a $\iota : J \to I$, such that V'_j is contained in $V_{\iota(j)} \forall j \in J$. Assume f to be a locally expanding map of a metric space. Then, the Markov property of the partition $\{V_i\}$ holds, if the pullback partition $\{f^{-1}(V_i)\}$ refines $\{V_i\}$. Intuitively, the refinement of $\{V_i\}$ by $\{f^{-1}(V_i)\}$ is the topological equivalent of the geometrically motivated intersection property in section 2.3. Additionally, the pullback f^{-1} is leveraged instead of the f . Both definitions coincide, but the definition at hand generalizes the Markov property by a topological argument.

Examples of Markov Partitions [27] . Let $f : X \to X$ be a globally expanding map, where X is a metric space. Let $X = \mathbb{R}^n$ and $f : x \mapsto 2x$. Create a partition of X in n-dimensional unit cubes and note that this partition $\{V_i\}$ is Markov.

If $f : X \to X$ is a strictly expanding linear map, then a Markov partition is made of convex polyhedral domains V_i with $diam(V_i) \leq 1$ and $inrad(V_i) \geq \epsilon > 0$.

However, Markov partitions for strictly expanding endomorphisms f on the n-torus $\mathbb{T}^n$ are rarely so simple. The construction of such a Markov partition, in general, delivers fractal subsets V_i. This phenomenon is unavoidable for the majority of f and in particular for all tori of dimension $n \geq 3$ [17].

Construction of Markov partitions for locally expanding maps [27]. Let $f : X \to X$ be a locally expanding map and $\{U_i\}_{i \in I}$ be a collection of open subsets in a topological space X, $U' \subset X$. Let I be endowed will a well-ordered structure and let $i_{min} = i_{min}(U') \in I$ be the minimal i for which $U_i \cap U' \neq \emptyset$.

Construct $\{U_i^{new}\}$ as following,

$$U_i^{new} = U_i \cup U' \text{ if } i = i_{min}$$

and

$$U_i^{new} = int(U_i \setminus U') = U_i \setminus \bar{U'} \text{ if } i \neq i_{min}.$$

Apply this to a all U_j' of a given collection $\{U_j'\}$ (and not only to U'). Hence, obtain $\{U_i^{new}\}$, denoted by $\{U_i^{new}\} = \{U_i\} \swarrow \{U_j'\}$. $\{U_i^{new}\}$ is obtained by attaching U_i to those U_j' for which $i = i_{min}(U_j')$ and by subtracting from U_i the closures of the other U_j'.

Let $\mathcal{V}_{I\prec}$ be the set of partitions $\{V_i\}_{i \in I}$ of X. Given a continuous map $f : X \to X$, let

$$\Upsilon_f : \mathcal{V}_{I\prec} \to \mathcal{V}_{I\prec}$$

be defined by $U_i = int(V_i)$ as follows

$$\Upsilon_f : \{U_i\} \mapsto \{U_i\} \swarrow \{f^{-1}(U_i)\}.$$

The pullback partition $\{f^{-1}(V_i)\}$ that refines $\{V_i\} \in \mathcal{V}_{I\prec}$ is then given as $\{V_i\}$ being the fixed point of the map $\Upsilon_f : \mathcal{V}_{I\prec} \to \mathcal{V}_{I\prec}$.

If f is a locally strictly expanding map and X is compact, then Υ_f is contracting regarding the Hausdorff metric in $\mathcal{V}_{I\prec}$. Since Υ_f is monotone, Banach's fixed point theorem applies [27].

From an algorithmic point of view, this is facilitation, since repeatedly applying Υ_f to an arbitrary $\{V_i\}_{i \in I}$ starting point in $\mathcal{V}_{I\prec}$ is guaranteed to converge to a Markov partition. Further, one observes that the forward periodic points that are fixed points of powers $\Upsilon_f : \mathcal{V}_{I\prec} \to \mathcal{V}_{I\prec}, i \in \mathbb{N}$, are dense in $\mathcal{V}_{I\prec}$. Therefore, these fixed points exist frequently in $\mathcal{V}_{I\prec}$. However, constructing this algorithm in a computer would require an efficient way to compute intersections between probably geometrically complex objects to represent sets V_i and $f^{-1}(V_j)$. Thus, implementing such a topologically motivated algorithm could be intractable.

Markov Partitions for Split Hyperbolic Actions [27]. Let $f : X \to X$ be a locally split hyperbolic homeomorphism. Let $\{V_i\}$ be a partition of X. $\{V_i\}$ is Markov, if the following two conditions hold:

$$(1)\ \forall V_i \subset X : V_i = V_i^{exp} \times V_i^{contr} \subset X,$$

where f^{-1} strictly contracts $V_i^{exp} \times v^{contr} \subset X\ \forall v^{contr} \in V_i^{contr}$ and f contracts $v^{exp} \times V_i^{contr}\ \forall v^{exp} \in V_i^{exp}$.

(2) If $L = L^{exp} \subset X$ is an expanding leaf, then the partition $\{L \cap V_i\}$ of this leaf by its intersection with V_i is refined by the pullback partition $\{f_{|L}^{-1}(L)\}$, where $f_{|L} : L \to X$ is the restriction of f to the expanding leaf L. Symmetrically, if $L = L^{contr}$, then $\{L \cap V_i\}$ is refined by the images of $\{f^{-1}(L) \cap V_i\}$ under $f_{|f^{-1}(L)}$.

One can adjust the argument of finding a fixed point of Υ_f to receive a Markov partition to the split hyperbolic partition $\mathcal{V}_{I\prec}^{\times} = \mathcal{V}_{I\prec}^{exp} \times \mathcal{V}_{I\prec}^{contr}$. Apply Υ_f as above to the expanding part $\mathcal{V}_{I\prec}^{exp}$ of the split hyperbolic partition. Further, apply $\Upsilon_{f^{-1}}$ to the contracting part $\mathcal{V}_{I\prec}^{contr}$. Again, one ends up with fixed points for Υ_f and $\Upsilon_{f^{-1}}$ that are dense in $\mathcal{V}_{I\prec}^{\times}$ [27].

Association between Markov partitions and graphs [27]. Let $G = G(\{V_i\}, f)$ be a graph associated with the Markov partition $\{V_i\}_{i\in I}$ of space X and a $\mathbb{Z}$-generated homeomorphism $f : X \to X$. $V(G)$ is the vertex set given by I. $E(G)$ is the edge set, such that pairs $i, j \in I$ with $f(V_i) \cap V_{i+1} \neq \emptyset$ define the edges.

If $\{V_i\}$ is sufficiently fine, then by Anosov's shadowing lemma, one receives a $\mathbb{Z}$-morphism from the space of double infinite paths in G to orbits in (X, f),

$$\phi : G(\{V_i\}, f)^{\leftrightsquigarrow} \to X.$$

Hence, global structural stability also applies for the shift of finite type induced by the partition $\{V_i\}$. Since $\{V_i\}$ is Markov, ϕ is finite-to-one.

Furthermore, by the Rufus Bowen theorem, the existence of Markov partitions for an even broader class of Axiom A systems is guaranteed [18]. Note that Anosov systems are a particular case of Axiom A systems. Even though the question about Markov partitions' existence and nature is extensively analyzed, one still lacks a well-formalized and tractable algorithmic procedure to construct Markov partitions automatically on a computer.

4 Algorithmic Construction of Markov Partitions

In this chapter, we build Markov partitions for given dynamic systems. Construction algorithms based on the findings from chapter 3 applicable to a broad class of dynamic systems are developed. We start by implementing the example introduced in section 2.3. Therefore, we reflect on the challenges and limitations of the approach proposed by [51]. Subsequently, we extend these semi-automated procedures for finding such Markov partitions to a unified, well-formalized, and automated Markov partition construction method extending research work of [24].

4.1 Implementation of Markov Partitions for a Toy Example

Consider the dynamic system $(\mathbb{T}^2, \phi)$ described at the end of section 2.3. The main goal of this section is to find an algorithmic way to construct and validate the arithmetically calculated Markov partition $P = \{P_1, P_2, P_3\}$.

As already observed, for $\phi(s,t) = (s+t, s)\ (mod\ 1)$, there is a $A \in \mathbb{Z}^{2\times 2}$ with

$$A\begin{pmatrix} s \\ t \end{pmatrix}\ mod\ 1 = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} s \\ t \end{pmatrix}\ mod\ 1 = \begin{pmatrix} s+t \\ s \end{pmatrix}\ mod\ 1 = \phi(s,t),$$

where $s, t \in \mathbb{R}$. Additionally, define the quotient group $\mathbb{T}^2 = \mathbb{R}^2/\mathbb{Z}^2$ by identifying some $(s,t) \in \mathbb{T}^2$ with the coset $(s,t) + \mathbb{Z}^2 = \{(s,t) + (a,b) : a \in \mathbb{Z}, b \in \mathbb{Z}\}$. Hence, $(s,t) = (s',t')$, if $(s,t) - (s',t') \in \mathbb{Z}^2$. Thus, coordinate-wise addition over $\mathbb{T}^2$ is equivalent to addition of cosets in $\mathbb{R}^2/\mathbb{Z}^2$. This is why the quotient mapping $q : \mathbb{R}^2 \to \mathbb{T}^2$ is defined as $q((s,t)) = (s,t) + \mathbb{Z}^2$. It holds:

$$q \circ A = \phi \circ q,\ q(Ax) = \phi(q(x)),\ x \in \mathbb{R}^2.$$

Thus, one can work equally well in the euclidean plane or on the torus. Moreover, the system $(\mathbb{T}^2, \phi)$ is a locally split hyperbolic system since the the matrix A induces hyperbolic system dynamics. For each eigenvalue λ of A, it holds $|\lambda| \neq 1$, which implies hyperbolicity. Since $|det(A)| = 1$, the dynamic system is invertible.

Since A is hyperbolic, there are contracting and expanding segments in $\mathbb{R}^2$, as seen in section 2.3. These properties are sufficient to construct Markov partitions.

Further, these segments are crucial to meet the intersection property of Markov rectangles introduced in section 2.3.

We observe the following:

$$|\lambda_i| < 1 \Leftrightarrow v_i \text{ is a basis vector for contracting segment}$$

and

$$|\lambda_i| > 1 \Leftrightarrow v_i \text{ is a basis vector for expanding segment,}$$

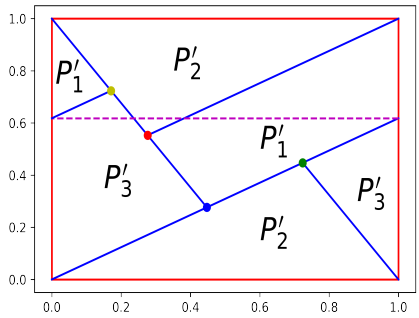
where λ_i is the i-th eigenvalue of A and v_i the corresponding eigenvector. Hence, these contracting and expanding spaces are one-dimensional for $(\mathbb{T}^2, \phi)$.

The crucial question is how to construct a partition $\mathcal{P}$ of $\mathbb{T}^2$, such that $\mathcal{P}$ meets the intersection property for all $P_i, P_j \in \mathcal{P}$ with $\phi(P_i) \cap P_j \neq \emptyset$. Namely such a $\phi(P_i) \cap P_j$ has to cut completely through P_j along the expanding segment, must not contain expanding boundaries of P_j and does not not completely contain the boundaries of P_j along the contracting segments. We make some qualitative observations for the given Markov partition in Figure 2.2.

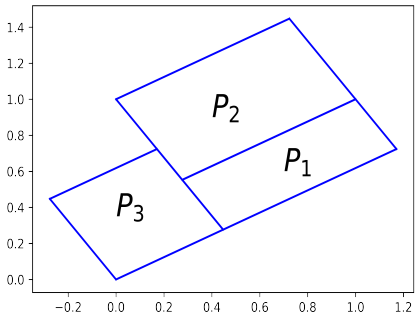
1. All $P_i \in \mathcal{P}$ have boundaries parallel to the expanding and contracting eigenvector directions.
2. Since the system is linear, a boundary parallel to eigenvector v_1 is always expanding, independent of the translation to the origin $0 \in \mathbb{R}^2$. Symmetrically, all boundaries parallel to v_2 are contracting.
3. Putting together observation 1. and 2., each P_i has boundaries that will further expand or contract under iterative application of ϕ.
4. Each P_i is a curvilinear rectangle since eigenvectors of real symmetric matrices are always orthogonal. Defining P_i to be oriented parallel to the eigenvectors guarantees the boundaries to be orthogonal to each other.
5. It is unclear how "long" or "short" boundaries in expanding and contracting direction have to be, but the Markov partition must be sufficiently fine-grained.

Next, we make these observations precise by constructing Markov partitions $\mathcal{P}$ for $(\mathbb{T}^2, \phi)$ as visualized in Figure 2.1. Our remarks are guided by [51]. The stated observations reduce to computing intersection points between expanding and contracting segments. Hence, the torus quotient simplifies the construction.

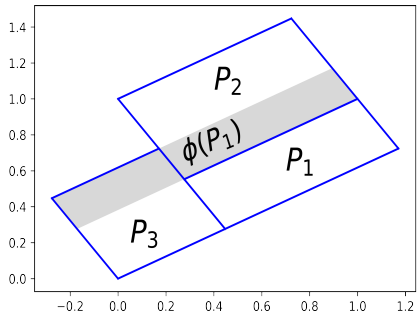
The torus $\mathbb{T}^2$ is identified with the unit square $[0, 1]^2 \subset \mathbb{R}^2$. We refer to the first eigenvector v_1 of A as the unstable (expanding) segment and to the second



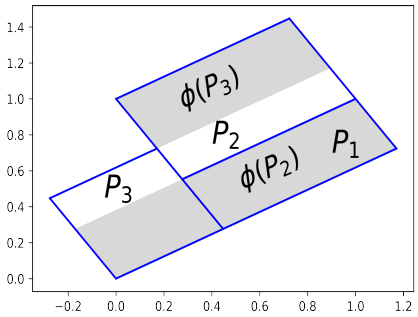
(a) Markov partition for dynamic system $(\mathbb{T}^2, \phi)$.



(b) Same Markov partition with identification in $\mathbb{R}^2$.



(c) System dynamics ϕ applied to the Markov rectangle P_1.



(d) System dynamics ϕ applied to the Markov rectangles P_2 and P_3.

Figure 4.1: (a) Markov partition for the dynamic system $(\mathbb{T}^2, \phi)$ by eigendecomposition of A and given identification between the euclidean plane and the torus by q (see the purple line for identification symmetries). The partition is mainly defined by the unit square's vertices (red lines) and the drawn intersection points (green, blue, red, yellow) between stable and unstable directions. (b) is the same Markov partition in the euclidean plane by gluing split P'_i together along the unit square edges. (c) + (d) are system dynamics ϕ applied to the Markov partition for one discrete time step of the system $(\mathbb{T}^2, \phi)$. Note that the partition is indeed a Markov partition since it fulfills the intersection property.

eigenvector v_2 of A as the stable (contracting) segment. One starts from the unit square's vertices and compute intersections of the lines following the stable and unstable directions. Figure 4.1c and Figure 4.1d show that the resulting partition is indeed a Markov partition. See Figure 4.1a for the construction work and compare to Figure 2.1.

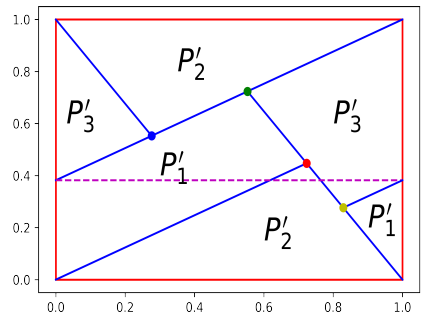
Precisely, start by drawing a line $l_{(0,0)}$ from point $(0,0) \in [0,1]^2$ in the direction of the unstable segment. Since the system dynamics are linear, the stable and unstable directions are independent of the line's actual origin. Hence, draw a line with origin $(0,0)$ in the direction of v_1 that completely cuts through the unit square. With the same line of reasoning, produce lines $l_{(1,0)}, l_{(0,1)}$ in the stable direction v_2. Both line endpoints of $l_{(1,0)}$ and $l_{(0,1)}$ are given by the (green and blue) intersection points with $l_{(0,0)}$. Next, produce a line $l_{(1,1)}$ again in unstable direction with an endpoint equal to the (red) intersection point with $l_{(0,1)}$. Last, extend the line $l_{(0,0)}$ by identification. The purple helper line visualizes these identification symmetries. The extension of $l_{(0,0)}$ ends in the (yellow) intersection point with $l_{(0,1)}$.

The Markov partition is equivalent to the shift of finite type characterized by the adjacency matrix $B = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$. To receive B, analyze the intersections $\phi(P_i) \cap P_j$, $i, j \in [n], n = 3$. Results are visualized in Figure 4.1c and Figure 4.1d.

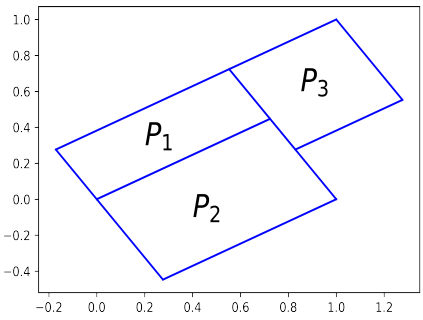
Further, the dynamic system described by A and the symbolic system defined by B are topologically equivalent. For example, compute the topological entropy by taking the logarithm of the largest eigenvalue of A and B. Both matrices have the same largest eigenvalue and hence the same topological entropy. Since entropy is an invariant, it is a strong indicator of a proper Markov partition.

However, there are open questions regarding the construction process. First, does it make a difference whether one first chooses $l_{(0,0)}$ or $l_{(1,1)}$ to cut through the whole unit square? Second, does one necessarily have to start with a branch in the unstable direction to produce a valid Markov partition, or could one also start selecting a branch in the stable direction? Third, why does the tracing of stable and unstable segments start at the unit square's vertices? In general, for a linear hyperbolic system, the direction of (un)stable segments does not depend on the origin.

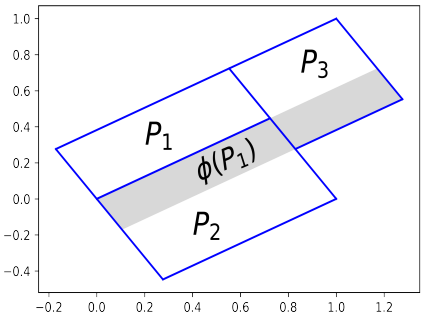
For the in-depth answer to the last question, we refer to section 4.2. In general, one starts such constructions of rectangle boundaries along both branches of stable and unstable manifolds in a hyperbolic fixed point of the dynamic system. For linear hyperbolic systems, a hyperbolic fixed point $z^* \in \mathbb{T}^2$ is always $z^* = 0$. Together



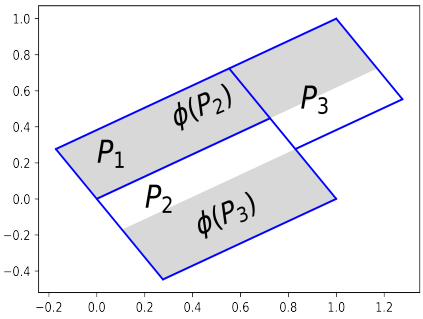
(a) Markov partition for dynamic system $(\mathbb{T}^2, \phi)$.



(b) Same Markov partition with identification in $\mathbb{R}^2$.

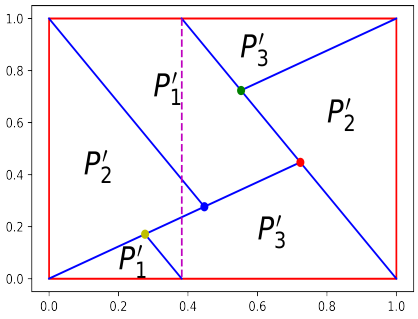


(c) System dynamics ϕ applied to the Markov rectangle P_1.

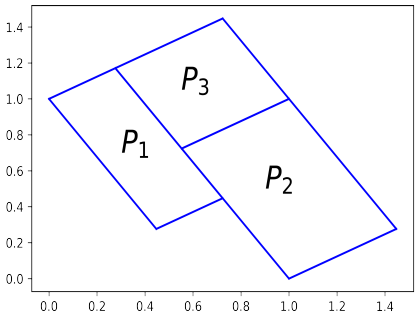


(d) System dynamics ϕ applied to the Markov rectangles P_2 and P_3.

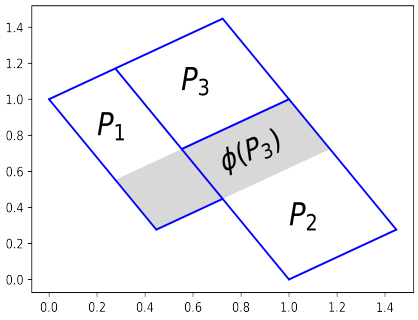
Figure 4.2: (a) + (b) show the same Markov partition construction as in Figure 4.1, but starting with drawing $l_{(1,1)}$ instead of $l_{(0,0)}$. Interchanging the starting unstable branches in z^* does not change the partition, but only mirrors it. Hence, the partition is Markov as shown in (c) + (d).



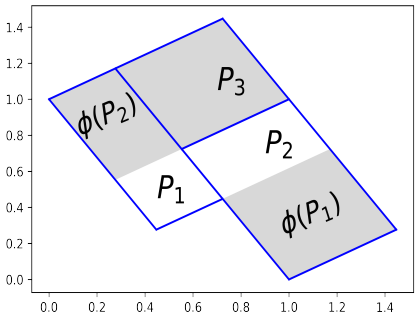
(a) Markov partition for dynamic system $(\mathbb{T}^2, \phi)$.



(b) Same Markov partition with identification in $\mathbb{R}^2$.



(c) System dynamics ϕ applied to the Markov rectangle P_3.



(d) System dynamics ϕ applied to the Markov rectangles P_1 and P_2.

Figure 4.3: (a) + (b) show a Markov partition where the construction is started with drawing $l_{(1,0)}$ instead of a branch in unstable direction. Notice that the partition is rotated by $90°$ compared to Figure 4.1. (c) + (d) are system dynamics ϕ applied to the the Markov partition once for one discrete time step of the system $(\mathbb{T}^2, \phi)$. Further, all $\phi(P_i) \cap P_j$ indeed fulfill the intersection property and hence the partition at hand is indeed a Markov partition.

with the identification z^* on the torus, $[z^*] \sim \{\begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}1\\1\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}0\\0\end{pmatrix}\}$, it is clear why the tracing starts at the vertices of the unit square.

To answer the first question, repeat the construction process by starting with $l_{(1,1)}$ instead of $l_{(0,0)}$. The result can be seen in Figure 4.2a. Recognize that interchanging the starting unstable branch in z^* leads to the same Markov partition up symmetry. Since the torus is identified with the unit square's edges, the partition is again Markov as shown in Figure 4.2c and Figure 4.2d. Further, the adjacency matrix B characterizing the induced shift of finite type is identical.

Next, start with drawing $l_{(1,0)}$ in unstable direction across the unit square and then proceed as usual. The resulting partition can be seen in Figure 4.3a. Since both eigenvectors along the stable and unstable segments are orthogonal to each other, the partition is rotated $90°$ counterclockwise compared to Figure 4.1. Moreover, by analyzing Figure 4.3c and Figure 4.3d, one sees that the partition indeed fulfills the intersection property and thus is Markov. The adjacency matrix T of the shift of finite type is

$$T = \begin{pmatrix}0 & 1 & 0\\ 1 & 0 & 1\\ 1 & 1 & 0\end{pmatrix} = B^T.$$

This relation is intuitive, since the partition in Figure 4.3a is just a $90°$ rotation of the partition in Figure 4.1a, which corresponds to a transposed matrix B. Further, the eigendecompositions $B = V\Lambda V^T$ and $T = B^T = (V\Lambda V^T)^T = V^T\Lambda V$ have the same eigenvalues. Therefore, both Markov partitions induce symbolic systems with the same topological entropy.

Overall, all three partitions are indeed Markov. Further, they share essential topological invariants and lead to equivalent shifts of finite type. We conclude that constructing Markov partitions is not about the order in which we trace stable and unstable branches. Merely, finding a hyperbolic fixed point z^* of the dynamic system and then tracing branches until intersection points between stable and unstable branches starting in z^* is crucial.

4.2 Algorithmic Construction of Markov Partitions

We formalize and implement an algorithmic framework to automatically construct Markov partitions for a broad class of dynamic systems. Therefore, we extend the proposed method by [24] to reduce manual computational effort and user interaction. Essential parts of [24]'s approach rely on visual feedback from a video-graphic device. Further, non-trivial implementation details are not explained. We adopt

the procedure such that an automated application to two-dimensional locally split hyperbolic systems is facilitated. Additionally, we precisely formalize and justify the proposed algorithm. Also, we provide a modular and easy-to-use implementation. [1]

Overview. The first method by [24] is conceptually related to the construction in section 4.1. In its core, it also traces stable and unstable manifolds and computes intersection points that define the vertices of the Markov rectangles. Moreover, the algorithm is guaranteed to meet the intersection property by creating sufficiently fine partitions.

Assumptions. Let (Ω, S) be a dynamic system. Ω is a two-dimensional system with locally split hyperbolic system dynamics S. Further, a (numerically) accessible hyperbolic fixed point, whose stable and unstable manifolds separately cover Ω densely, exists.

Let z^* be a hyperbolic fixed point with W^u and W^s being the unstable and stable manifolds in z^* respectively. Assume $\bar{W}^u = \Omega$ and $\bar{W}^s = \Omega$ such that all assumptions are met.

Notation. If $x \in W^s$, let W^s_x be the segment of W^s between z^* and x for any $x \in \Omega$. Analogous notation holds for $y \in W^u$.

If $x_1, x_2 \in W^s$ symmetrically placed around z^* (in the sense of a point-symmetry), then

$$S(W^s_{x_1}) \subset W^s_{x_1} \text{ and } S(W^s_{x_2}) \subset W^s_{x_2} \tag{4.1}$$

or

$$S(W^s_{x_1}) \subset W^s_{x_2} \text{ and } S(W^s_{x_2}) \subset W^s_{x_1}. \tag{4.2}$$

Analogously, if $y_1, y_2 \in W^u$ symmetrically placed w.r.t. z^* (in the sense of a point-symmetry), then

$$S^{-1}(W^u_{y_1}) \subset W^u_{y_1} \text{ and } S^{-1}(W^u_{y_2}) \subset W^u_{y_2} \tag{4.3}$$

or

$$S^{-1}(W^u_{y_1}) \subset W^u_{y_2} \text{ and } S^{-1}(W^u_{y_2}) \subset W^u_{y_1}. \tag{4.4}$$

Think about W^u and W^s as the one-dimensional spaces spanned by the two eigenvectors with eigenvalues of magnitude larger and smaller than one, respectively. Under the application of S, W^s contracts since the corresponding eigenvalue $|\lambda| < 1$ and recall that W^s covers Ω densely. Hence, one case holds depending on the exact placement of x_1 and x_2 and the precise magnitude and sign of λ. The

[1] Python code: `https://github.com/juliusrueckin/masters-thesis`

symmetric argument holds for W^u and the eigenvalue $|\lambda| > 1$ under application of S^{-1}. S^{-1} is a contraction, since $|\lambda^{-1}| < 1$.

The algorithmic procedure works as follows. Trace the two branches of W^s, namely $W^s_{x_1}$, $W^s_{x_2}$, and the two branches of W^u, namely $W^u_{y_1}$, $W^u_{y_2}$, in parallel. Let each branch of W^s end onto some branch of W^u. Conversely, let each branch of W^u end onto W^s. The resulting (curvylinear) rectagle partition has the property

$$S(\cup_{i=1}^{N} \partial^s R_i) \subset \cup_{i=1}^{N} \partial^s R_i \text{ and } S^{-1}(\cup_{i=1}^{N} \partial^u R_i) \subset \cup_{i=1}^{N} \partial^u R_i, \tag{4.5}$$

where $R = \{R_1, ..., R_N\}$ is the rectangle partition and $\partial^s R_i$ and $\partial^u R_i$ are the stable and unstable boundaries of the rectangle R_i.

By construction, the rectangle boundaries are along W^s and W^u. Equation 4.1 to Equation 4.4 for $W^s_{x_1}, W^s_{x_2}$ and $W^u_{y_1}, W^u_{y_2}$ under the application of S and S^{-1} directly imply Equation 4.5. Note that Equation 4.5 is another formulation of the second half of the intersection property formulated in section 2.3. If $S(R_i) \cap R_j \neq \emptyset$, then $S(R_i) \cap R_j$ cuts completely through R_j in direction of W^u. Further, $S(R_i) \cap R_j$ does not contain $\partial^u R_j$, since the R_i are open sets by definition. However, $S(R_i) \cap R_j$ is not necessarily a single connected strip. If R is not sufficiently fine, there exist $i, j \in [N]$, such that $S(R_i) \cap R_j$ is disconnected, i.e. $S(R_i)$ cuts R_j at least twice. Repeatedly make use of Equation 4.5 by prolonging the two branches of W^s and W^u until they again end onto W^u and W^s respectively. Repeat until R is sufficiently fine. Hence, $S(R_i) \cap R_j$ is guaranteed to be a single connected strip. Thus, the procedure ensures a valid Markov partition R.

On the one hand, the procedure naturally leverages locally split hyperbolic properties examined in chapter 3. On the other hand, the required algorithmic steps are non-trivial to implement. Thus, we extend the work of [24] to a well-defined and automated algorithm.

We consider the dynamic system $(\mathbb{T}^2, A)$ examined in section 4.1. $(\mathbb{T}^2, A)$ is time-discrete, $M = \mathbb{T}^2$ is a differentiable manifold, and the diffeomorphism $f : \mathbb{T}^2 \to \mathbb{T}^2$ is defined by $f(x) = Ax$, where $x \in \mathbb{T}^2$. First, we show how to find a hyperbolic fixed point z^*.

Definition Hyperbolic Fixed Point [38]. Let $f : M \to M$ be a diffeomorphism on M. $z^* \in M$ is a fixed point of the system (M, f), if $f(z^*) = z^*$. Further, z^* is a hyperbolic fixed point, if for all eigenvalues λ of the Jacobian in z^* ($Df(z^*)$), it holds $|\lambda| \neq 1$.

$f(z^*) = z^*$ reduces to solving a linear equation system in the case of $(\mathbb{T}^2, A)$.

Solving for $Az^* = z^*$ leads to a unique fixed point $z^* = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$. For all linear f, $z^* = 0$ is a fixed point. Since f is linear, the Jacobian is independent of z^*. Hence, $Df(z^*) = D(Az^*) = A^T = A$, since A is symmetric. For all eigenvalues of A, it holds $|\lambda| \neq 1$. Thus, z^* is a hyperbolic fixed point. Additionally, the system is locally split hyperbolic in any $x \in \mathbb{T}^2$. Also, $rank(A) = 2$ and $dim(\mathbb{T}^2) = 2$. Thus, the system $(\mathbb{T}^2, A)$ is two-dimensional.

However, in general, $z^* = f(z^*)$ is hard to solve. Hence, we propose an easy numerical schema for solving the fixed point equation $f(z^*) = z^*$. Transform the fixed point problem into a root finding problem:

$$f(z^*) = z^* \Leftrightarrow f(z^*) - z^* = 0$$

or in particular for our linear problem at hand:

$$Az^* = z^* \Leftrightarrow Az^* - z^* = 0.$$

Define $f(z^*) - z^* = 0$ to be the objective function and provide a function $Df(x) = D(Ax - x) = A^T - I = A - I$ that calculates the Jacobian of f in $x \in M$. There are multiple libraries that one could use for automatic differentiation [45]. In our implementation, we utilize the Levenberg-Marquardt method to solve this setup numerically. It is proven to be more robust than the classical Gauss-Newton method, even for bad starting conditions [49]. For the resulting fixed point z^*, check if it is hyperbolic by computing the eigendecomposition of $Df(z^*)$ with the help of standard libraries. If each eigenvalue $|\lambda| \neq 1$, then z^* is hyperbolic. Here, since $D(Az^*) = A$, $z^* = 0$ is indeed hyperbolic.

Next, we develop a method to trace the two branches of W^u and W^s, such that one can compute intersection points between them. Notice the difference in tracing these branches compared to section 4.1. [24] traces branches in parallel while in section 4.1, one sequentially traces stable and unstable branches. Tracing branches in parallel requires a parallel computation of the intersection points. Further, one must decide whether a stable branch ends onto an unstable branch or vise versa. In Figure 4.4a, the branches are traced in parallel. If some branch ends onto another branch or vise versa is induced by analyzing which branch meets the intersection point earlier in time. We introduce notation to make this intuition precise.

Definition - Parameterized Curve [13]. Let $I \subset \mathbb{R}$ be an interval on the real line. A parameterized curve is a smooth function $r : I \to \mathbb{R}^n$. The curve is given by $r(t) = [r_1(t), ..., r_n(t)]$, where $t \in I$ and $r_i : I \to \mathbb{R}$ for all $i \in [n]$. The image $r(I)$ of a curve is called trace. The graph of a function $f : \mathbb{R} \to \mathbb{R}$ is a curve $r : I \to \mathbb{R}^2$ given by $r(t) = [r_1(t), r_2(t)]$, where $r_1(t) = t$ and $r_2(t) = f(t)$ for all $t \in I$. r is

called a regular curve, if $\nabla_t r(t) \neq 0$ for all $t \in I$. If $\|\nabla_t r(t)\| = 1$, then r is called a unit speed curve.

A parameterized curve r evolves in space $\mathbb{R}^n$ over time $t \in I$, where the curve moves with velocity $\|\nabla_t r(t)\|$ at some point in time $t \in I$. Since W^s and W^u are one-dimensional spaces, $W^u_{y_1}$ and W^s_{x1} are also one-dimensional spaces. Hence, both branches $W^u_{y_1}$ and W^s_{x1} are curves $q : I \to \mathbb{R}^2$ and $r : I \to \mathbb{R}^2$. Since both spaces are linear, describe both curves by the origin z^*, which is the hyperbolic fixed point, and the eigenvectors v^u (unstable) and v^s (stable) respectively:

$$q(t) = z^* + t \cdot v^u = z_0^* + t \cdot v^u = [0,0]^T + t \cdot v^u$$

and

$$r(s) = z^* + s \cdot v^s = z_1^* + s \cdot v^s = [1,0]^T + s \cdot v^s,$$

where $t, s \in I$ and where $[0,0]^T$ and $[1,0]^T$ are both identified with the coset of z^*.

Computing intersection points reduces to solve the linear equation system $q(t) = r(s)$ for $t \in I$ and $s \in I$:

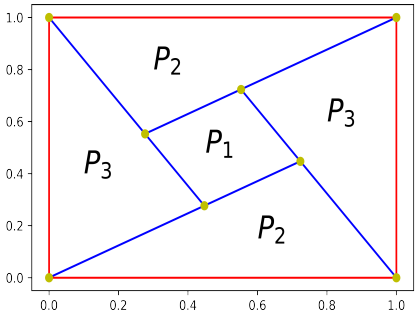
$$\begin{aligned}
&q(t) = r(s) \\
&\Leftrightarrow z_0^* + t \cdot v^u = z_1^* + s \cdot v^s \\
&\Leftrightarrow \langle v^u, z_0^* \rangle + t \cdot \langle v^u, v^u \rangle = \langle v^u, z_1^* \rangle + s \cdot \langle v^u, v^s \rangle, \\
&\langle v^s, z_0^* \rangle + t \cdot \langle v^s, v^u \rangle = \langle v^s, z_1^* \rangle + s \cdot \langle v^s, v^s \rangle \\
&\Leftrightarrow \langle v^u, z_0^* \rangle + t = \langle v^u, z_1^* \rangle, \\
&\langle v^s, z_0^* \rangle = \langle v^s, z_1^* \rangle + s \\
&\Leftrightarrow t = \langle v^u, z_1^* - z_0^* \rangle, \\
&s = \langle v^s, z_0^* - z_1^* \rangle,
\end{aligned}$$

by the scalar product properties. Further, v^u and v^s are orthogonal, so $\langle v^u, v^s \rangle = 0$.

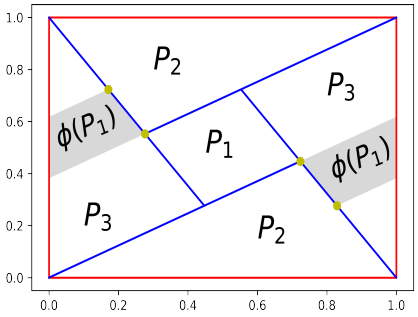
Moreover, $\|\nabla_t q(t)\| = \|v^u\| = 1$ and $\|\nabla_s r(s)\| = \|v^s\| = 1$, since eigenvectors are unit vectors by definition. Hence, both curves are unit speed curves, i.e. they evolve with the same (constant) speed in all $t, s \in I$. Thus,

$$\begin{aligned}
&|t| \geq |s| \Leftrightarrow W^u_{y_1} = q(t) \text{ ends onto } W^s_{x_1} = r(s), \\
&|t| < |s| \Leftrightarrow W^s_{x_1} = r(s) \text{ ends onto } W^u_{y_1} = q(t).
\end{aligned}$$

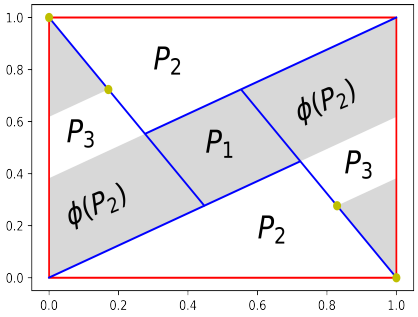
One can now check which branch reaches the intersection point earlier in time. The branch that reaches the intersection point at a later point in time stops in this point. Conversely, the branch that reaches the point earlier in time does not stop.



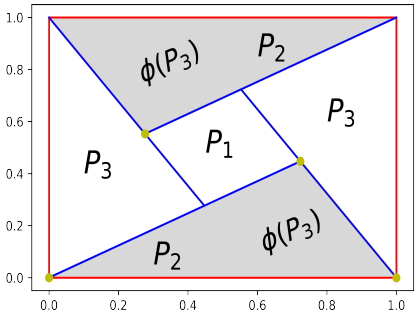
(a) Partition for dynamic system $(\mathbb{T}^2, \phi)$.



(b) System dynamics ϕ applied to the rectangle P_1.



(c) System dynamics ϕ applied to the rectangle P_2.



(d) System dynamics ϕ applied to the rectangle P_3.

Figure 4.4: (a) visualizes the resulting partition when tracing the linear branches until the intersection with another branch. The yellow dots mark the intersection points between these parameterized curves in the unit square's interior. The identification of the hyperbolic fixed point z^* with the unit square's vertices completes the partition's construction. (b)-(d) displays the resulting intersections $\phi(P_i) \cap P_j$. Note that $\phi(P_2) \cap P_3$ does not fulfill the intersection property, and thus the partition at hand is not a Markov partition.

Repeating these computations for all combinations of a stable and unstable branch gives four intersection points. In total, there are eight points in the unit square that define the vertices of all rectangles of the partition. Two rectangles always share such a vertex since the partition is dense in $\mathbb{T}^2$ by construction. The resulting partition is visualized in Figure 4.5. $\phi(P_1)$ and $\phi(P_3)$ both fulfill the intersection property, but $\phi(P_2)$ cuts through P_3 twice. Hence, $\phi(P_2) \cap P_3$ is not a single connected strip. Thus, the partition is not sufficiently fine to be Markov. Repeating the construction process by prolonging all branches until the next intersection with another branch results in a Markov partition visualized in Figure 4.5c.

However, in general, a dynamic system is rarely linear. Hence, the above results do not apply. Thus, in the following, we also introduce a generally applicable algorithm for nonlinear two-dimensional locally split hyperbolic dynamic systems. Suppose z^* to be a hyperbolic fixed point of a nonlinear dynamic system $(\mathbb{T}^2, \phi)$. Since ϕ is nonlinear, W^s and W^u are also nonlinear. Hence, the tracing of branches is challenging. Since W^s and W^u are stable and unstable manifolds, they are invariant manifolds by definition. Therefore, one can approximate W^s and W^u by applying the stable manifold theorem.

Stable manifold theorem [28]. Let Λ be a compact invariant set for a C^r diffeomorphism $f : \mathbb{R}^n \to \mathbb{R}^n$ with a hyperbolic structure $E^s_\Lambda \oplus E^u_\Lambda$. Then there is an $\epsilon > 0$ and there are two collections of C^r manifolds $W^s_\epsilon(x)$, $W^u_\epsilon(x)$, $x \in \Lambda$ which have the following properties:

1. $y \in W^s_\epsilon(x)$ if and only if $d(f^n(x), f^n(y)) \leq \epsilon$ for all $n \geq 0$.
   $y \in W^u_\epsilon(x)$ if and only if $d(f^n(x), f^n(y)) \leq \epsilon$ for all $n \leq 0$.
2. The tangent spaces of $W^s_\epsilon(x)$ and $W^u_\epsilon(x)$ at x are E^s_x and E^u_x respectively.
3. There are constants $C > 0$, $0 < \lambda 1$ such that if $y \in W^s_\epsilon(x)$, then $d(f^n(x), f^n(y)) \leq C\lambda^n$ for $n \geq 0$, and if $y \in W^u_\epsilon(x)$, then $d(f^{-n}(x), f^{-n}(y)) \leq C\lambda^n$ for $n \geq 0$.
4. $W^s_\epsilon(x)$ and $W^u_\epsilon(x)$ are embedded disks in $\mathbb{R}^n$. The mappings from Λ to the function space of C^r embeddings of disks into $\mathbb{R}^n$ given by $x \mapsto W^s_\epsilon(x)$ and $x \mapsto W^u_\epsilon(x)$ are continuous.

Properties one to three of the stable manifold theorem are direct consequences of locally split hyperbolic systems, as seen in chapter 3. The fourth property provides that the tangent spaces and the diameters of $W^s_\epsilon(x)$ and $W^u_\epsilon(x)$ change continuously in x. By hyperbolic structure of Λ, $W^s_\epsilon(x)$ and $W^u_\epsilon(x)$ intersect transversally in exactly one point, namely x. Thus, these invariant manifolds are tangent to the stable and unstable eigenspaces $E^s(z^*)$ and $E^u(z^*)$ of the Jacobian $Df(z^*)$. Further, since tangent spaces of $W^u(x)$ and $W^s(x)$ are one-dimensional, change

continuously in x, and W^u, W^s are invariant spaces, one can approximate them by a first-order integration. Their approximations are obtained by integrating $E^u(x)$ and $E^s(x)$ over time, starting in $x = z^*$ [29]. For literature about how to efficiently trace higher-dimensional invariant manifolds, we refer to [29].

These theoretical justifications induce the following numerical tracing schema for W^u and W^s. Approximate $W^u(z^*)$ and $W^s(z^*)$ by evaluating the flows

$$w^u_{t+1} = w^u_t + \delta \cdot v^u(w^u_t)$$

and

$$w^s_{t+1} = w^s_t + \delta \cdot v^s(w^s_t),$$

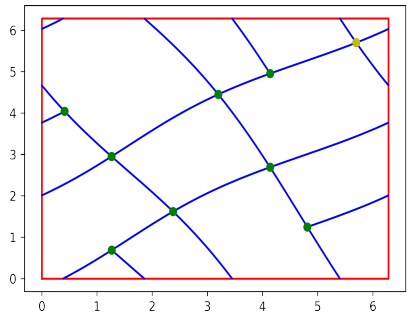
where $v^u(w^u_t)$ and $v^s(w^s_t)$ are the two eigen-subspaces of $D\phi(w^u_t)$ and $D\phi(w^s_t)$. Choose $\delta > 0$ to be a sufficiently small integration constant.

The approximated manifolds w^u_{t+1} and w^s_{t+1} are traced with unit speed, since the eigenvectors of all $D\phi(w^{\cdot}_t)$ are of unit length by definition. Hence, after each integration step, check if the linear piecewise branches of w^u_{t+1} and w^s_{t+1} intersect with a branch of w^s_{t+1} and w^u_{t+1}. A branch w^u_t ends onto a branch of w^s_q, where $t, q \in \mathbb{N}$ are the integration steps, if $t > q$, else w^s_q ends onto w^u_t.

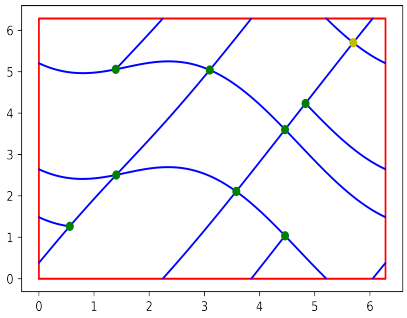
However, tracing these manifolds over the torus $\mathbb{T}^2$ imposes a unique challenge. A manifold is not continuous as required for standard algorithms to detect intersection points but only a piecewise continuous. Continuity breaks at points of identification. Thus, a data structure for these manifolds is required to keep track of these piecewise continuous sections. The system dynamics $\phi(x)$ are essentially defined by a diffeomorphism $f : \mathbb{R}^2 \to \mathbb{R}^2$ acting on the euclidean plane and then performing the identification $\phi(x) = f(x)$ *mod* 1. Therefore, if identification occurs, a new continuous section of the manifold is added. Suppose one traces $W^u(z^*)$ by computing the flow w^u_{t+1}. Then, in each integration step $t + 1$, one checks:

$$\text{identification occurs } \Leftrightarrow w^u_{t+1} \neq w^u_{t+1} \textit{ mod } 1.$$

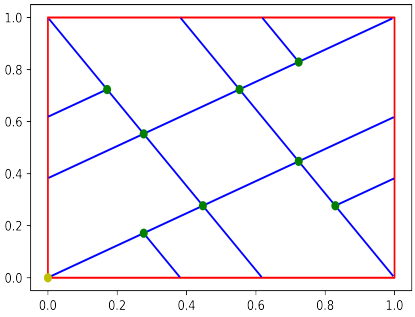
Construct a collection of curves for each branch to trace, here $W^u(z^*)$. Start with $w^u_0(z^*) = z^*$ and integrate w^u_{t+1}. In each integration step t, add the next point w^u_{t+1} on the manifold to its current curve, which is a list of points. If identification occurs, add a new curve to the collection. For such a data structure, we leverage the python library *shapely*. It also implements functionality to compute intersection points between these objects [25]. For the extensive pseudo-code of the overall construction method, see Algorithm 1. Additionally, consider Algorithm 2 for required non-trivial helper functions.



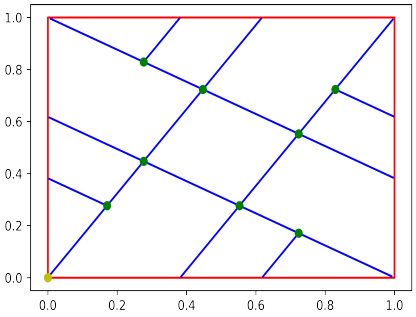
(a) Self-constructed nonlinear dynamic system.



(b) Nonlinear dynamic system from [24].



(c) Linear dynamic system from section 4.1



(d) Linear dynamic system from [24].

Figure 4.5: (a) + (b) are Markov partitions of nonlinear dynamic systems $(\mathbb{T}^2, \phi)$ as defined below. (c) + (d) are Markov partitions for the already seen linear systems in section 4.1 and [24]. All Markov partitions are constructed by executing Algorithm 1. The yellow point marks a hyperbolic fixed point of the respective system. The green points mark the points of intersections between the stable and unstable branches.

Algorithm 1 Construct partition of hyperbolic nonlinear system$(\mathbb{T}^2, \phi_\epsilon)$

procedure CONSTRUCTPARTITION($\phi_\epsilon, D\phi_\epsilon, \delta, m_{id}$, *num_iters*)
  $z^* \leftarrow$ LEVENBERGMARQUARDROOTFINDING($\phi_\epsilon, D\phi_\epsilon, x_0 \in \mathbb{T}^2$)
  if $z^* \neq \phi_\epsilon(z^*)$ **then**
    return z^* is not a fixed point. Something went wrong.
  if *not* LOCALLYHYPERBOLIC(z^*) **then**
    return z^* is not a hyperbolic fixed point. Something went wrong.
  branches $\leftarrow \{W^u_{y_1} : [[z^*]], W^u_{y_2} : [[z^*]], W^s_{x_1} : [[z^*]], W^s_{x_2} : [[z^*]]\}$
  stable_branches $\leftarrow [W^u_{y_1}, W^u_{y_2}]$
  unstable_branches $\leftarrow [W^s_{x_1}, W^s_{x_2}]$
  intersection_points $\leftarrow [\,]$
  for $i \in$ [*num_iters*] **do**
    trace_branches $\leftarrow [W^u_{y_1}, W^u_{y_2}, W^s_{x_1}, W^s_{x_2}]$
    stopped_branches $\leftarrow [\,]$
    while *length(stopped_branches)* $<$ *length(trace_branches)* **do**
      trace_branches $\leftarrow$ *trace_branches* $\setminus$ *stopped_branches*
      for *branch_key* $\in$ *trace_branches* **do**
        last_wt $\leftarrow$ *branches[branch_key][-1][-1]*
        approx_func, rev_dir $\leftarrow$ GETAPPROXFUNCTION(*branch_key*)
        new_wt $\leftarrow$ *approx_func(last_wt,* $D\phi_\epsilon, \delta$*, rev_dir)*
        if *not* IDENTIFICATIONOCCURS(*new_wt*, m_{id}) **then**
          branches[branch_key][-1].append(new_wt)
        else
          new_wt $\leftarrow$ *new_wt mod* m_{id}
          branches[branch_key].append([new_wt])
      for *unstable_branch* $\in$ *unstable_branches* **do**
        for *stable_branch* $\in$ *stable_branches* **do**
          if *unstable_branch*, *stable_branch* $\in$ *stopped_branches* **then**
            continue
          old_ps $\leftarrow \{z^*,$ *intersection_points*$\}$
          $p \leftarrow$ *unstable_branch.intersections(stable_branch)* $\setminus$ *old_ps*
          dist_stable $\leftarrow$ DIST$(p,$ *branches[stable_branch][-1][-1]*$, m_{id})$
          dist_unstable $\leftarrow$ DIST$(p,$ *branches[unstable_branch][-1][-1]*$, m_{id})$
          latter_branch $\leftarrow$ *stable_branch*
          if *dist_unstable* < *dist_stable* **then**
            latter_branch $\leftarrow$ *unstable_branch*
          if *latter_branch* $\notin$ *stopped_branches* **then**
            stopped_branches.append(latter_branch)
            intersection_points.append(p)
  return *branches*, *intersection_points*

Algorithm 2 Helper functions for Algorithm 1

procedure DIST(x, y, m_{id})
    return *min*$\{\|x-y\|_2, \|x-y-m_{id}\cdot\mathbf{1}\|_2, \|x-y+m_{id}\cdot\mathbf{1}\|_2\}$
procedure IDENTIFICATIONOCCURS(x, m_{id})
    return $x \neq x$ *mod* m_{id}
procedure LOCALLYHYPERBOLIC(x, $D\phi_\epsilon$)
    $\Lambda \leftarrow$ EIGENVALUES($D\phi_\epsilon(x)$)
    $n \leftarrow$ *length*(Λ)
    return $\forall i \in [n] : \Lambda_{ii} \neq 1$
procedure APPROXUNSTABLE($x, D\phi_\epsilon, \delta$, *rev_dir*)
    dir_sign $\leftarrow -1$ **if** *rev_dir* **else** 1
    return $x +$ *dir_sign* $\cdot\, \delta \cdot$ COMPUTEUNSTABLEEIGENSPACE($x, D\phi_\epsilon$)
procedure APPROXSTABLE($x, D\phi_\epsilon, \delta$, *rev_dir*)
    dir_sign $\leftarrow -1$ **if** *rev_dir* **else** 1
    return $x +$ *dir_sign* $\cdot\, \delta \cdot$ COMPUTESTABLEEIGENSPACE($x, D\phi_\epsilon$)
procedure COMPUTEUNSTABLEEIGENSPACE($x, D\phi_\epsilon$)
    $\Lambda, V \leftarrow$ EIGENDECOMPOSITION($D\phi_\epsilon(x)$) ▷ Assume $|\Lambda_{11}| > 1 > |\Lambda_{22}|$
    return $V_{1,:}$
procedure COMPUTESTABLEEIGENSPACE($x, D\phi_\epsilon$)
    $\Lambda, V \leftarrow$ EIGENDECOMPOSITION($D\phi_\epsilon(x)$) ▷ Assume $|\Lambda_{11}| > 1 > |\Lambda_{22}|$
    return $V_{2,:}$
procedure GETAPPROXFUNCTION(*branch_key*)
    if *branch_key* $= W^u_{y_1}$ **then**
        return APPROXUNSTABLE, *True*
    if *branch_key* $= W^u_{y_2}$ **then**
        return APPROXUNSTABLE, *False*
    if *branch_key* $= W^s_{x_1}$ **then**
        return APPROXSTABLE, *True*
    if *branch_key* $= W^s_{x_2}$ **then**
        return APPROXSTABLE, *False*

We shortly introduce a nonlinear example of a hyperbolic system to showcase the algorithm. Suppose $(\mathbb{T}^2, \phi_\epsilon)$ to be a nonlinear dynamic system. ϕ_ϵ is a nonlinear perturbation of the example $\phi : \mathbb{T}^2 \to \mathbb{T}^2$ defined by $x \mapsto Ax \ mod\, 1$ from section 4.1. Define $\phi_\epsilon : \mathbb{T}^2 \to \mathbb{T}^2$ as follows:

$$\phi_\epsilon(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}) = \begin{pmatrix} x_1 + x_2 + \epsilon \cdot \cos x_1 \\ x_1 \end{pmatrix} \ mod\, 2\pi.$$

Hence, the Jacobian $D\phi_\epsilon$ of ϕ_ϵ is given by:

$$D\phi_\epsilon(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}) = \begin{pmatrix} 1 - \epsilon \cdot \sin x_1 & 1 \\ 1 & 0 \end{pmatrix}.$$

Note that ϕ_ϵ is area-preserving and invertible for all ϵ and hyperbolic for small ϵ. The resulting Markov partition under application of the described algorithm is visualized in Figure 4.5a.

For algorithm validation, we also reproduce the nonlinear example $(\mathbb{T}^2, S_\epsilon)$ of [24]. $S_\epsilon : \mathbb{T}^2 \to \mathbb{T}^2$ is defined by:

$$S_\epsilon(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}) = \begin{pmatrix} x_1 + x_2 + \epsilon \cdot \cos x_1 \\ x_1 + 2x_2 + 2\epsilon \cdot \cos x_1 \end{pmatrix} \ mod\, 2\pi.$$

Consequently, the Jacobian DS_ϵ of S_ϵ is given by:

$$DS_\epsilon(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}) = \begin{pmatrix} 1 - \epsilon \cdot \sin x_1 & 1 \\ 1 - 2\epsilon \cdot \sin x_1 & 2 \end{pmatrix}.$$

Again, S_ϵ is invertible, area-preserving, and hyperbolic. Compare the partition results from [24] with Figure 4.5b and note that both partitions coincide.

All in all, this completes the development of an automated procedure to construct Markov partitions for two-dimensional nonlinear, locally split hyperbolic dynamic systems. We also publish a modular and easy-to-use python implementation of the proposed algorithm. [2]

[2] Code: `https://github.com/juliusrueckin/masters-thesis`

5 From Markov Partitions to Markov Decision Processes

This chapter investigates missing algorithmic and theoretical work to build Markov decision processes (MDP) for dynamic systems. Markov partitions discretize the state space in a markovian way but do not transport all the information required to define an MDP. One lacks information about the transition probability kernel of the MDP. Thus, we examine different Monte Carlo methods to estimate the transition probability kernel.

Consider the dynamic system and its Markov partition described at the end of section 2.3. The Markov partition $\mathcal{P} = \{P_1, P_2, ..., P_n\}$ for the system (M, ϕ) naturally discretizes the phase space M by aggregating states into Markov rectangles. $\mathcal{P}$ supports the Markov property, since the corresponding shift space $X_{\mathcal{P},\phi}$ is a 1-step shift of finite type. Hence, for a point $x \in X_{\mathcal{P},\phi}$ and a $i \in \mathbb{Z}$, the occurrence of the symbol $x_{i+1} \in x$ merely depends on the symbol x_i. However, the notion of a classical (infinite horizon) MDP requires more elements to be defined.

Definition MDP [5]. A Markov decision process (with infinite horizon) is a tuple $(\mathcal{X}, \mathcal{U}, \mathcal{D}, P, g, \gamma)$ defined as following:

1. $\mathcal{X}$ is a set with $|\mathcal{X}| = n < \infty$ and is called state space.
2. $\mathcal{U}$ is a set with $|\mathcal{U}| < \infty$ and is called action space.
3. $\mathcal{D} \subset \mathcal{X} \times \mathcal{U}$ is the set of admissible state-action tuples, where $\mathcal{U}_k = \{u \in \mathcal{U} : (x_k, u) \in \mathcal{D}\}$ is the set of admissible actions $u \in \mathcal{U}$ at some state $x_k \in \mathcal{X}$. Obviously, $|\mathcal{D}| < \infty$, since $|\mathcal{X}| < \infty$ and $|\mathcal{U}| < \infty$.
4. $P : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \to [0, 1]$ is called a state transition kernel and holds for some fixed pair $(x_k, u_k) \in \mathcal{D}$ a probability measure $\mathbb{P}(x_{k+1}|x_k, u_k) = p(x_{k+1}|x_k, u_k)$, where p is the probability of action $u_k \in \mathcal{U}$ in state $x_k \in \mathcal{X}$ leading to a state transition from x_k to some state $x_{k+1} \in \mathcal{X}$.
5. $g : \mathcal{D} \times \mathcal{X} \to \mathbb{R}$ is the cost function that defines one-stage costs of some admissible state-action pair $(x_k, u_k) \in \mathcal{D}$ and a possible follow-up state x_{k+1}, also called transition tuple (x_k, u_k, x_{k+1}).
6. $\gamma \in (0, 1)$ is the discount factor for future one-stage costs.

We obey the technical details of measure theory in the given definition. Here, assume $\mathcal{X}$ and $\mathcal{U}$ to be finite and naturally endowed with respective σ-algebra. Further, assume p to be a probability measure over the σ-algebra of $\mathcal{X}$. Also, assume g to be a measurable function over the set of transition tuples. There are rarely useful functions g that are not measurable. For further measure-theoretic details, we refer to [5]. Moreover, $\mathcal{D}$, p and g are independent of a discrete time step n. Hence, assume the Markov process to be homogeneous. For a shift space X, the occurrence of a word $w = x_{[i,j]} \in x$ for some point $x \in X$ and indices $i, j \in \mathbb{Z}$ is independent of i, j. It merely depends on the allowed blocks $\mathcal{B}(X)$. Thus, the assumption of homogeneous MDPs fits the developed symbolic dynamics theory.

Define the state space $\mathcal{X} = \mathcal{P} = \{P_1, ..., P_n\}$ by identifying the states $x_i \in \mathcal{X}$ with subsets $P_i \in \mathcal{P}$ for all $i \in [n]$. The order of enumeration for the sets in the partition $\mathcal{P}$ does not affect any further results, but has to be fixed once for state-partition identification. Also, the number of states is determined by the granularity of the Markov partition $\mathcal{P}$.

Next, observe that $\gamma \in (0, 1)$ is a hyper-parameter that is task-specifically tuned. Further, $g : \mathcal{D} \times \mathcal{X} \to \mathbb{R}$ is also task-specific and has to be designed. There is much literature on how to design cost or reward functions in dynamic programming and reinforcement learning [1] [46]. Besides, in general, one should consider the optimization dynamics of the chosen cost function. This discussion is not within the scope of our work. We refer to further literature [32].

Additionally, the action space $\mathcal{U}$ and the admissible state-action tuples $\mathcal{D}$ are task-dependent. Nevertheless, for abstraction purposes, endow $\mathcal{D}$ with a (measurable) mapping $f : \mathcal{X} \to \mathcal{U}$ containing all information about the set of admissible actions $\mathcal{U}_k \subset \mathcal{U}$ an agent could perform in some state $x_k \in \mathcal{X}$. Thus, $(x_k, f(x_k)) \in \mathcal{D}$. Further, let $\mathcal{D}(x_k) = \{u_k \in \mathcal{U} | (x_k, u_k) \in \mathcal{D}\}$ be the set of admissible actions in state $x_k \in \mathcal{X}$.

Last, we formalize the transition probability kernel P. For some $P_i = x_i \in \mathcal{X}$ and $P_k = x_k \in \mathcal{X}$, based on the Markov partition $\mathcal{P}$ and the system dynamics ϕ, it holds:

$$p(x_k | x_i, \mathbf{0}) > 0 \Leftrightarrow \phi(P_i) \cap P_k \neq \emptyset \Leftrightarrow B_{i,k} > 0 \; \forall i, k \in [n],$$

by definition of the Markov partition and its vertex shift B. Equivalently:

$$p(x_k | x_i, \mathbf{0}) = 0 \Leftrightarrow \phi(P_i) \cap P_k = \emptyset \Leftrightarrow B_{i,k} = 0 \; \forall i, k \in [n],$$

where $u_i = \mathbf{0}$ is the 'idle' action. Hence, an agent does not actively interact with the environment but is merely autonomously influenced by the system dynamics ϕ.

This reveals a limitation in the interpretation of Markov partitions. Markov partitions carry the Markov property under system dynamics. Thus, Markov partitions are sophisticated objects to discretize a state space for an MDP. Nevertheless, a Markov partition yields these favorable properties if and only if an agent does not interact with the system itself by any action. To see why this also induces a limitation of applicability, we introduce the policy iteration algorithm and discuss in which case Markov partitions still yield strong mathematical guarantees.

The policy iteration algorithm works as follows [7]:

1. Choose (randomly) an initial policy π_0. Let $k = 1$.
2. Compute V_{π_k} by $T_{\pi_k} V_{\pi_k} = V_{\pi_k}$.
3. Compute π_{k+1} by $T_{\pi_{k+1}} V_{\pi_k} = T_g V_{\pi_k}$.
4. If $\pi_{k+1} = \pi_k$, then stop. Else set $k = k + 1$ and go to Step 2.

$V_\pi : \mathcal{X} \to \mathbb{R}$ is the value function of a currently followed policy π by the agent:

$$V_\pi(x) = \mathbb{E}_{p_\pi(x'|x)}[g(x, \pi(x), x') + \gamma V(x')].$$

The policy $\pi : \mathcal{X} \to \mathcal{U}$ specifies the action $u \in \mathcal{U}$ an agent should perform in a state $x \in \mathcal{X}$. $T_\pi : \mathbb{R}^n \to \mathbb{R}^n$ is the bellman operator for a fixed policy π:

$$T_\pi V(x) = \mathbb{E}_{p_\pi(x'|x)}[g(x, \pi(x), x') + \gamma V(x')].$$

$T_g : \mathbb{R}^n \to \mathbb{R}^n$ is the optimal (or greedy) Bellman operator computing for each state $x \in \mathcal{X}$ the minimal value by greedily choosing the locally optimal action $u \in D(x)$ with minimal value:

$$T_g(V)(x) = min_u \, \mathbb{E}_{p(x'|x,u)}[g(x, u, x') + \gamma V(x')].$$

Hence, based on some value function V_π, a policy $\pi_g(V_\pi)$ greedily induced by T_g fulfills $\pi_g(V_\pi)(x) \in argmin_u \, \mathbb{E}_{p(x'|x,u)}[g(x, u, x') + \gamma V_\pi(x')]$. Thus, π is the optimal policy an agent could follow within the environment, if and only if $T_\pi V^* = T_g V^*$, where V^* is the optimal value function. This also motivates the iterative approach of policy evaluation (step 2) and policy improvement (step 3), since $V_{\pi_{k+1}} \leq V_{\pi_k}$. Hence the convergence to an optimal policy is guaranteed by the contraction and monotonicity properties of the bellman operators T_π and T_g. For a detailed explanation and proofs, see [7].

By definition of the policy evaluation step (step 2), one computes a value function under some fixed policy π_k. Hence, the agent does interact with the environment (M, ϕ) via this fixed policy. Further, the policy affects the transition probability

kernel P and V_{π_k}, since expectations over p_{π_k} are evaluated. If a policy always performs a **0**-action, i.e. $\pi_k(x) = \mathbf{0}$ for all $x \in \mathcal{X}$, then the Markov partition encodes information about $p(x'|x, \mathbf{0})$.

However, the policy improvement step (step 3) greedily chooses a new policy $\pi_{k+1}(x)$ taking the action with expected minimal value $V_{\pi_k}(x)$ for all $x \in \mathcal{X}$. Hence, in the next policy iteration step $k+1$, $p_{\pi_{k+1}}$ changes. Thus, the partition $\mathcal{P}$ is not Markov anymore. Now, not only the system dynamics ϕ influence the overall system response but also the policy π_{k+1}. Consider a linear time-invariant system with system dynamics represented by a matrix A, i.e. $\phi(x) = Ax$. Then, $\psi(x) = Ax + Bu$ characterizes the overall system response, where u are the system inputs.

There could be different approaches to tackle the mismatch. First, one could argue that within each policy iteration, one constructs a new Markov partition of the phase space considering both the policy π and the system dynamics ϕ. However, there are drawbacks and open questions. On the one hand, computing new Markov partitions could blow up the policy iteration algorithm's computational costs. On the other hand, changing the state space representation poses questions regarding the policy iteration algorithm's convergence guarantees. Moreover, locally split hyperbolicity over the phase space must be guaranteed under all chosen policies π_k to construct Markov partitions. Since hyperbolic systems are structurally stable, adding small perturbations to a policy again results in overall hyperbolic system responses. So there is evidence to at least hope for a solution that suitably constrains the policy space to ensure hyperbolicity. However, it is questionable if such restrictions lead to an optimal policy.

In this work, we reduce Markov partitions' application and experimental analysis in dynamic programming to a specific policy evaluation step. Hence, the agent will follow a fixed policy π_k. Further, Markov partitions are created for the system response of a particular policy evaluation step. If an agent follows the **0**-policy, one can directly proceed with the Markov partition construction and transition kernel estimation. However, restricting to such a trivial policy is not feasible for evaluating Markov partitions applied to a policy evaluation step. Thus, from now on, we choose such systems ϕ and policies π_k that result in system responses still fulfilling the requirements to build a Markov partition. Suppose ϕ to capture the system responses from now on and directly apply the above results as usual.

At first glance, a restriction to a single policy evaluation step seems trivial. However, the policy evaluation step itself is often solved iteratively instead of computing a closed-form solution. Finding such a closed-form solution is often numerically unstable and expensive to compute. Hence, one utilizes an iterative policy evaluation schema [7]:

1. Let $V_0 \in \mathbb{R}^n$ be arbitrarily chosen. Let k = 0.
2. Compute $V_{k+1} = T_\pi V_k$
3. Stop if $\|V_{k+1} - V_k\|_\infty < \epsilon$. Else set $k = k + 1$ and go to step 2.

Note that $\epsilon > 0$ should be a sufficiently small constant. There are non-trivial questions regarding the application and advantages of Markov partitions. E.g., do Markov partitions guarantee a faster convergence to the value function V_π compared to comparably fine-grained regular grid-like partitions? Do these convergence differences dependent on the number of states, i.e., on the partition's granularity? How could one utilize superior convergence to influence the overall performance of a policy iteration algorithm? These open questions must be examined specifically for the policy evaluation step. Consequently, reducing our work to a fixed policy evaluation step still raises complex questions we examine experimentally in chapter 6.

Next, we estimate the transition probabilities under a policy π from x_i to x_k for all $i, k \in [n]$, where $B_{i,k} \neq 0$. For these state transitions, one cannot apriori infer the state transition probability just by $\mathcal{P}$. Moreover, we introduce the paradigm shift that ϕ characterizes the overall system response for simplicity.

Note that if $\phi(P_i) \cap P_k \neq \emptyset$ for $P_k = x_k \in \mathcal{X}$, then $p(x_k|x_i, u_{i\to k}) > 0$. Further, there could be many such $k \in [n]$, since the image $\phi(P_i)$ could cut through multiple P_k. For a fixed k, $\phi(P_i) \cap P_k$ contains at least one element $p_k \in P_k$ but it could also hold $\phi(P_i) \cap P_k = P_k$. Consider Figure 2.2 for $P_i = P_2$. Observe that $\phi(P_2)$ cuts through $P_k = P_1$ with $\phi(P_2) \cap P_1 = P_1$ and through $P_k = P_3$ with $\phi(P_2) \cap P_3 \subset P_3$. Hence, $p(x_2|x_2, f(x_2)) = B_{2,2} = 0$, $p(x_1|x_2, f(x_2)) > 0$ and $p(x_3|x_2, f(x_2)) > 0$. It feels natural to determine the probabilities $p(x_3|x_2, f(x_2))$ and $p(x_1|x_2, f(x_2))$ by analyzing the area of overlap $\phi(P_i) \cap P_k$ relative to the overall area of the image $\phi(P_i)$. This intuition reflects the possibility of an agent moving from P_i to some P_k given the system response induced by ϕ. Roughly estimate the length of the expanding boundary of P_1 as double the length of the expanding boundary of P_3. Hence, $p(x_1|x_2, f(x_2)) \approx \frac{2}{3}$ and $p(x_3|x_2, f(x_2)) \approx \frac{1}{3}$.

We make this intuition precise. For simplicity, assume the phase space M to be discrete and $|M| < \infty$. Therefore, for all $P_i \in \mathcal{P}$ it holds that P_i is discrete and $|P_i| < \infty$. Hence, the probability measure p reduces to a counting problem. Let $P_i, P_k \in \mathcal{P}$. Suppose $\phi(P_i) \cap P_k \neq \emptyset$. Then, it holds:

$$p(x_k|x_i, f(x_i)) = \frac{|\phi(P_i) \cap P_k|}{|\phi(P_i)|}.$$

We prove that p is indeed a probability density function. Therefore, one shows that $p(x_k|x_i, f(x_i)) \in [0,1]$ for any x_i, x_k and for a fixed x_i, $\int_{\mathcal{X}} p(x_k \mid x_i) dx_k = 1$.

Proof. For arbitrary $x_i, x_k \in \mathcal{X}$, it holds that $0 \leq p(x_k|x_i, f(x_i)) \leq 1$, since $0 \leq |\phi(P_i) \cap P_k| \leq |\phi(P_i)|$ by definition of set intersection.

For an arbitrarily fixed $x_i \in \mathcal{X}$, it holds:

$$\sum_{k=1}^{n} p(x_k|x_i, f(x_i)) = \sum_{k=1}^{n} \frac{|\phi(P_i) \cap P_k|}{|\phi(P_i)|} = \frac{|\phi(P_i) \cap (\cup_{k=1}^{n} P_k)|}{|\phi(P_i)|}$$
$$= \frac{|\phi(P_i) \cap M|}{|\phi(P_i)|} = \frac{|\phi(P_i)|}{|\phi(P_i)|} = 1,$$

where one used the definition of p and set operations. Observe that $M = \cup_{k=1}^{n} P_k$ by definition of a topological partition. Thus, p is a probability measure. □

Analogous, define p for arbitrary $\mathcal{C}^1$-smooth manifolds M. Let $x_i, x_k \in \mathcal{X}$ be arbitrary. The indicator function of a set $A \subseteq M$ and a point $p \in M$ is defined as:

$$\mathbf{1}_A(p) = \begin{cases} 1 & \text{if } p \in A \\ 0 & \text{if } p \notin A. \end{cases}$$

Then, the probability measure p is defined as:

$$p(x_k|x_i, f(x_i)) = \frac{\int_M \mathbf{1}_{\phi(P_i) \cap P_k}(p) dp}{\int_M \mathbf{1}_{\phi(P_i)}(p) dp},$$

where $p \in M$ are all points in M to integrate over. The proof that this definition of p is a probability density function is analogous to the proof above. Plug in for the integral definition of p and verify that the properties of a probability density function again hold. For some geometrically simple intersections, one can solve surface or volume integrals.

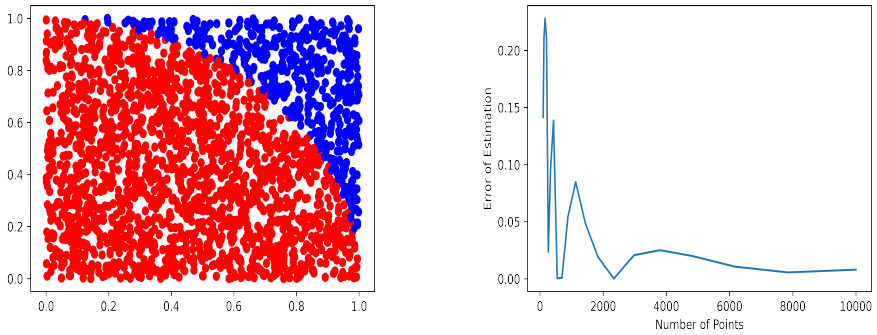
However, in general, calculating these integrals over arbitrary geometry is challenging. Thus, we develop generally applicable Monte Carlo algorithms to estimate the state transition probability matrix P. Consider $\phi : M \to M$ and the Markov partition $\mathcal{P}$ of M to be given. Create (multiple) Markov chains of state transitions, i.e., random walks over the graph of possible state transitions defined by the adjacency matrix B. Since B does not encode any precise information about probabilities $p(x_k|x_i, \pi(x_i))$, guide the random walks by the system responses ϕ to approximate the probability distributions in question.

Start by choosing an $i \in [n]$ uniformly at random. Then, sample a particle $p \in P_i$ uniformly at random. Set $k \in [n]$ such that $\phi(p) \in P_k$. Note that in general, there is exactly one such k. Count a state transition from x_i to x_k and proceed with $i := k$. Repeat for the fixed length of a random walk. Then, restart a new random walk and repeat until a fixed number of random walks is performed. Last, normalize the count matrix of occurred state transitions row-wise to estimate the probability matrix P. For a formalization in pseudo-code, see Algorithm 3.

Algorithm 3 Estimate probability matrix P by random walks of particles in (M, ϕ)

```
1:  procedure EstimateProbabilityMatrix(φ, 𝒫, l, m)
2:      n ← |𝒫|
3:      P ← 0 ∈ [0,1]^{n×n}                          ▷ initialize probability matrix
4:      C ← RandomWalks(φ, 𝒫, l, m)                  ▷ C ∈ ℕ^{n×n}
5:      for i ∈ [n] do                               ▷ loop over all states x_i
6:          P_{i,:} ← 1/‖C_{i,:}‖_1 · C_{i,:}        ▷ normalize count matrix row-wise
7:      return P
8:  procedure RandomWalks(φ, 𝒫, l, m)                ▷ m is number and l length of RW
9:      n ← |𝒫|
10:     C ← 0 ∈ ℕ^{n×n}                              ▷ initialize count matrix
11:     numwalks ← 0
12:     while numwalks < m do
13:         numsteps ← 0
14:         i ← sample uniformly at random from [n]  ▷ choose state RW starts in
15:         while numsteps < l do
16:             p ← sample uniformly at random from P_i   ▷ P_i ∈ 𝒫, P_i = x_i
17:             k ← choose k ∈ [n], such that φ(p) ∈ P_k   ▷ ∃!k ∈ [n]
18:             C_ik ← C_ik + 1
19:             i ← k                                ▷ walk to next state
20:             numsteps ← numsteps + 1
21:         numwalks ← numwalks + 1
22:     return C
```

Assume a random walk to be sufficiently long or the number of random walks to be sufficiently high, i.e., l or m large. Hence, each state $i \in [n]$ is visited at least once. Further, from a state i, suppose to visit at least one successor state k. This implies that $\|C_{i,:}\|_1 > 0$ in line 6. Second, $int(P_i)$ is dense in P_i for all $i \in [n]$. Thus, one almost never samples a $p \in P_i$ in line 16 with more than exactly one k such that $\phi(p) \in P_k$. Even if there are two or more such k, one could choose a k uniformly at random without losing correctness of the estimate $p(\cdot|x_i, \pi(x_i))$.



(a) Points sampled uniformly at random from the unit square. Red points are in the unit disk.

(b) Convergence of π-estimate to real value of π at around $2 \cdot 10^4$ points.

Figure 5.1: Classic example for a Monte Carlo method to estimate the number π. The same geometrical intuition is leveraged to estimate the transition probability matrix P.

In general, one could set $m = 1$ to avoid one nested while loop. Then, one single Markov chain or random walk with length $l \to \infty$ is produced. Further, the Markov property $\mathcal{P}$ guarantees that reaching a successor state merely depends on the current state. Hence, the estimate for $P \in [0,1]^{n \times n}$ is independent of the state in which the random walker started. Nevertheless, in practical applications, it is beneficial to restart a random walk from different starting states i. Suppose the system responses ϕ allow only for a slow exploration of the state space, i.e., many random walk steps in approximately the same region are performed before transitions in other areas of the state space occur. Further, if these transitions appear rarely, it is more efficient to exploit a uniform random sampling of start states. Thus, it prevents a biased estimate of the probability distribution by choosing an uncomfortable start state once. In particular, it facilitates the exploration of highly unlikely state space regions. The sensitivity of an estimate regarding choices m and l is experimentally examined in chapter 6.

The random walk algorithm is morally motivated by simulating particles in a dynamic system guided by the system responses ϕ. However, there is another classical Monte Carlo method to estimate P. Consider estimating the number π by sampling $t \to \infty$ points uniformly at random from an unit square as displayed in Figure 5.1. Evaluating the number of points $a \in [0,1]^2$ fulfill $\|a\|_2 \leq 1$. Let $|\{a : \|a\|_2\}| = t'$. Then, π is estimated by:

$$4 \cdot p(\|a\|_2 \leq 1) = 4 \cdot \frac{t'}{t} = \pi.$$

We adapt this intuition to estimate areas of overlap $\phi(P_i) \cap P_k$ for all $i, k \in [n]$. Fix some state $i \in [n]$. Sample $r \in \mathbb{N}_+$ points $p \in P_i$ uniformly at random. For

Algorithm 4 Estimate probability matrix P by Monte Carlo π method

procedure ESTIMATEPROBABILITYMATRIX($\phi, \mathcal{P}, r$) ▷ $r \in \mathbb{N}_+$
  $n \leftarrow |\mathcal{P}|$
  $P \leftarrow 0 \in [0,1]^{n \times n}$
  $C \leftarrow 0 \in \mathbb{N}^{n \times n}$
  for $i \in [n]$ **do** ▷ loop over all states x_i
    samples $\leftarrow 0$
    while *samples* $< r$ **do** ▷ sample r successor states for state i
      $p \leftarrow$ *choose uniformly at random from* P_i
      $k \leftarrow$ *choose* $k \in [n]$, *such that* $\phi(p) \in P_k$ ▷ $\exists! k \in [n]$
      $C_{ik} = C_{ik} + 1$
      samples $\leftarrow$ *samples* $+ 1$
    $P_{i,:} = \frac{1}{\|C_{i,:}\|_1} \cdot C_{i,:}$
  return P

Algorithm 5 Estimate probability matrix P by Monte Carlo π method

procedure ESTIMATEPROBABILITYMATRIX($\phi, \mathcal{P}, c, \tau$) ▷ $c \in \mathbb{N}_+$, $\tau \in (0,1)$
  $n \leftarrow |\mathcal{P}|$
  $P \leftarrow 0 \in [0,1]^{n \times n}$
  $P^{(old)} \leftarrow 1 \in [0,1]^{n \times n}$ ▷ used to store estimate of c steps ago
  $C \leftarrow 0 \in \mathbb{N}^{n \times n}$
  for $i \in [n]$ **do** ▷ loop over all states x_i
    samples $\leftarrow 0$
    while $\left\| P_{i,:} - P_{i,:}^{(old)} \right\|_\infty \geq \tau$ **do** ▷ estimate for state i not converged
      $p \leftarrow$ *choose uniformly at random from* P_i
      $k \leftarrow$ *choose* $k \in [n]$, *such that* $\phi(p) \in P_k$ ▷ $\exists! k \in [n]$
      $C_{ik} = C_{ik} + 1$
      samples $\leftarrow$ *samples* $+ 1$
      if *samples* mod $c = 0$ **then**
        $P_{i,:}^{(old)} \leftarrow P_{i,:}$
        $P_{i,:} \leftarrow \frac{1}{\|C_{i,:}\|_1} \cdot C_{i,:}$
  return P

each sample p, choose $k \in [n]$, such that $\phi(p) \in P_k$. Count a state transition from i to k. Last, normalize these counts and yield the transition probability estimate $p(x_k|x_i, \pi(x_i))$ for all $k \in [n]$. Repeat this procedure for all states $i \in [n]$. For precise pseudo-code, see Algorithm 4.

Observe that a proper choice for $r \in \mathbb{N}_+$ is highly dependent on the state i. For example, consider P_1 in Figure 2.2. Suppose $\phi(P_1)$ to behave such that the area of overlap between $\phi(P_1)$ and P_2 is very large and the area of overlap between $\phi(P_1)$ and P_3 is almost 0. Then, r has to be sufficiently large, since the probability to sample a $p \in P_1$ with $\phi(p) \in P_3$ is almost 0. Thus, a precise probability estimate requires significantly more samples p. Therefore, instead of investing much time in tuning an appropriate $r \in \mathbb{N}_+$, introduce a convergence criterion. Each $c \in \mathbb{N}_+$ steps, one updates the i-th row of probability matrix P by the collected count data in $C_{i,:}$. Further, introduce a convergence threshold $\tau \in (0, 1)$. If some old estimate (from c steps ago) of $P_{i,:}$ differs by less than τ in its maximal deviating state $j \in [n]$, the estimates $P_{i,:}$ are considered to be converged. These improvements are formalized in Algorithm 5. The overhead of finding a suitable r is eliminated and an adaptive convergence criterion for each state $i \in [n]$ is implemented.

This completes the construction of an MDP. We discussed the integration of Markov partitions into the DP-framework. Further, we showed resulting limitations and suggested solutions to bypass them. Last, we defined a Markov partition based state transition probability measure. Additionally, we developed two Monte Carlo algorithms to estimate this probability measure for arbitrary geometries. [1]

[1] Code for algorithms: `https://github.com/juliusrueckin/masters-thesis`.

6 Experiments and Applications

This chapter experimentally investigates the performance of our Markov partition based MDP framework developed in chapter 4 and chapter 5 applied to DP-algorithms. Therefore, we construct MDPs $(\mathcal{X}, \mathcal{U}, P, g, \gamma)$ for agents acting in a dynamic system environment following a fixed policy π. Within this setting, the convergence behavior of Monte Carlo based policy evaluation algorithms is analyzed. First, the parameter sensitivity of the two Monte Carlo algorithms estimating $p(x_k|x_i, \pi(x_i))$ introduced in chapter 5 is examined. Second, the MDP's convergence properties are compared to the regular grid-like discretizations. In general, the experiments answer the questions raised in chapter 5 regarding the integration of Markov partitions into policy evaluation algorithms.

6.1 Experiments - Monte Carlo Algorithms

Algorithm 3 and Algorithm 5 are proposed to estimate the state transition probability kernel of an MDP. We evaluate the probability estimation's quality dependent on the choice of hyper-parameters. We refer to Algorithm 3 as the random walk method. Symmetrically, we refer to Algorithm 5 as the π-sampling method.

Implementations. A python implementation for both algorithms is published[1]. For a detailed description of the algorithms, we refer to chapter 5. The published implementations of the algorithms work exactly like the pseudo-codes state. However, they were extended by parallelization of computations to improve the run time drastically. The random walk algorithm is parallelized by equally splitting the to be executed number of random walks m over all available CPU cores. The π-sampling method is parallelized by equally distributing the number of subsets n in the Markov partition $\mathcal{P}$ over all CPU cores.

Hardware. All experiments are executed on standard hardware. The machine has 8GB memory and an Intel Core i5 7th generation quad-core CPU with 3,8 GHz. The experiments run within a docker container to facilitate the installation and execution. The docker container has access to all hardware resources.

[1] Code: `https://github.com/juliusrueckin/masters-thesis`

Parameter choices. For the random walk method, 20 different choices for the number of random walks m equally distributed over the interval $[1, 300]$ in combination (cross-product) with 20 different choices for the length of random walks l equally distributed over the interval $[1, 100]$ are taken. For the π-sampling method, 20 different choices for the number of sample points per subset c equally distributed over an interval $[1, 300]$ in combination with ten different choices for the convergence threshold τ logarithmically distributed over an interval $[10^{-2}, 10^{-6}]$ are considered.

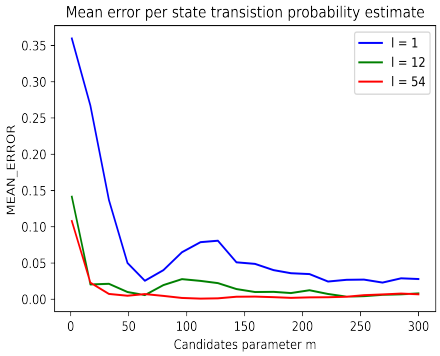
Baseline. We choose the typical example of a linear dynamic system over the torus described in section 2.3. See section 4.1 for a deeper analysis of its Markov partitions. In the experiments, the Markov partition $\mathcal{P}$ visualized in Figure 4.3 is used. For this rather simple system, one can easily compute the exact overlapping area of some $\phi(P_i)$ and P_j required to infer the ground truth probabilities as defined in chapter 5. This ground truth is utilized to quantify the probability estimation error of both Monte Carlo algorithms.

Evaluation strategy. We experiment with all stated combinations of hyper-parameters. On the one hand, the run time of each run with a fixed hyper-parameter combination is measured as a proxy for convergence speed. On the other hand, the estimation error is evaluated by computing the difference between the estimated probability matrix $P^{est} \in [0, 1]^{n \times n}$ and the ground truth probability matrix $P \in [0, 1]^{n \times n}$ given by our baseline. One quantifies this difference by computing the following evaluation criteria for each run:

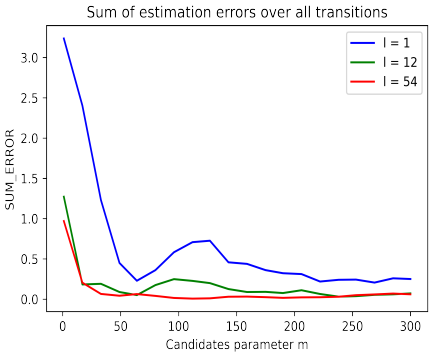
- $MEAN\_ERROR = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} |P_{i,j} - P_{i,j}^{est}|$,
- $SUM\_ERROR = \sum_{i=1}^{n} \sum_{j=1}^{n} |P_{i,j} - P_{i,j}^{est}|$,
- $MAX\_ERROR = max_{i,j \in [n]} |P_{i,j} - P_{i,j}^{est}|$,
- $MEDIAN\_ERROR = median(abs(P - P^{est}))$, where $abs(\cdot)$ is the absolute value applied to each element of the matrix $P - P^{est}$.

Further, each run with a fixed hyper-parameter combination is repeated ten times. One calculates the average of each evaluation criterion and the average run time to account for naturally occurring probabilistic variations. Also, the variances are calculated but not stated, since they are negligible small in the experiments.

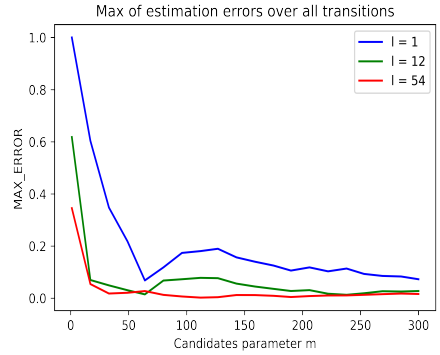
Results - Random walk method. Empirical results for the random walk method can be seen in Figure 6.1. All evaluation criteria displayed in Figure 6.1a to Figure 6.1c show that for around $m = 75$ executed random walks the estimation error shrinks to a suitable magnitude for large $l = 54$. For $m > 250$ and $l \geq 12$, the estimate almost converged to the ground truth. Hence, the method can precisely



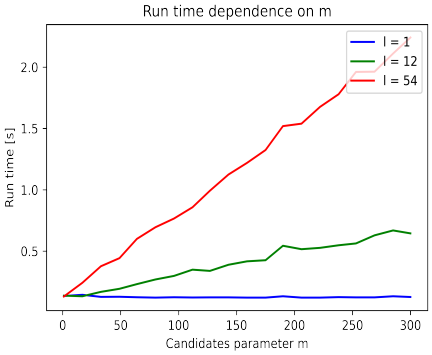
(a) Mean error of random walk estimate



(b) Sum error of random walk estimate



(c) Maximal error of random walk estimate



(d) Run time of random walk

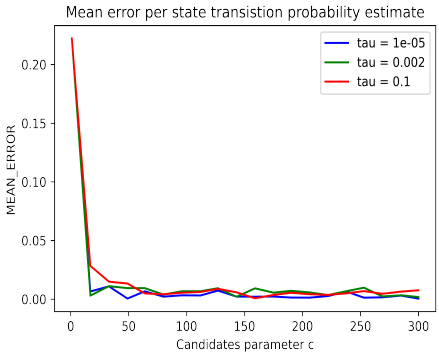
Figure 6.1: (a), (b) and (c) show the mean, sum and maximal estimation error of the random walk method respectively, depending on the number of executed walks m. A long (red), medium (green) and short (blue) random walk of length l is chosen for reference. (d) shows the the run time depending on the number of executed walks m.

estimate the ground truth by the law of large numbers. Only for extremely short random walks, the method fails to estimate the probabilities within an acceptable number of random walks m. Thus, a trade-off between the length and number of random walks is most efficient to estimate transition probabilities in practice. Further, the maximal estimation error vanishes for $m > 250$ and $l \geq 12$. This is especially desirable to infer an optimal policy over the whole state space. Last, the run time linearly depends on the number of random walks m. By parallelization over m, one reduces this increase by a constant factor. However, the run time is still linear in m. Also, the run time is linear in l. Nevertheless, the method converges to reasonable estimates in a moderate run time. Therefore, it is comparably cheap to increase m and l for more complex state space partitions.

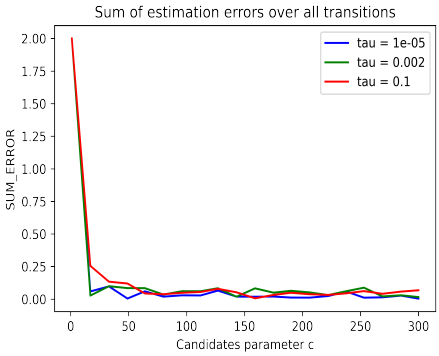
Results - π-sampling method. Empirical results for the π-sampling method can be seen in Figure 6.2. All evaluation criteria displayed in Figure 6.2a to Figure 6.2c show that for around $c = 100$ the estimation error shrinks to a suitable magnitude. For $c > 250$, the estimate almost converged to the ground truth. Hence, the method can indeed converge to the ground truth. Only for large τ, the estimate does not converge since a large τ is a poor approximation for convergence. Next, the mean error and the maximal estimation error shrinks to zero for sufficiently small $\tau \leq 10^{-4}$. Thus, sufficiently precise probability estimates for all states are guaranteed. The median estimation error results are not shown since they behave almost identically to the mean estimation error. Last, the run time is very low. Therefore, the convergence speed is especially suitable for more complex state space partitions. Further, the run time is almost constant over c since the computation is parallelized over all $n = 3$ subsets of the Markov partition. n is not larger than the number of CPU cores over which the computations are parallelized. However, from a theoretical perspective, one experiences a linear run time increase for more fine-grained partitions. Additionally, for too small convergence thresholds $\tau \leq 10^{-5}$, one enters a highly unstable convergence regime. Sometimes, significantly more probability updates after sampling the next c points must be performed to convergence. Thus, we propose setting τ to around 10^{-4}. Further, introducing sufficiently large $c > 250$ more fine-grained partitions is computationally cheap.

Comparison of algorithms. Both algorithms can converge to the ground truth probability within a suitable range of parameters. Moreover, in the case of $m \to \infty$, $l = 1$ and $c \to \infty$, both algorithms are essentially identical. However, in practice, a mixture between the number of random walks $m \geq 150$ and their length $l \geq 12$ yields the best results. For the π-sampling method, a fixed threshold $\tau = 10^{-4}$ and a moderate $c \geq 250$ guarantee reasonable estimates and a short run time.

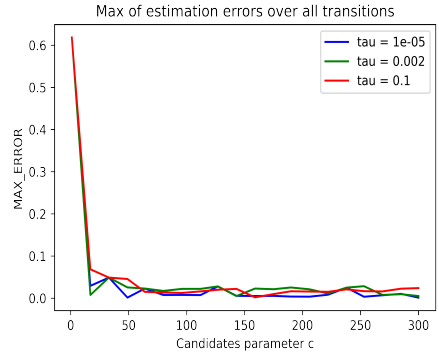
In general, the run time difference between the random walk and π-sampling method is significant. At least for state space partitions with a small number of sub-



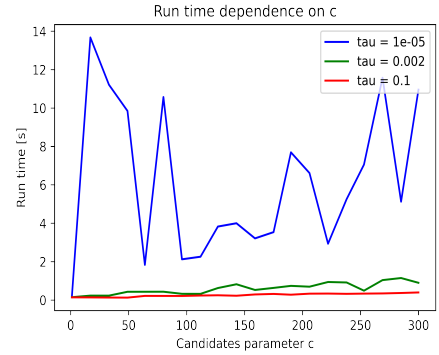
(a) Mean error of π-method estimate



(b) Sum error of π-method estimate



(c) Maximal error of π-method estimate



(d) Run time of π-method

Figure 6.2: (a), (b) and (c) show the mean, sum and maximal estimation error of the π-sampling method respectively, depending on the number of sampled points c. A large (red), medium (green) and small (blue) convergence threshold τ is chosen for reference. (d) shows the the run time depending on the number of sampled points c.

sets $|\mathcal{P}| = n$, the π-sampling method shows significant run time advantages over the random walk method. In practice, even for large c, the run time is still surprisingly low. Of course, these differences are induced by the parallelization possibilities each algorithm provides. Parallelizing the π-sampling method over its number of states n is more efficient than parallelizing over potentially many m random walks of length $l \ll c$. Therefore, we consider the π-sampling method to be less hyper-parameter sensitive and the overall more efficient estimation method.

6.2 Experiments - Dynamic Programming Algorithms

As argued in chapter 5, the experiments are restricted to policy evaluation algorithms. In particular, they are focused on iterative Monte Carlo based policy evaluation algorithms. We investigate if Markov partitions imply superior convergence behavior compared to regular grid-like partitions. The results under multiple dynamic systems, partitions and policies are the basis for discussing whether the properties of Markov partitions are indeed transferable to ADP.

Implementation. The parallelized python implementation of the π-sampling method is used to estimate the state transition probability kernel. For constructing the Markov partitions of the system response ϕ, the implementation proposed in chapter 4 is utilized. Last, we also publish an implementation of the iterative policy evaluation algorithm. The implementation is a direct translation of the iterative policy evaluation schema introduced in chapter 5.[2]

Hardware. All experiments are executed on the same hardware and docker setup as described in section 6.1. No asynchronous version of policy evaluation is leveraged. Hence, only one CPU-core is utilized at a time while executing the iterative policy evaluation. For reference to asynchronous approaches, see [10].

Baseline. The baselines to partition the continuous phase space are regular grid-like discretizations. The grid boundaries are chosen so that each subset covers approximately the same area of the phase space. We construct multiple baseline partitions with equal spacing along one or both dimensions.

Parameter settings. For the π-sampling method, $\tau = 10^{-4}$ is chosen. The number c of sampled points per state before executing a probability update is chosen to be $c = 3000$, since experiments in section 6.1 show that increasing c does not raise computational costs significantly. At the same time, c is sufficiently large to estimate the probability kernel, even for fine-grained partitions.

[2] Code: `https://github.com/juliusrueckin/masters-thesis`

Evaluation strategy. Two-dimensional locally split hyperbolic dynamic systems are examined since Markov partitions can be constructed for these systems. The agent follows a fixed policy π for which a policy evaluation is computed iteratively. The system dynamics and the followed policies π are varied. One analyzes the policy evaluation under partitions of different granularity. For each experiment, multiple regular grid-like partitions are constructed. The convergence of the π-sampling method and the policy evaluation are investigated.

The agent's goal is to reach a target state or point in the phase space. The setting is inspired by the finite, non-continuous 2D grid example, where an agent has to reach a target state in as few as possible steps. We adopt this scenario to a system control motivated equivalent. The agent should reach a target point and stay within a neighborhood around the target point. Such a target point could be a fixed point of a system as in most system control settings. A famous system control example is the pendulum-v0 environment of openai gym, where one has to control the system by torque inputs to swing up the pendulum such that it stays upright[3]. For an elaborate discussion of possible real-world applications, see section 6.3.

To be precise, the goal is characterized by a simple cost function. Let (M, ϕ) be the environment over the phase space M with system responses $\phi : M \to M$. Let $y \in M$ be the target point and $\mathcal{X} = \mathcal{P} = \{P_1, ..., P_i, ...P_n\}$ be the discretized state space $\mathcal{X}$ by a partition $\mathcal{P}$ of the continuous phase space M. The cost function $g : \mathcal{X} \to \mathbb{R}_{\geq 0}$ is defined as follows:

$$g(P_i) = dist(P_i, y),$$

where $P_i \in \mathcal{P}$ and $dist : \mathcal{X} \times M \to \mathbb{R}_{\geq 0}$ is the shortest distance between the center of subset P_i and y. In the experiments, multiple $y \in M$ are chosen that could also coincide with fixed points of (M, ϕ).

Last, we introduce measures to evaluate the convergence behavior of the π-sampling method and the policy evaluation. Therefore, count the number of estimation updates $p(\cdot|P_i)$ until convergence for each state P_i. For details see chapter 5. Let $C \in \mathbb{N}^n$ hold these counts, where $n = |\mathcal{X}|$. The following convergence measures for the π-sampling method are used:

- $SUM\_ITERS = \sum_{i=1}^{n} C_i$,
- $AVG\_ITERS = \frac{1}{n} \cdot \sum_{i=1}^{n} C_i$,
- $MAX\_ITERS = max_{i \in [n]} C_i$.

[3] `https://gym.openai.com/envs/Pendulum-v0/`

Partition	Sum of Iterations	Average Iterations	Maximal Iterations
horizontal	111.00	37.00	37.00
vertical	102.00	34.00	34.00
equally_3_1	84.00	28.00	34.00
equally_3_2	106.00	35.33	36.00
markov_1	35.00	11.67	21.00
markov_2	35.00	11.67	17.00

Table 6.1: Convergence measures of the π-sampling method for the dynamic system from section 4.1. Both 3 subsets Markov partitions outperform the grid-like 3 subsets partitions.

The π-sampling method is repeated for each state space partition ten times and calculate the average as well as the variance for all three convergence measures. The variances are negligible small for all experiment runs.

The convergence behavior of the iterative policy evaluation schema is quantified by the following two measures:

- $MAX\_DIST = max_{i \in [n]} |V_i^k - V_i^{k-1}|$,
- $L2\_DIST = \left\| V^k - V^{k-1} \right\|_2$,

where $V^k, V^{k-1} \in \mathbb{R}^n$ are the value function estimates of the current iteration k and the previous estimate of iteration $k-1$. To evaluate the overall convergence behavior, we track both measures over all iterations k until ϵ-convergence occurs for a sufficiently small $\epsilon > 0$. For details see chapter 5.

Results - π-sampling method. Let the classical example of a dynamic system $(\mathbb{T}^2, \phi)$ as introduced in section 2.3 be the environment. Suppose that the agent follows the **0**-policy, i.e. $\pi(P_i) = 0$ for all $P_i \in \mathcal{X}$. We experiment with two different Markov partitions of $\mathbb{T}^2$. *markov_1* is the partition displayed in Figure 4.3 and *markov_2* is the partition displayed in Figure 4.1. Both Markov partitions have 3 subsets. Hence, we compare to different equally spaced baseline partitions with 3 subsets. *horizontal* and *vertical* are equally partitioned along the first or second dimension respectively. *equally_3_1* and *equally_3_2* are two partitions that are equally split along both dimensions at the same time.

Table 6.1 summarizes the convergence results of the executed π-sampling method for the constructed partitions. Clearly, the π-sampling method shows faster convergence behavior for both Markov partitions than all baseline partitions. Additionally, the probability estimates converge faster for the worst-case state quantified

Partition	Sum of Iterations	Average Iterations	Maximal Iterations
horizontal	453.00	64.71	79.00
vertical	360.0	51.43	60.00
equally_6_1	195.0	32.50	66.00
equally_6_2	210.0	35.00	52.00
markov	84.00	12.00	43.00

Table 6.2: Convergence measures of the π-sampling method for the dynamic system from section 4.1. The 7 subsets Markov partition outperforms even the smaller grid-like partitions.

by the MAX_ITERS measure. Furthermore, the overall required estimation updates over all states are significantly lower for both Markov partitions indicated by the SUM_ITERS measure.

Next, we analyze if and how the convergence behavior depends on the degree of the state space partition's granularity. Therefore, we construct a 7 subset Markov partition as displayed in Figure 4.5c. For comparison, we build *horizontal* and *vertical* partitions with 7 equally spaced subsets along one dimension. Further, *equally_6_1* and *equally_6_2* are 3×2 and 2×3 grids with 6 subsets equally spaced along both dimensions. Table 6.2 summarizes the convergence speed results. Again, the Markov partition induces a significantly faster average and overall convergence of the Monte Carlo algorithm compared to the grid-like partitions, although *equally_6_1* and *equally_6_2* consist only of 6 instead of 7 subsets.

Last, the construction process of the Markov partition displayed in Figure 4.5c is extended to another iteration. This again leads to a Markov partition as proven in section 4.2. This partition consists of 11 subsets. We build baseline partitions with 4×3 and 3×4 grids, namely *equally_12_1* and *equally_12_2*. Table 6.3 summarizes the convergence measure results. Similarly, also the fine-grained Markov partition with 11 subsets outperforms its competitor partitions by a significant margin in terms of average and worst-case convergence speed per state. In fact, the baseline partitions require more total iterations to converge to a probability estimate.

All in all, Markov partitions induce superior average and overall convergence speed of the Monte Carlo probability estimation compared to baseline partitions of comparable size and granularity. Moreover, the baseline and Markov partition's relative performance difference appears to be stable with increased granularity. Thus, the absolute effect of superior convergence speed could increase.

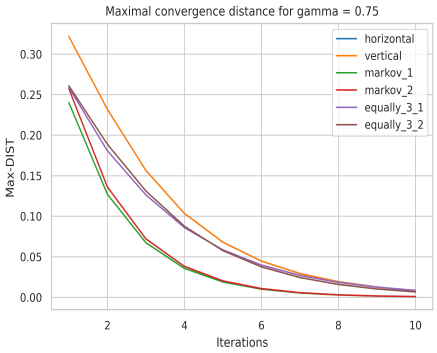
Partition	Sum of Iterations	Average Iterations	Maximal Iterations
horizontal	710.00	64.54	74.00
vertical	784.00	71.27	81.00
equally_12_1	417.00	34.75	52.00
equally_12_2	431.00	35.92	52.00
markov	105.20	9.56	28.10

Table 6.3: Convergence measures of the π-sampling method for the dynamic system from section 4.1. The 11 subsets Markov partition outperforms the grid-like partitions.

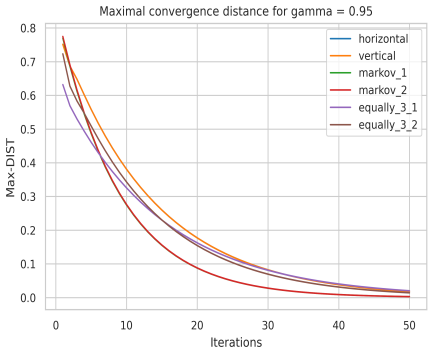
Results - Policy Evaluation. Let the environment, policy, and partitions be as above. We investigate two major questions regarding iterative policy evaluation. First, do Markov partitions lead to a less required number of iterations until ϵ-convergence to the true value function occurs? In the experiments, $\epsilon = 10^{-6}$ to have a precise approximation of the true value function under consideration of limited machine precision. Second, could Markov partitions lead to an accelerated convergence regime, i.e., with which speed shrinks the difference of two consecutive policy evaluation iteration steps $|V^k - V^{k-1}|$ to zero? If Markov partitions induce overall fewer iterations until convergence to the true value function, then the policy evaluation schema could be accelerated using Markov partitions. Additionally, if Markov partitions lead to an accelerated convergence regime in the first iterations of policy evaluation, an optimistic policy evaluation algorithm could be accelerated.

We analyze both aspects considering partitions of different granularity. Further, ten different discount factors are chosen equally distributed on the interval $(0.05, 0.95)$ to examine effects concerning increasing future long-term costs.

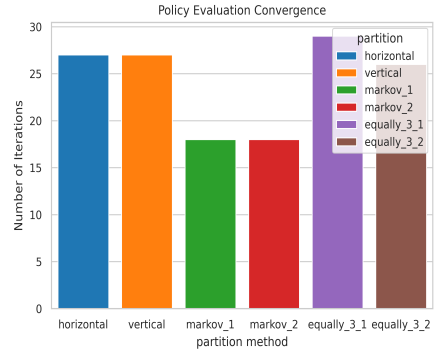
Figure 6.3 summarizes the findings for a state space partition with 3 subsets. Observe that for the two representatively selected discount factors $\gamma \in \{0.75, 0.95\}$, Figure 6.3a and Figure 6.3b clearly show an accelerated convergence regime under application of both Markov partitions. The same plots for the $L2\_DIST$ are not displayed since the results are almost identical. Moreover, Figure 6.3c and Figure 6.3d both show that the total number of iterations until ϵ-convergence occurs is significantly lower for both Markov partitions compared to all baseline partitions. The absolute speed up is enhanced with increasing discount factor γ. Last, Figure 6.3e shows that Markov partitions lead to a lower number of iterations until convergence for all chosen γ. Hence, an accelerated convergence regime shown in Figure 6.3a and Figure 6.3b is transferable to other choices of γ. The effect increases with γ as Figure 6.3f shows.



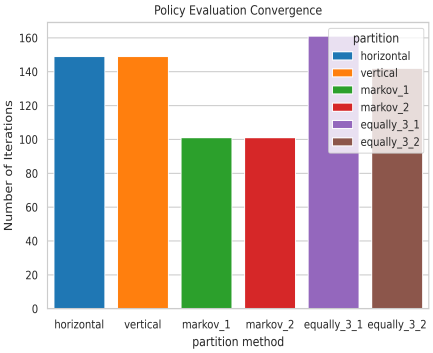
(a) Convergence regime for $\gamma = 0.75$



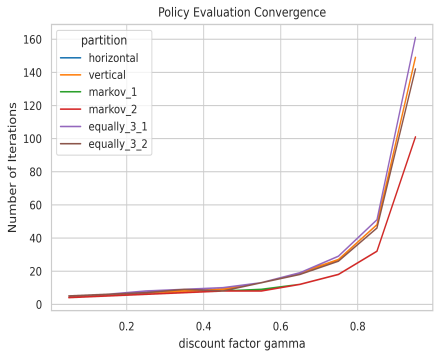
(b) Convergence regime for $\gamma = 0.95$



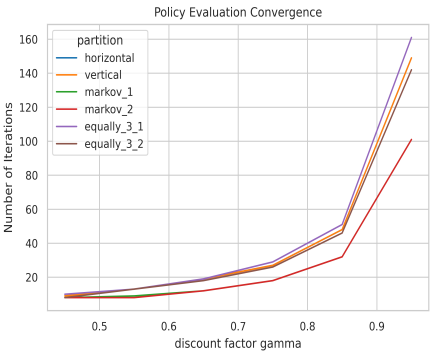
(c) Iterations until convergence for $\gamma = 0.75$



(d) Iterations until convergence for $\gamma = 0.95$

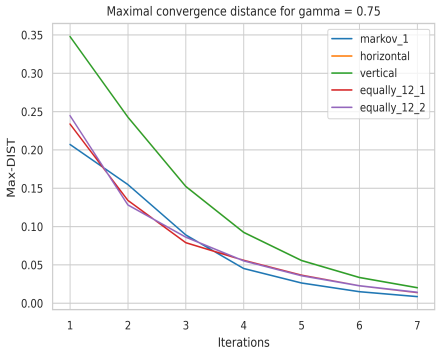


(e) Iterations until convergence over all γ

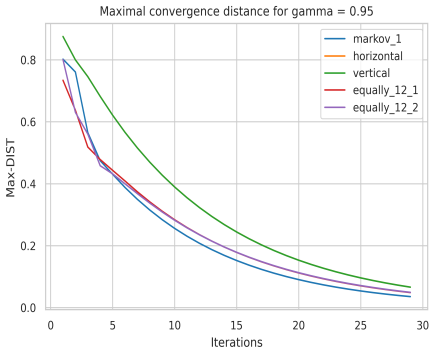


(f) Iterations until convergence for $\gamma > 0.4$

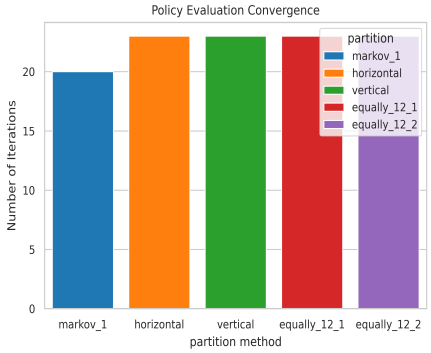
Figure 6.3: The convergence behavior for a 3 subset state space iterative partition policy evaluation is visualized. (a) + (b) show the MAX_DIST over the iterations for two fixed discount factors γ. (c) + (d) show the total number of iterations until ϵ-convergence to the true value function occurs for those two γ. (e) shows the number of iterations until ϵ-convergence over all $\gamma \in (0, 1)$ and (f) highlights the differences especially for more important future costs $\gamma \in (0.4, 1.0)$.



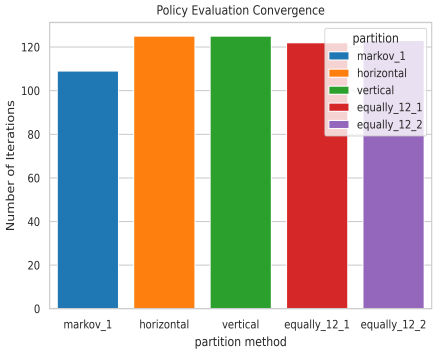
(a) Convergence regime for $\gamma = 0.75$



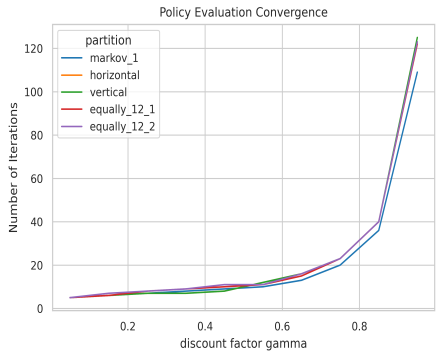
(b) Convergence regime for $\gamma = 0.95$



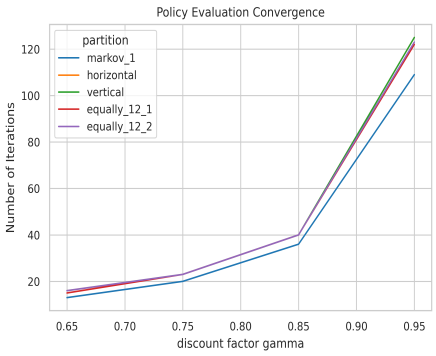
(c) Iterations until convergence for $\gamma = 0.75$



(d) Iterations until convergence for $\gamma = 0.95$



(e) Iterations until convergence over all γ



(f) Iterations until convergence for $\gamma > 0.6$

Figure 6.4: The convergence behavior for a 11 subset state space iterative partition policy evaluation is visualized. (a) + (b) show the MAX_DIST over the iterations for two fixed discount factors γ. (c) + (d) show the total number of iterations until ϵ-convergence to the true value function occurs for those two γ. (e) shows the number of iterations until ϵ-convergence over all $\gamma \in (0, 1)$ and (f) highlights the differences especially for more important future costs $\gamma \in (0.6, 1.0)$.

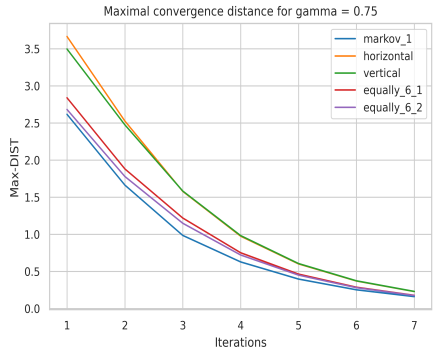
Partition	Sum of Iterations	Average Iterations	Maximal Iterations
horizontal	101.00	14.43	17.00
vertical	120.00	17.14	21.00
equally_6_1	113.00	18.83	26.00
equally_6_2	85.80	14.30	17.90
markov	83.00	11.86	15.00

Table 6.4: Convergence of the π-sampling method for the nonlinear system from [24] visualized in Figure 4.5d. The Markov partition slightly outperforms the grid-like partitions.

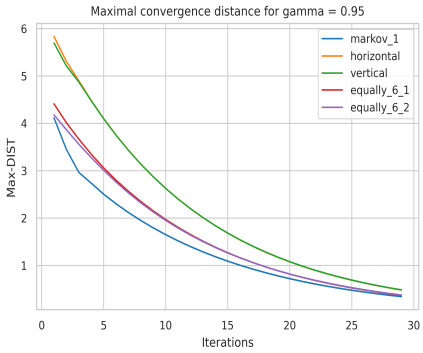
Figure 6.4 summarizes the main results for a partition with 11 subsets. The 7 subset Markov partition is also evaluated but not explicitly stated here since the results perfectly fit the overall observations. Again, Figure 6.4a and Figure 6.4b indicate that Markov partitions imply accelerated convergence to a true value function. The absolute convergence advantages increase with γ. However, the differences between Markov partitions and a few baseline partitions seem to shrink slightly with the state space discretizations' increasing granularity. The same analysis with the $L2\_DIST$ measure yields almost identical results. Furthermore, Figure 6.4c and Figure 6.4d give evidence for a superior overall ϵ-convergence of Markov partitions. Nevertheless, also for these measures, there is a slight trend that with further discretization granularity, the differences slightly shrink. In general, the convergence advantages of Markov partitions are still particularly evident for larger γ as Figure 6.4e and Figure 6.4f show.

Results - Nonlinear System Responses. Suppose that the linear system dynamics of the environment $(\mathbb{T}^2, f)$ are given by the diffeomorphism $f(x) = Ax = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} x$ known from the linear system example of [24]. Next, suppose that the agent follows a policy $\pi : \mathbb{T}^2 \to \mathbb{T}^2$ defined by $\pi(x) = \epsilon \begin{pmatrix} 1 & 0 \\ 2 & 0 \end{pmatrix} x$ for some constant $\epsilon \in (0, 1]$. Then, the overall system response $\phi : \mathbb{T}^2 \to \mathbb{T}^2$ is described by $\phi(x) = Ax + \pi(x)$. The setup is equivalent to the nonlinear example introduced by [24]. We construct a 7 subsets Markov partition for $\epsilon = 0.7$ as visualized in Figure 4.5d. As baseline partitions, again the regular grid-like partitions with 6 and 7 subsets are used.

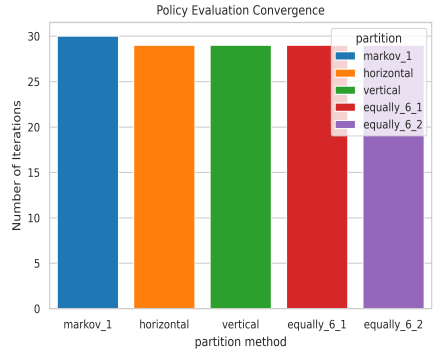
Table 6.4 summarizes the convergence results of the π-sampling method for the nonlinear system responses ϕ. The Markov partition enables a faster overall and average convergence of the Monte Carlo estimation. However, the absolute conver-



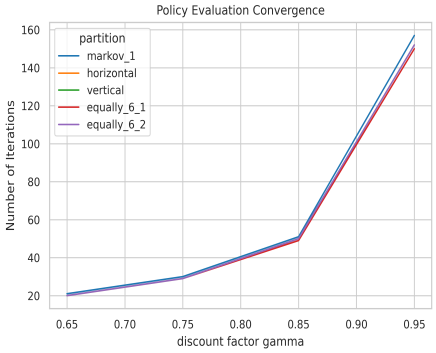
(a) Convergence regime for $\gamma = 0.75$



(b) Convergence regime for $\gamma = 0.95$



(c) Iterations until convergence for $\gamma = 0.75$



(d) Iterations until convergence for $\gamma > 0.6$

Figure 6.5: The convergence behavior of iterative policy evaluation for a 7 subset state space partition of a nonlinear system is visualized. (a) + (b) show the MAX_DIST over the iterations for two fixed discount factors γ. (c) shows the total number of iterations until ϵ-convergence to the true value function occurs. (d) shows the number of iterations until ϵ-convergence over all $\gamma \in (0.6, 1.0)$.

gence differences do not appear to be as substantial as within the linear systems above. Particularly, the *equally_6_2* shows only slight disadvantages. Furthermore, Figure 6.5 shows the convergence results of the iterative policy evaluation. Figure 6.5a and Figure 6.5b indicate that the Markov partition could impose small benefits in the first iterates of the convergence regime. Nonetheless, the overall convergence advantages of Markov partitions are not as evident as for the linear examples above. In particular, Figure 6.5c and Figure 6.5d show that there is no difference in the number of total iterations over all $\gamma \in (0, 1)$.

In general, the performance is highly dependent on the right choice of hyper-parameters in both algorithms. Further, the choices of those hyper-parameters mutually determine each other. Hence, it is even harder to find an overall suitable hyper-parameter configuration. Most crucial, the integration constant δ of the Markov partition construction algorithm must be drastically lowered to guarantee a sufficient first-order approximation of the nonlinear manifolds. Of course, such an increased sampling rate of points on these manifolds leads to a blow-up in the algorithm's run time. Setting $\delta = 10^{-4}$ is a trade-off between run time and approximation accuracy.

Furthermore, it is not straightforward to choose the number of sampled points c and the convergence threshold τ. If τ is set too low and c is set too high, then it is likely that by the first-order approximation induced imprecision at the boundaries of the partition subsets, sampling noise will occur. Samples $p \in P_i$ that are close to a boundary of P_i could transition into a wrong successor state P_j for $\phi(p) \in P_j$. The evaluation for which subset i, j it holds $p \in P_i$ and $\phi(p) \in P_j$ is particularly challenging for p close to ∂P_i. Hence, the choice of c and τ has to be balanced with δ, which implicitly determines the maximal exactness in the probability estimations. In the experiments, $\tau = 10^{-4}$ and $c = 600$ is a reasonable trade-off between overall accuracy and suppressing sampling noise. If sampling noise occurs, then for some or all $x_i, x_j \in \mathcal{X}$ with $\phi(P_i) \cap P_j = \emptyset$, $p(x_j|x_i, \pi(x)) > 0$ with some small noise value. This phenomenon destroys the efficiency of Markov partitions in capturing the probability measure $p(\cdot|x_i, \pi(x))$ compactly. Thus, for not carefully chosen hyper-parameters δ, c, τ, Markov partitions induce no superior performance.

Interpretation and Discussion. The experiments show faster convergence speed for Monte Carlo probability estimates given a Markov partition state space discretization. From a measure-theoretic point of view, Markov partitions and their induced 1-step shifts of finite type capture the probability measure p_π more efficiently. There are relations between the topological concept of Markov partitions and measure theory by multiple variational principles. For example, the topological entropy of finite type shifts is the maximal measure-theoretic entropy over all the dynamic system's measurable functions. Such examples could provide theoretical

evidence for Markov partitions' superior convergence performance. However, an in-depth analysis of relations between Markov partitions and measure-theoretic concepts is beyond this work's context. We refer to [30] for a detailed analysis.

Overall, Markov partitions induce sparser transition probability matrices compared to random grid-like partitions. Therefore, many states are not reachable from a current state. This speeds up the convergence of probability estimates. Thus, on average and worst-case, we discover superior convergence behavior when utilizing Markov partitions. Therefore, induced sparsity is part of a theoretical explanation for the experiment results. Furthermore, the sparsity could be beneficial for a closed-form solution of V_π. Let $G_\pi(x) = \mathbb{E}_{p_\pi(x'|x)}[g(x, \pi(x), x')]$ and P_π be the transition probability matrix while following a policy π. Then, $V_\pi = G_\pi + \gamma P_\pi V_\pi$ is the fixed point that yields the true value function V_π [8]. For this linear equation system with sparse matrices, there are computationally efficient methods [19]. Additionally, Markov partitions could induce accelerated temporal difference policy evaluation. Last, the convergence of the iterative policy evaluation schema itself increases. Experiments show a noticeable effect for larger γ.

However, most of the results are only empirical. Follow-up practical and theoretical questions are open to research. Further, Markov partitions impose strong assumptions. The phase space must be (two-dimensional) locally split hyperbolic everywhere. Next, Markov partitions are not straightforward to construct for all systems, as seen in chapter 3. Additionally, the superior convergence behavior is only applicable to a fixed policy evaluation step. It requires a state space partition that considers the system dynamics and the followed policy. Thus, it is not clear if such performance advantages (partially) carry over to a complete policy iteration.

6.3 Discussion of Real-World Applications

Hyperbolic dynamics is a well-developed theory with enormous progress in characterizing dynamic systems' long-term behavior that appears to have chaotic behavior. Further, the global structural stability of such systems is guaranteed. Hence, a hyperbolic dynamic system is agnostic to small random perturbations or noise. The noisy system is again hyperbolic and equipped with the same global behavior as the non-perturbed system. Such rich theorems are also the key to our algorithmic construction of Markov partitions. In the end, it allows the fusion of Markov partitions with MDPs and policy evaluation algorithms in chapter 5.

The structurally stable behavior of hyperbolic dynamic systems is desirable for many mechanical systems, e.g., if some production process suffers from small noise. Further, some physical systems are indeed endowed with hyperbolic struc-

ture, e.g., most chaotic systems. However, there are many real-world examples, e.g., in fluid dynamics [41], mechanics [15] or other fields of physics [12] that do not fit into the classical theory of uniform hyperbolicity as introduced in chapter 3. Thus, the strict notion of hyperbolicity is often softened in various physical systems [52]. However, it remains unclear if Markov partitions exist or can be constructed for these generalized system structures. The difficulties in applying hyperbolicity to real-world physical systems also explain that most literature focuses on examining elementary hyperbolic dynamic systems. Consequently, Markov partition constructions are exclusively performed for artificially constructed systems. Nevertheless, there are significant research efforts designing systems that fulfill hyperbolicity rather than searching for real-world examples occurring as hyperbolic systems [35] [42].

Furthermore, there are examples of control problems closely related to an agent performing optimal actions within a hyperbolic system. For example, hyperbolic dynamics are utilized to find fuel- and time-efficient orbits of a spacecraft from the earth to the moon [14]. Another example is a passively walking compass-gait walker that suffers from chaotic dynamics required to be controlled to achieve a stable walk. Some methods leverage the local hyperbolic structure to construct a state feedback controller stabilizing the fixed point of the respective Poincaré map [26]. Additionally, for examples filling the hyperbolic systems theory with examples from the physics domain, see [42].

Nevertheless, finding a Markov partition is hard. It is even more challenging for systems that cannot be described by differential equations. Markov partitions rely on the notion of continuous behavior over time. The partition boundaries are often highly convoluted or high dimensional. Real-world experiments with finite sample frequencies and finite precision measurements cannot meet these demanding assumptions. Thus, there is much work done to construct Markov partitions with finite-resolution orbits approximately. These methods approximately ensure a partition's Markov property and can be used to estimate global system properties as its entropy [55]. It feels natural to investigate if such an approximated Markov partition is also suited for DP-applications.

All in all, most real-world systems suffer from the restrictions Markov partitions pose on the phase space. Simple yet classical control examples cannot fulfill these strict assumptions. E.g., the famous DP-environment of a (damped) pendulum does not have a locally split hyperbolic structure everywhere[4]. However, its control goal coincides with a hyperbolic fixed point in its phase space [11]. The Hartman-Grobman theorem guarantees an approximation of the system in a neighborhood around this hyperbolic fixed point by linearizing its dynamics [56]. Hence, lineariz-

[4] See openai gym's pendulum-v0: `https://gym.openai.com/envs/Pendulum-v0/`

ing the system around its hyperbolic fixed point induces a locally split hyperbolic structure. This allows to construct a Markov partition of the phase space around its hyperbolic fixed point. Thus, if a system is ensured to stay in a sufficiently small neighborhood around its hyperbolic fixed point, an MDP can be constructed to learn a control law. This observation could provide a solution to apply our work to a broader class of system control problems.

7 Related Work

Our work on fusing symbolic dynamics and dynamic programming is three-folded, so is the related work. We utilize Markov partitions to generate expressive state space discretizations. Markov partitions establish the basis for deriving a Markov decision process (MDP) to execute approximate dynamic programming (ADP) algorithms in dynamic system environments.

First, Markov partitions have their roots in dynamic systems theory. Hence, primary literature to understand the main concepts and applications of Markov partitions is investigated. Additionally, work presenting results regarding the existence of Markov partitions and their limitations is reviewed. Second, Markov partitions are closely related to symbolic dynamics. Symbolic dynamics is a tool to transform a continuous dynamic system into a symbolic dynamic system by utilizing Markov partitions. Further, the long-term behavior and invariants of such symbolic systems are efficiently computable. Therefore, essential work in the field of symbolic dynamics and how it motivates our work is analyzed. Finally, we construct Markov partition state space representations supporting essential properties of an MDP. Thus, our work also fits into a corpus of work dedicated to representing and approximating state spaces in continuous dynamic programming and reinforcement learning settings. On the one hand, there are classical approaches, e.g., dimensionality reduction and feature engineering. On the other hand, more advanced techniques were invented, e.g., symbolic dynamic programming for first-order MDPs minimally partitioning a state space [16].

7.1 Origins in the Theory of Dynamic Systems

Gromov introduces hyperbolicity at the intersection of algebraic topology, geometry, and symbolic dynamics [27]. He states fundamental results about hyperbolicity in each domain and unifies the statements by drawing connections between the different fields. Markov partitions are introduced as an essential tool for this unification. An extensive analysis of [27] in our work shows that the hyperbolicity of dynamic systems is closely related to Markov partitions' existence. The rigorous study of locally split hyperbolic diffeomorphisms clarifies the limitations in constructing Markov partitions.

For an extensive summary of hyperbolic dynamic systems, we refer to [56]. They explain fundamental results about hyperbolic systems. Further, we highlight [28] as an intuitive introduction to differential equations and their connection to time-discrete diffeomorphisms over manifolds. Both resources provide crucial background knowledge to develop a clear picture of our work.

Last, [29] proposes an efficient method to approximate nonlinear invariant manifolds in a computer. Invariant manifolds represent the Markov partition boundaries in our developed construction procedure. Thus, [29] enables us to trace nonlinear invariant manifolds of a Markov partition in a computer.

7.2 Symbolic Dynamics and Markov Partitions

Lind and Marcus propose comprehensive literature for studying symbolic dynamics [44]. They present symbolic dynamics as a part of the dynamic systems theory and elaborate on concrete applications, e.g., data structures. Further, they connect dynamic and symbolic systems by introducing Markov partitions. Simple examples of arithmetic Markov partition constructions are highlighted. Hence, [44] motivates our theoretical work on symbolic dynamics. It is the starting point for our work in researching convenient state space discretization techniques applicable to ADP.

Over the years, the notion of Markov partitions initially shaped by Adler and Weiss [2] and extended by Bowen [18] has been adapted for non-invertible dynamical systems. So-called extended Markov partitions transfer the advantages of symbolic dynamics to a broader class of dynamic systems [57]. However, most Markov partition construction work requires tedious manual calculations that do not generalize to other systems.

Thus, we review the findings of [51] in connecting symbolic dynamics and Markov partitions. [51] briefly describes a semi-automated technique to construct Markov partitions for simple linear hyperbolic toral automorphisms. We investigate natural variations and extensions and show that these methods are equally valid. Additionally, we identify gaps in their general applicability and algorithmic formulation. Finally, this leads us to a more advanced approach.

[24] presents such an algorithmic procedure. Since our work requires building MDPs at scale to fuse symbolic dynamics and ADP, such an automatic Markov partition construction is one of our work's primary goals. Therefore, we expand [24]'s method. They fail in producing an end-to-end automized approach creating Markov partitions. On the one hand, the algorithm relies on knowledge about essential characteristics, such as fixed points. On the other hand, they require access to a video

graphic device to manually supervise the construction process. We fill the gaps missing to deliver a fully automated procedure. Moreover, we precisely formalize, justify, and implement our proposed extension of [24]'s work.

7.3 Representation Learning in Dynamic Programming

Much research in the community of dynamic programming and reinforcement learning is devoted to representing and approximating continuous systems. There are many classical approaches clustered around the term approximate dynamic programming [54]. The newest research in deep reinforcement learning tries to decouple the goal of learning a representation and learning a policy [58]. The bandwidth of research impressively shows the demand for solutions to the problem. Overall, representation learning in dynamic programming or reinforcement learning is two-folded. There are two curses of dimensionality in real-world applications. First, the state space is often continuous or high dimensional. Thus, a discretized or approximated representation in a computer is required. Second, the action space could be continuous.

Our work fits into the research line dedicated to the first curse of dimensionality, namely the representation of state spaces. The pool of techniques to represent continuous state spaces is large but divisible in two research directions. In general, we aim to learn an optimal policy of an agent fulfilling a given goal. On the one hand, there are problems for which a state transition probability model and a reward model are specified. At least, they can be estimated, e.g., by Monte Carlo techniques [8]. This class of problems is typically referred to as planning problems tackled by linear programming or ADP. On the other hand, some problems make it hard to specify the environment or probability model. Thus, one is limited to the environment's observable states or the agent's observable behavior under some actions in such an environment. These problems are typically called learning problems and are tackled with (deep) reinforcement learning techniques. Our work focuses on planning problems with continuous state spaces and clearly defined environments by known system dynamics. The theoretical work and experiments are focused on classical Monte Carlo based iterative policy evaluation methods [8].

Basic techniques to approximate a continuous state space $\mathcal{S}$ involve the construction of basis functions $\phi_1, ..., \phi_K$, where $x \mapsto (\phi_1(x), ..., \phi_K(x))^T$ for all basis functions and $x \in \mathcal{S}$. Hence, the dimensionality of $\mathcal{S}$ is reduced by defining such basis functions with $K \ll dim(\mathcal{S})$ [20]. These basis functions are often tedious to construct by hand and are highly task-dependent. That is why, for long years, efforts were put into automizing this process. For example, much work is proposed in learning lower-dimensional subspaces [39]. In general, an automatically con-

structed state space representation is also our work's goal, but it is of a slightly different motivation. We discretize the continuous state space by leveraging knowledge about the environment. Overall, we build a state space representation by utilizing a well-defined construction procedure instead of formulating a vague and less explainable learning task.

Most frequently, state spaces are discretized by primitive regular grid-like partitions without paying attention to the effects on the ADP algorithm. However, one aims to guarantee "similarity" of aggregated states and smoothness of the optimal costs, such that converging to an optimal policy is facilitated [9]. In many physical, real-world applications, an agent interacts in an environment precisely described by a dynamic system. Thus, we investigate if Markov partitions are more sophisticated for discretizing a continuous state space and achieving superior ADP performance. To the best of our knowledge, there is no work utilizing Markov partitions or similar concepts to construct sophisticated MDPs for continuous state space problems.

8 Conclusion and Future Work

We propose a well-studied fusion of symbolic dynamics, particularly Markov partitions, with approximate dynamic programming (ADP). Therefore, we introduce an algorithmic procedure for building Markov partitions of various dynamic systems. Based on these state space representations, we derive Markov decision processes (MDP) utilized in classical Monte Carlo based iterative policy evaluation algorithms. We perform an extensive analysis of Markov partitions' strengths and limitations. Markov partitions naturally arise for locally split hyperbolic systems, e.g., Anosov systems, as highlighted in chapter 3. Further, the presence of Markov partitions is proven even for a broader class of Axiom A systems [18], but seem only to be naturally constructible for locally split hyperbolic systems.

In general, local hyperbolicity across the whole state space is a beneficial property since it leads to the system's global structural stability. Global structural stability enables a comparison of systems by quasimorphisms. Thus, under small perturbations of the system, we can still directly transfer global properties and invariants. However, Markov rectangle boundaries are not necessarily simple to represent or calculate in a computer. Bowen shows for diffeomorphisms over the torus of dimensionality larger two that boundaries are not smooth. Hence, fractalization of partition boundaries occurs [17]. This is a substantial restriction since three- or higher-dimensional systems are typical in real-world applications.

Furthermore, most Markov partitions known from the literature suffer from tedious arithmetic calculations that do not apply to other systems. We bypass these shortcomings by proposing an algorithmic procedure to build Markov partitions in a computer. Therefore, we analyze the hand-crafted construction work of [51], which finally leads us to an automized approach by extending the work of [24]. With our method, no prior knowledge of the dynamic system, manual calculations, or visual feedback loops are required. Removing these hand-operated steps is vital facilitation concerning the construction of MDPs at scale. Moreover, we can derive shifts of finite type for all systems for which we construct such Markov partitions. Hence, we can also apply symbolic dynamics techniques to compute the continuous dynamic system's global invariants such as periodicity or entropy.

However, our Markov partition algorithm admits a few limitations. First, it is tailored to the torus and its unique challenges regarding data structures to trace

piecewise continuous partition boundaries. An adaption of the algorithm to other quotient groups has to be invented. Second, we rely on the strong assumption of local split hyperbolicity at any point in the state space. Hence, contracting and expanding eigenvectors of the tangent space's jacobian must exist to approximate nonlinear boundaries. Third, assembling Markov partitions in a computer for tori of dimension three or higher is, to some extent, infeasible [17]. Thus, we do not extend the algorithm beyond state space dimensionality larger two, which excludes many real-world applications.

Next, we infer MDPs based on Markov partitions. Therefore, we formalize an MDP and discuss the challenges in calculating a state transition probability kernel. We propose and implement two Monte Carlo methods to overcome these challenges. Finally, we fuse the notion of Markov partitions, their respective MDPs, and Monte Carlo based policy evaluation algorithms. However, the partition's Markov property is only fulfilled for a fixed policy evaluation step. The application of Markov partitions to system dynamics without paying attention to the current agent's policy neutralizes the Markov property. Hence, if applied to the whole policy iteration algorithm with a static state space partition, we lose the benefits of Markov partitions compared to poorly justifiable regular grid-like partitions.

The performed experiments show that applications to everyday robotics or system control settings are challenging. These systems do not necessarily meet the required assumptions to build Markov partitions. Nevertheless, our experiments also show that Monte Carlo based iterative policy evaluation algorithms could profit from Markov partitions. Markov partitions usually lead to faster convergence of Monte Carlo estimates. Additionally, iterative policy evaluation's convergence speed increases, at least for linear systems. Increased convergence speed is especially interesting for applying optimistic policy iteration. Further, we postulate that these convergence advantages could be transferred to temporal difference policy evaluation. However, we experience difficulties in reliably generating superior convergence of Markov partitions for highly nonlinear systems. The induced approximation error of the nonlinear partition boundaries affects the estimation of the probability kernel. Poor Monte Carlo algorithm's hyperparameter choices could neutralize the Markov partition's measure-theoretic advantages.

Overall, many questions are open for follow-up research. We identify three main blocks from practical short term research to long term theoretical research. First, one should facilitate the usability of the proposed Markov partition construction algorithm. The adaption to other quotient groups besides the torus and automatically performed validations of the Markov property are helpful. These features extend the applicability without in-depth user-knowledge about Markov partitions.

Second, more extensive experiments enhance the quantitative evidence for our results. On the one hand, we should perform experiments that show if an initially fixed Markov partition is still superior to regular grid-like partitions. On the other hand, a more extensive study of suitable real-world applications and systems is required. It is unclear which (higher dimensional) systems provide the assumptions of local split hyperbolicity. Additionally, tracing of invariant manifolds in higher dimensions has to be implemented. [29] proposes an efficient algorithm.

Third, extending our framework beyond fixed policies is necessary to apply it to a complete policy iteration algorithm. A simple but flawed workaround is to fix Markov partitions for system dynamics initially. A long-term solution demands research on how to ensure hyperbolicity under changing policies. Only slightly changing policies could ensure the Markov property. One could utilize the hyperbolic system's global structural stability. Therefore, interpret small variations in policies as perturbations to the system that do not change the general hyperbolic structure. Last, it has to be examined if convergence to an optimal policy is guaranteed when Markov partitions are recomputed after each update of the greedily induced policy.

Bibliography

[1] P. Abbeel and A. Y. Ng. “Apprenticeship learning via inverse reinforcement learning”. In: *Proceedings of the twenty-first international conference on Machine learning*. 2004, p. 1.

[2] R. L. Adler and B. Weiss. “Entropy, a complete metric invariant for automorphisms of the torus”. In: *Proceedings of the National Academy of Sciences of the United States of America* 57(6) (1967), p. 1573.

[3] H. Alonso, A. A. Bliznyuk, and J. E. Gready. “Combining docking and molecular dynamic simulations in drug design”. In: *Medicinal research reviews* 26(5) (2006), pp. 531–568.

[4] A. Amon. “Nonlinear dynamics”. Ph.D. thesis. Université Rennes 1, 2007.

[5] N. Bäuerle and U. Rieder. “Markov decision processes”. In: *Jahresbericht der Deutschen Mathematiker-Vereinigung* 112(4) (2010), pp. 217–243.

[6] S. Behnia, A. Akhshani, S. Ahadpour, H. Mahmodi, and A. Akhavan. “A fast chaotic encryption scheme based on piecewise nonlinear chaotic maps”. In: *Physics Letters A* 366(4-5) (2007), pp. 391–396.

[7] D. P. BERTSEKAS. “Approximate Dynamic Programming”. In: (2012).

[8] D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, and D. P. Bertsekas. *Dynamic programming and optimal control*. Vol. 1. (2). Athena scientific Belmont, MA, 1995.

[9] D. P. Bertsekas and J. N. Tsitsiklis. “Neuro-dynamic programming: an overview”. In: *Proceedings of 1995 34th IEEE Conference on Decision and Control*. Vol. 1. IEEE. 1995, pp. 560–564.

[10] D. P. Bertsekas and H. Yu. “Distributed asynchronous policy iteration in dynamic programming”. In: *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE. 2010, pp. 1368–1375.

[11] J. Bevivino. “The path from the simple pendulum to chaos”. In: *Dynamics at the Horsetooth* 1(1) (2009), pp. 1–24.

[12] I. A. Bizyaev, A. V. Borisov, and A. O. Kazakov. “Dynamics of the Suslov problem in a gravitational field: Reversal and strange attractors”. In: *Regular and Chaotic Dynamics* 20(5) (2015), pp. 605–626.

[13] A. Bobenko. *Differentialgeometrie von Kurven und Flächen*. URL: https://page.math.tu-berlin.de/~bobenko/Lehre/Skripte/KuF.pdf. Last visited on 2020/09/08. 2006.

[14] E. M. Bollt and J. D. Meiss. "Targeting chaotic orbits to the Moon through recurrence". In: *Physics Letters A* 204(5-6) (1995), pp. 373–378.

[15] A. V. Borisov and I. S. Mamaev. "Strange attractors in rattleback dynamics". In: *Physics-Uspekhi* 46(4) (2003), p. 393.

[16] C. Boutilier, R. Reiter, and B. Price. "Symbolic dynamic programming for first-order MDPs". In: *IJCAI*. Vol. 1. 2001, pp. 690–700.

[17] R. Bowen. "Markov partitions are not smooth". In: *Proceedings of the American Mathematical Society* (1978), pp. 130–132.

[18] R. Bowen. "Markov partitions for Axiom A diffeomorphisms". In: *American Journal of Mathematics* 92(3) (1970), pp. 725–747.

[19] T. A. Davis. *Direct methods for sparse linear systems*. SIAM, 2006.

[20] D. P. De Farias and B. Van Roy. "The linear programming approach to approximate dynamic programming". In: *Operations research* 51(6) (2003), pp. 850–865.

[21] J. G. De Jalon and E. Bayo. *Kinematic and dynamic simulation of multibody systems: the real-time challenge*. Springer Science & Business Media, 2012.

[22] A. De Madrid, S. Dormido, and F. Morilla. "Reduction of the dimensionality of dynamic programming: A case study". In: *Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251)*. Vol. 4. IEEE. 1999, pp. 2852–2856.

[23] R. V. Florian. "Correct equations for the dynamics of the cart-pole system". In: *Center for Cognitive and Neural Studies (Coneural), Romania* (2007).

[24] V. Franceschini and F. Zironi. "On constructing Markov partitions by computer". In: *Journal of statistical physics* 40(1-2) (1985), pp. 69–91.

[25] A. Graser. "Movingpandas: Efficient structures for movement data in python". In: *GIForum* 1 (2019), pp. 54–68.

[26] H. Gritli, N. Khraief, and S. Belghith. "Chaos control in passive walking dynamics of a compass-gait model". In: *Communications in Nonlinear Science and Numerical Simulation* 18(8) (2013), pp. 2048–2065.

[27] M. Gromov. "Hyperbolic dynamics, Markov partitions and Symbolic Categories, Chapters 1 and 2." In: (2016).

[28] J. Guckenheimer and P. Holmes. *Nonlinear oscillations, dynamical systems, and bifurcations of vector fields*. Vol. 42. Springer Science & Business Media, 2013.

[29] J. Guckenheimer and A. Vladimirsky. “A fast method for approximating invariant manifolds”. In: *SIAM Journal on Applied Dynamical Systems* 3(3) (2004), pp. 232–260.

[30] S. A. HARDER. “GIBBS MEASURES AND SYMBOLIC DYNAMICS”. In: ().

[31] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. “Deep reinforcement learning that matters”. In: *arXiv preprint arXiv:1709.06560* (2017).

[32] G. Hinton. “Where do features come from?” In: *Cognitive science* 38(6) (2014), pp. 1078–1101.

[33] G. Hummer and A. Szabo. “Optimal dimensionality reduction of multistate kinetic and Markov-state models”. In: *The Journal of Physical Chemistry B* 119(29) (2015), pp. 9029–9037.

[34] E. Infeld and G. Rowlands. *Nonlinear waves, solitons and chaos*. Cambridge university press, 2000.

[35] O. B. Isaeva, A. Y. Jalnine, and S. P. Kuznetsov. “Arnold’s cat map dynamics in a system of coupled nonautonomous van der Pol oscillators”. In: *Physical Review E* 74(4) (2006), p. 046207.

[36] R. Islam, P. Henderson, M. Gomrokchi, and D. Precup. “Reproducibility of benchmarked deep reinforcement learning tasks for continuous control”. In: *arXiv preprint arXiv:1708.04133* (2017).

[37] A. Jain and G. Stock. “Identifying metastable states of folding proteins”. In: *Journal of chemical theory and computation* 8(10) (2012), pp. 3810–3819.

[38] W. Just and F. Vivaldi. *Dynamical Systems*. URL: `http://www.maths.qmul.ac.uk/~fvivaldi/teaching/ltcc_dyn/notes.pdf`. Last visited on 2020/08/20. 2018.

[39] P. W. Keller, S. Mannor, and D. Precup. “Automatic basis function construction for approximate dynamic programming and reinforcement learning”. In: *Proceedings of the 23rd international conference on Machine learning*. 2006, pp. 449–456.

[40] J. Kober, J. A. Bagnell, and J. Peters. “Reinforcement learning in robotics: A survey”. In: *The International Journal of Robotics Research* 32(11) (2013), pp. 1238–1274.

[41] S. P. Kuznetsov. “Plate falling in a fluid: Regular and chaotic dynamics of finite-dimensional models”. In: *Regular and Chaotic Dynamics* 20(3) (2015), pp. 345–382.

[42] S. P. Kuznetsov and V. P. Kruglov. “On some simple examples of mechanical systems with hyperbolic chaos”. In: *Proceedings of the Steklov Institute of Mathematics* 297(1) (2017), pp. 208–234.

[43] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis. “Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers”. In: *IEEE Control Systems Magazine* 32(6) (2012), pp. 76–105.

[44] D. Lind, B. Marcus, L. Douglas, M. Brian, et al. *An introduction to symbolic dynamics and coding*. Cambridge university press, 1995.

[45] D. Maclaurin, D. Duvenaud, and R. P. Adams. “Autograd: Effortless gradients in numpy”. In: *ICML 2015 AutoML Workshop*. Vol. 238. 2015, p. 5.

[46] M. J. Mataric. “Reward functions for accelerated learning”. In: *Machine learning proceedings 1994*. Elsevier, 1994, pp. 181–189.

[47] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. “Playing atari with deep reinforcement learning”. In: *arXiv preprint arXiv:1312.5602* (2013).

[48] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. “Human-level control through deep reinforcement learning”. In: *nature* 518(7540) (2015), pp. 529–533.

[49] J. J. Moré. “The Levenberg-Marquardt algorithm: implementation and theory”. In: *Numerical analysis*. Springer, 1978, pp. 105–116.

[50] D. Nagel, A. Weber, B. Lickert, and G. Stock. “Dynamical coring of Markov state models”. In: *The Journal of chemical physics* 150(9) (2019), p. 094111.

[51] N. Nijimbere. “Markov partition and symbolic dynamics of hyperbolic toral automorphisms”. Ph.D. thesis. Faculty of Science and Engineering, 2017.

[52] Y. B. Pesin and Y. B. Pesin. *Lectures on partial hyperbolicity and stable ergodicity*. Vol. 34. European Mathematical Society, 2004.

[53] M. A. Porter, P. G. Kevrekidis, and C. Daraio. “Granular crystals: Nonlinear dynamics meets materials engineering”. In: *Physics Today* 68(LA-UR-15-21727) (2015).

[54] W. B. Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*. Vol. 703. John Wiley & Sons, 2007.

[55] N. Rubido, C. Grebogi, and M. S. Baptista. “Entropy-based generating Markov partitions for complex systems”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 28(3) (2018), p. 033611.

[56] M. Shub. *Global stability of dynamical systems*. Springer Science & Business Media, 2013.

[57] K. Stolot. “An extension of Markov partitions for a certain toral endomorphism.” In: *Zeszyty Naukowe Uniwersytetu Jagiellońskiego. Universitatis Iagellonicae Acta Mathematica* 1255 (2001), pp. 263–279.

[58] A. Stooke, K. Lee, P. Abbeel, and M. Laskin. “Decoupling Representation Learning from Reinforcement Learning”. In: *arXiv preprint arXiv:2009.08319* (2020).