INSTITUTE FOR COMPUTATIONAL PHYSICS

UNIVERSITY OF STUTTGART

SIMULATION TECHNOLOGY DEGREE COURSE

Master thesis

Simulating Stochastic Processes with Variational Quantum Circuits

Examiner

Prof. Dr. Christian HOLM

Institute for Computational Physics

Submitted by

Author	Daniel FINK
Matriculation number	3148684
SimTech-Nr.	87
Submission date	February 22, 2022

Preface and Declaration on Autonomy

This thesis was created as part of the Simulation Technology master degree course in the winter semester of 2021/2022 as a collaboration between the Institute of Computational Physics from the University of Stuttgart (US) and the AG Eisert from the Free University of Berlin (FUB). The advisors of the project were Prof. Dr. Jens Eisert (FUB), Prof. Dr. Christian Holm (US), Dr. Nora Tischler (FUB), Dr. Ryan Sweke (FUB), and M.Sc. Paul Fährmann (FUB).

I would first like to express my thanks to Prof. Jens Eisert, who made it possible for me to carry out my work as this collaboration and whose expertise was invaluable in formulating the research questions of this work. Moreover, I want to thank you for giving me insights into your wonderful research group in Berlin.

Furthermore, I would like to express my gratitude to the members of the AG Eisert. I would particularly like to single out Nora Tischler, Ryan Sweke, and Paul Fährmann for their extraordinary support throughout the entire project. I have enjoyed every single meeting with you and it has always moved me forward. Even in times when it was very stressful, you were always there to help and advise me.

In addition, I would like to thank the Quantum Machine Learning group of the AG Eisert. At each meeting, I was able to learn a lot and the subsequent discussions inspired me enormously.

I hereby declare that I wrote this master thesis independently and did not make use of any support or sources other than those mentioned in the paper.

Daniel Fink

Stuttgart, February 22, 2022

Abstract

Simulating future outcomes based on past observations is a key task in predictive modeling and has found application in many areas ranging from neuroscience to the modeling of financial markets. The classical provably optimal models for stationary stochastic processes are so-called ϵ-machines, which have the structure of a unifilar hidden Markov model and offer a minimal set of internal states. However, these models are not optimal in the quantum setting, i.e., when the models have access to quantum devices. The methods proposed so far for quantum predictive models rely either on the knowledge of an ϵ-machine, or on learning a classical representation thereof, which is memory inefficient since it requires exponentially many resources in the Markov order. Meanwhile, variational quantum algorithms (VQAs) are a promising approach for using near-term quantum devices to tackle problems arising from many different areas in science and technology. Within this work, we propose a VQA for learning quantum predictive models directly from data on a quantum computer. The learning algorithm is inspired by recent developments in the area of implicit generative modeling, where a kernel-based two-sample-test, called maximum mean discrepancy (MMD), is used as a cost function. A major challenge of learning predictive models is to ensure that arbitrarily many time steps can be simulated accurately. For this purpose, we propose a quantum post-processing step that yields a regularization term for the cost function and penalizes models with a large set of internal states. As a proof of concept, we apply the algorithm to a stationary stochastic process and show that the regularization leads to a small set of internal states and a constantly good simulation performance over multiple future time steps, measured in the Kullback-Leibler divergence and the total variation distance.

Contents

List of Figures

List of Tables

1 Introduction

Can we predict the future based on past observations? In certain scenarios, this is the case, for example in classical mechanics where the trajectory of a particle can be uniquely determined based on the location and the momentum. However, if the system to be modeled becomes more complex, it may not be possible to include all influences, such that the future can be predicted *exactly*. Moreover, many processes observed in nature behave *stochastically* and therefore do not exhibit one unique trajectory, but only a probability for the occurrence of certain events. For such stochastic processes, simulation techniques can be used to show *possible* future outcomes. Nowadays, simulations are a major component in many different areas, rising from the natural sciences of biology, chemistry, and physics to the areas of finance, sports, and social sciences. Here, simulating is commonly understood as taking a model with given initial starting conditions, and showing possibilities for how a system evolves in time. Such simulation models for stochastic processes are called *predictive models* [1] since they show possible futures based on past observations. The more complex these processes are, the more complex a predictive model must be to faithfully simulate possible futures [2]. However, these models can also be used to gain more insights into the process itself [3–6], and it is thus of great interest to build simple models and to avoid unnecessary complexity, which follows the principle of Occam's razor. In general, a predictive model must use some memory to store relevant information about the past of a process [7], which is related to the process's Markov order [2] and can be associated with some internal states of the model. The simulation of more complex stochastic processes thus has higher memory requirements [2], i.e., the models require more internal state. A limited memory size therefore necessarily leads to statistical errors during the simulation. These memory-limited models can be seen as approximate predictive models since they only approximate the true behavior of the underlying stochastic process. Furthermore, such approximate models also arise when a predictive model is learned based only on finitely many observations of a process [8–11]. However, recent developments in the domain of quantum computing have shown that these errors can be reduced if the model has access to quantum devices [11]. Yet, the proposed methods so far for obtaining such *quantum predictive models* [11, 12] are memory inefficient, since they make use of classical representations of the models, which require exponentially many resources in the Markov order. Thus, these approaches are insufficient for practical applications.

Meanwhile, in the area of quantum machine learning, an approach for training quantum models directly on quantum devices shows promise, which is known as variational quantum algorithms (VQAs). Here, a quantum model is given as a quantum circuit that is executed directly on a quantum device. The circuit itself offers trainable parameters and is called a variational quantum circuit (VQC). With the use of classical resources, the parameters of the model are trained in a hybrid quantum-classical optimization procedure and it is shown that such algorithms can already be applied to near-term quantum devices [13–16]. Learning a predictive model directly on quantum devices can be memory efficient, since the model itself is given as a quantum circuit, and no classical representation is needed. Therefore, such an approach can be beneficial for practical applications. Thus, the overall goal of the work at hand is to develop a variational quantum learning algorithm for approximate predictive models. Here, we take as input only some observations of a given stochastic process, which offers a way to directly obtain a model based only on data and without further knowledge of the process itself.

Within this work, we focus on discrete-time stochastic processes. These are processes that emit an output symbol x from some alphabet $A \subset \mathbb{N}_0$ at discrete points in time $t \in \mathbb{Z}$. Moreover, we consider stationary processes, which means that the process's unconditional joint probability distribution is the same for any time step $t \in \mathbb{Z}$, as explained later in more detail in Section 2.1.2. To develop a quantum learning algorithm for such processes, we first define a class of

stochastic processes, namely the period-N uniform renewal processes [11], that are considered in detail throughout this work. Based on that, we derive an *ansatz* for the variational quantum circuit, which is the set and structure of quantum operations that are performed on a quantum computer. This ansatz offers a similar structure to already proposed ansätze in the field of supervised machine learning [11, 17, 18]. Next, we define a cost function for the learning algorithm, where one part is taken by the so-called maximum mean discrepancy (MMD) [19], which is a kernel-based two-sample test that has already been successfully used in the field of implicit generative modeling [18, 20]. The other part of the cost function is a regularization term, which is specifically chosen to learn quantum predictive models and we show that it can be computed based on a quantum post-processing step that is applied only during the training. Furthermore, we derive expressions for the gradient of the proposed cost function and show that it can be estimated efficiently with the use of *similar* quantum circuits. For this purpose, we use a common technique from the field of quantum machine learning, called the *parameter-shift rule* [21], but extend it to be applicable for our ansatz.
The output of our learning algorithm is a set of parameters that can be used together with the quantum circuit to simulate arbitrarily many future time steps of a stochastic process. We validate the algorithm by applying two versions of it to the period-2 uniform renewal process and study the learned models. Here, one version takes only the MMD as a cost function, whereas the other version also includes the proposed regularization term. We show that a model training solely based on the MMD can only simulate one future time step accurately, while the other version shows a constantly good performance over multiple future time steps. Herein, the simulation performance is measured with the Kullback-Leibler (KL) divergence and the total variation (TV) distance – two commonly used distance measures for probability distributions [11, 17]. The best model learned by our algorithm offers a value for the KL divergence of less than 10^{-4} and the TV distance does not exceed a value of 10^{-2} over 15 simulation steps. Moreover, we visualize the internal states of the quantum models and show that using our proposed regularization term leads to a small set of internal states, which is in particular responsible for a consistently good simulation performance over multiple future time steps.

The work at hand is structured as follows. First, the basics of stochastic processes, quantum computing, and predictive models are introduced in Chapter 2. Additionally, a literature review of related works on classical and quantum predictive models, as well as an overview of variational quantum algorithms is given. Next, in Chapter 3, the methodology of developing a variational quantum learning algorithm is presented. This includes in particular the introduction of the used validation metrics, i.e., the KL divergence and TV distance, as well as the definition of the cost function. The latter also includes the definition of the proposed regularization term as well as a way how it can be computed based on a quantum post-processing step. Moreover, the construction of a suitable quantum circuit that serves as an ansatz is discussed. Furthermore, the parameter-shift rule is extended and expressions for the gradient of the cost function are derived. Next, the results of the numerical experiments are presented and discussed in Chapter 4. Here, we analyze the inherent stochastic error of the MMD, which offers a way to determine what values can be expected for the cost function, and present the validation results afterward. This includes the simulation performance over multiple future time steps as well as visualizing the internal quantum states of the model. Lastly, the work is concluded and an outlook is given in Chapter 5.

2 Basics and Literature Review

Within this chapter, we introduce the definitions and the basic theory related to stochastic processes and quantum computation. Additionally, methods and results of prior publications related to simulating stochastic processes with quantum devices and the use of variational quantum circuits are presented. The chapter is arranged as follows. In Section 2.1, we start with explaining the foundations of stochastic processes, which includes the concepts of measure theory and probabilities in Section 2.1.1, that leads to the formal definition of stochastic processes in Section 2.1.2. Distance measures for probability distributions that are crucial for this work are presented in Section 2.1.3. Afterward, the fundamental principles of quantum computation are explained in Section 2.2, which includes quantum measurements, quantum gates, and quantum circuits, that lead to the framework of variational quantum algorithms in Section 2.2.4. An overview of predictive models is given in Section 2.3, where a distinction is made between exact and approximate models. In Section 2.4, a brief overview about implicit generative models is given, which includes in particular the maximum mean discrepancy in Section 2.4.1 and a discussion about the relation to predictive models in Section 2.4.2. At the end of this chapter, we briefly summarize the related works discussed earlier and outline which components are put together and built on top of another to develop a quantum learning algorithm for predictive models.

2.1 Foundations of Stochastic Processes

The goal of this section is to formally define a stochastic process. Therefore, necessary definitions from measure theory are introduced first. Afterwards, based on these definitions, the concepts of probability theory are briefly explained. Finally, we formally define stochastic processes based on these concepts.

2.1.1 Measure Theory and Probabilities

The following definitions can be found in introductory textbooks about measure theory and probabilities such as "A Modern Approach to Probability Theory" by B. Fristedt and L. Gray [22]. Within this section and throughout this work, only discrete probability spaces are considered.

Given a set Ω of possible outcomes of a stochastic experiment, the starting point in measure theory is the definition of measurable subsets of Ω, which are characterized by a quantity called the *σ-algebra.*

Definition 2.1. Let Ω be an arbitrary set. A σ-algebra of Ω is a set $F \subseteq 2^{\Omega}$ fulfilling $\Omega, \emptyset \in F$ and is closed under complementation and countable unions. The members $A \in F$ are said to be *measurable* with respect to F.

In the above definition, 2^{Ω} refers to the power set of Ω. Based on the σ-algebra, *measurable spaces* can be defined as follows:

Definition 2.2. A measurable space is a pair (Ω, F), where Ω is a non-empty set and $F \subseteq 2^{\Omega}$ is a σ-algebra.

Adding a *probability measure* to a measurable space yields a *probability space*:

Definition 2.3. A probability space is a triple (Ω, F, P) consisting of a measurable space (Ω, F) and a probability measure P, which is a function $P : F \to [0,1]$, such that $P(\Omega) = 1$ and

$$P\Big(\bigcup_{m=1}^{\infty} A_m\Big) = \sum_{m=1}^{\infty} P(A_m) \tag{1}$$

for each pairwise disjoint sequence $(A_m)_{m\in\mathbb{N}}$ of members of F. For $A \in F$, $P(A)$ is called the *probability of A.*

Based on that, mappings between measurable spaces can be defined, leading to the concept of a *random variable*:

Definition 2.4. Let (Ω, F) and (Ψ, G) be measurable spaces. A *measurable function* from (Ω, F) to (Ψ, G) is a function $X : \Omega \to \Psi$ such that $X^{-1}(B) \in F$ for every $B \in G$. If (Ω, F, P) is a probability space, then X is called a random variable.

With the use of random variables, stochastic experiments can be modeled formally. Of central importance here is the *probability distribution* of a random variable:

Definition 2.5. Let X be a random variable from (Ω, F, P) to (Ψ, G). Then, the function P_X defined as $P_X : G \to [0,1]$, $P_X(A) = P(X^{-1}(A))$ is a probability measure and (Ψ, G, P_X) a probability space. P_X is called the probability distribution of X.

The probability distribution P_X is a probability measure by definition. On the other hand, using the identity map as a random variable, a probability distribution becomes a probability measure and the terms probability measure and probability distribution become essentially synonyms. Thus, one usually does not care about the underlying probability space and just talks about the random variable itself [22]. Following this, the probability distribution of a random variable X is simply denoted as P and the event that X takes the value $x \in \Psi$ is denoted as $X = x$. Thus, one writes $P(X = x) = P\big(X^{-1}(x)\big)$.
The definition of a probability distribution can be extended to take multiple random variables into account, leading to the definition of the *joint (probability) distribution*:

Definition 2.6. Let (Ω, F, P) be a probability space and $(\Psi_1, G_1), (\Psi_2, G_2)$ be measurable spaces. Moreover, let X_1 and X_2 be random variables, both defined on (Ω, F, P), with values in (Ψ_1, G_1) and (Ψ_2, G_2), respectively. The joint probability distribution of X_1 and X_2 is the probability measure P_{X_1,X_2} defined on $(\Omega \times \Omega, F \otimes F)$ by

$$P_{X_1,X_2}(A) = P\big((X_1, X_2) \in A\big) \tag{2}$$

for all $A \in F \otimes F$, where $F \otimes F$ denotes the σ-algebra on the Cartesian product $\Omega \times \Omega$.

Similar to the case with two random variables, this can be extended for finitely many random variables $X_1, X_2, \ldots, X_n, n \in \mathbb{N}$. Additionally, having n random variables X_i and n possible outcomes $x_i \in F$, one simply writes $P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$ for the value of the joint distribution. The *independence* of random variables is characterized by their joint distribution as well.

Definition 2.7. Two random variables X_1 and X_2 are called independent if and only if

$$P(X_1, X_2) = P(X_1)P(X_2), \tag{3}$$

i.e., if their joint distribution is the product of the individual probability distributions.

Here, the notation $P(X)$ is used as an abbreviation for $P(X = x)$ for all $x \in \Omega$. In this context, the individual distributions $P(X_1)$ and $P(X_2)$ are called the *marginal distribution* of X_1 and X_2, respectively. Besides the joint distribution, the distribution of a random variable X_1 *conditioned* on the outcome of another random variable X_2 can be defined as well.

Definition 2.8. Let (Ω, F, P) be a probability space, X_1, X_2 random variables defined on Ω, $x_1, x_2 \in F$ and $P(x_2) > 0$. The conditional probability of $X_1 = x_1$ given $X_2 = x_2$ is defined as

$$P(X_1 = x_1 | X_2 = x_2) = \frac{P(X_1 = x_1, X_2 = x_2)}{P(X_2 = x_2)}. \tag{4}$$

For conditional probabilities, also the short notations $P(X_1|X_2 = x_2), P(X_1 = x_1|X_2)$ and $P(X_1|X_2)$ are used throughout this work. One helpful lemma about conditional probabilities is the following *law of total probability.*

Lemma 2.1. Let X_1, X_2 be random variables defined on Ω and (x_i) a countable infinite partition of Ω. Then

$$P(X_1) = \sum_i P(X_1, X_2 = x_i) = \sum_i P(X_1|X_2 = x_i)P(X_2 = x_i). \tag{5}$$

Given the basic definitions from probability theory, stochastic processes can now be defined formally.

2.1.2 Stochastic Processes

Definitions of stochastic processes can also be found in common textbooks about probabilistic models and probability theory. One of these is "Introduction to Probability Models" by S. Ross [23], which the following definitions are based upon.

Definition 2.9. A *stochastic process* $\{X_t, t \in T\}$ is a collection of random variables, i.e., for each $t \in T$, X_t is a random variable. The set of possible values Ω that the random variables can assume is called the *state space* of the stochastic process.

Within this work, only the case of $T = \mathbb{Z}$ is considered. Such a stochastic process is called a discrete-time process and the values of the index set $t \in T$ are called time steps. Additionally, the state space is considered to be a finite alphabet A. Without loss of generality, it is assumed that the alphabet is a finite subset of the natural numbers, i.e., $A \subset \mathbb{N}_0$. Throughout this work, only a special class of stochastic processes are considered, namely *stationary stochastic processes.*

Definition 2.10. A stochastic process is called stationary or time-invariant if all random variables have the same joint distribution. That is, for any set of n index values $t_0, t_1, ...t_n \in \mathbb{Z}$ and any $\tau \in \mathbb{Z}$ it holds that

$$P(X_{t_0+\tau}, X_{t_1+\tau}, ...X_{t_n+\tau}) = P(X_{t_0}, X_{t_1}, ...X_{t_n}). \tag{6}$$

As a short-hand notation, we define the *future* and the *past* of a stochastic process.

Definition 2.11. Let $\{X_t, t \in \mathbb{Z}\}$ be a stationary stochastic process and $t_0 \in \mathbb{Z}$ be arbitrary. We define $\overleftarrow{X} = ..., X_{t_0-2}, X_{t_0-1}$ as the past and $\overrightarrow{X} = X_{t_0}, X_{t_1}, ...$ as the future of the stochastic process. Moreover, we define the sequences $X_{m:n} = X_m, X_{m+1}, ..., X_{n-1}$, $m < n$. Sequences in the state space $x_t \in A$ are defined analogously as $x_{m:n} = x_m, x_{m+1}, ..., x_{n-1}$ as well as $\overleftarrow{x} = x_{-\infty:t_0}$ and $\overrightarrow{x} = x_{t_0,\infty}$.

Note that for stationary stochastic processes, the future and the past are independent of any specific $t_0 \in \mathbb{Z}$. Stochastic processes can also be characterized by the "amount" of past information needed to accurately sample future time steps.

Definition 2.12. Let $\{X_t, t \in \mathbb{Z}\}$ be a stochastic process. The process is said to have Markov order $\kappa \in \mathbb{N}_0$ if

$$P(X_1|\overleftarrow{X}) = P(X_1|X_{-\kappa+1:1}), \tag{7}$$

i.e., if the future state depends only on the past κ states.

A special class of stochastic processes are *Markov chains.*

Definition 2.13. A stochastic process $\{X_t, t \in \mathbb{N}_0\}$ is called a Markov chain or described as Markovian, if it has Markov order $\kappa = 1$, i.e., if

$$P(X_1|\overleftarrow{X}) = P(X_1|X_0). \tag{8}$$

A stationary stochastic process has either a finite Markov order $\kappa \in \mathbb{N}_0$ or an unbounded Markov order $\kappa \to \infty$, which are then also called *non-Markovian.* For Markov chains, a central quantity is the *transition matrix*, that represents the transition probabilities of the process.

Definition 2.14. Let $\{X_t, t \in \mathbb{Z}\}$ be a Markov chain with alphabet A. The matrix $P \in \mathbb{R}^{|A|\times|A|}$ with elements $P_{i,j} = P(X_1 = x_j|X_0 = x_i)$ is called the transition matrix of the process. These elements describe the probability of the process to transition from state x_i to x_j within the next time step.

A specific past $\overleftarrow{x}$ together with a specific future $\overrightarrow{x}$ is called an instance of a stochastic process, which occurs according to the conditional probability $P(\overrightarrow{X} = \overrightarrow{x}|\overleftarrow{X} = \overleftarrow{x})$.
It can be confusing to deal with probability distributions $P(\overrightarrow{X}|\overleftarrow{X})$ with an uncountable set of random variables. Or in other words, how does a joint probability distribution $P(X_{-\infty,\infty})$ look and is it well defined? In practice, stochastic processes are modeled based on finite sets of random variables. This means, that one considers joint probability distributions $P(X_{-i,i})$ for arbitrary but fixed $i \in \mathbb{N}$. These can be grouped together into the family of finite probability distributions $\{P(X_{-i,i})\}_{i\in\mathbb{N}}$. The Extension Theorem of Kolmogorov [24] then guarantees that if this family fulfills certain requirements, there exists a stochastic process with probability space (Ω, F, P') such that all finite probability distributions coincide with P'.
Within this work, a learning algorithm is developed to train a model that can be used to simulate a stationary stochastic process. Here, simulating is meant as replicating the future behavior of a stochastic process, given some past observations. Formally, let $P(\cdot|\overleftarrow{x})$ be the conditional probability distribution of a stationary stochastic process for a specific past $\overleftarrow{x}$. A model is then initialized based on $\overleftarrow{x}$ and produces outcomes $x_1, x_2, \ldots$ according to a conditional probability distribution $\hat{P}(\cdot|\overleftarrow{x})$. We call an outcome of fixed length $L \in \mathbb{N}$ a future trajectory. How well a model can simulate the true process, i.e., how much the two distributions differ, is referred to as the performance of the model. Training a model is therefore understood as gradually fine-tuning the parameters of the model with the aim to increase its performance. A natural question is how the performance can be quantified. Mathematically, the difference between two probability distributions can be expressed with the use of a distance measure for probability distributions. Two commonly used measures for this task are introduced next.

2.1.3 Distance Measures for Probability Distributions

In general, we want to have a measure $\mathscr{D}(P, \hat{P})$ to be small, if the two distributions are *similar* in some sense. There are several ways to quantify the similarity of two probability distributions, where one way is based on so-called f-divergences.

Definition 2.15. Let $f : (0,\infty) \to \mathbb{R}$ be a convex function with $f(1) = 0$. Moreover, let P and $\hat{P}$ be two probability distributions with $supp(P) \subseteq supp(\hat{P})$, where $supp(P) = \{x \in \Omega : P(x) \neq 0\}$. The f-divergence of P from $\hat{P}$ is then defined as

$$\mathscr{D}_f(P, \hat{P}) = \sum_{x\in\Omega} \hat{P}(x) f\left(\frac{P(x)}{\hat{P}(x)}\right). \tag{9}$$

Different choices of the function f lead to different f-divergences. One commonly used divergence is the Kullback-Leibler (KL) divergence, which can be obtained by choosing $f(x) = x\log_2(x)$.

Definition 2.16. Let P and $\hat{P}$ be probability distributions with $supp(P) \subseteq supp(\hat{P})$. The Kullback-Leibler (KL) divergence is defined as

$$\mathscr{D}_{KL}(P,\hat{P}) = \sum_{x\in\Omega} P(x)\log_2\Big(\frac{P(x)}{\hat{P}(x)}\Big). \tag{10}$$

While the KL divergence is a distance measure for probability distributions, it is not a metric in the mathematical sense. This is because it is not symmetric and does not satisfy the triangle inequality. An f-divergence, which is also a metric, is the total variation (TV) distance and it can be obtained with $f(x) = \frac{1}{2}|x-1|$.

Definition 2.17. Let P and $\hat{P}$ be probability distributions. The total variation (TV) distance is defined as

$$\mathscr{D}_{TV}(P,\hat{P}) = \frac{1}{2}\sum_{x\in\Omega} |P(x)-\hat{P}(x)|. \tag{11}$$

This distance is a particularly strong metric insofar as it sums up all statistical errors without taking the relative occurrence of these errors into account. On the other hand, the KL divergence is also relatively strong since it upper bounds the TV distance, which is known as Pinsker's inequality.

Lemma 2.2. Let P and $\hat{P}$ be probability distributions with $supp(P) \subseteq supp(\hat{P})$. Then

$$\mathscr{D}_{TV}(P,\hat{P}) \leq \sqrt{\frac{1}{2}\mathscr{D}_{KL}(P,\hat{P})}. \tag{12}$$

Both distance measures compare probability distributions in some sense. However, in the domain of predictive modeling, we are dealing with conditional probability distributions and we are aiming to compare the distribution of a model with the distribution from the underlying stochastic process. Thus, we extend these measures within this work to also include the conditional behavior and to be applicable for stochastic processes. The extension is not straightforward and we therefore present it later in Section 3.2.
In this work we will focus on an algorithm whose training is performed with the use of quantum devices, i.e., a quantum learning algorithm. Therefore, the foundations of quantum computing are introduced next.

2.2 Quantum Computing

In the following paragraphs, the concepts of quantum computing that are important for the development of a quantum learning algorithm for predictive models are explained. Basic definitions and an introduction to quantum computation can be found in common textbooks such as "Quantum Computation and Quantum Information" by Nielsen and Chuang [25], which the following section is based on.

For classical computing, the fundamental unit of storing information is the *bit*, which is often physically realized by a silicon-based transistor that can be either in the 0 or 1 state. The analogous unit in the quantum computing realm is the *qubit* (quantum bit), which can be physically realized, e.g., by the spin of an electron. Here, the spin-up state of an electron can correspond to the classical 0 state and the spin-down state to the 1 state. Just as the spin of an electron can be in superposition, this also applies to the state of a qubit. Thus, a quantum state can be written as

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{13}$$

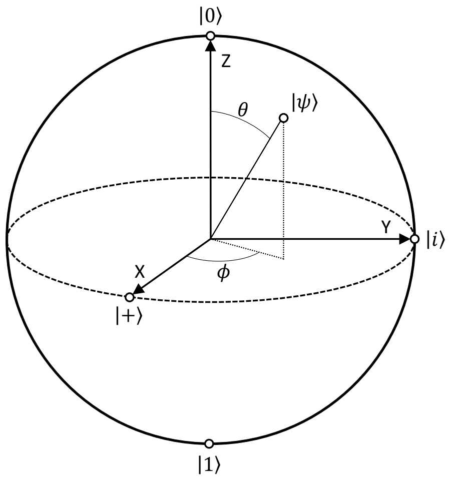

Figure 1: The Bloch sphere representation for an arbitrary qubit state $|\psi\rangle$ defined by the rotation angles θ and ϕ. The computational basis states $|0\rangle, |1\rangle$ as well as the special states $|+\rangle$ and $|i\rangle$ are illustrated. The figure is based on Figure 1.3 from Ref. [25].

with $\alpha, \beta \in \mathbb{C}$ and using the bracket notation $|\cdot\rangle$. This is yet not the general form of a qubit, since we also need to take measurement probabilities into account. When measuring a qubit, e.g., measuring the spin of an electron, the outcome is either 0 corresponding to a spin-up with probability $|\alpha|^2$, or 1 corresponding to a spin-down with probability $|\beta|^2$, and the state of the qubit collapses to either $|0\rangle$ or $|1\rangle$, respectively. While the qubit is defined by a quantum mechanical property like the spin, in quantum computing it is treated as an abstract mathematical object, hiding hardware-specific details, which allows to simply work with abstract mathematical spaces as the domain of computation. Thus, the state of a qubit is generally described by a unit vector $|\psi\rangle$ in a two-dimensional complex Hilbert space, denoted as H_2, i.e.,

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = \begin{bmatrix}\alpha\\ \beta\end{bmatrix} \in H_2 \quad \text{with} \quad |\alpha|^2 + |\beta|^2 = 1, \tag{14}$$

and the states $|0\rangle$ and $|1\rangle$ are called the *computational basis*. The normalization condition is introduced, such that the squared amplitudes $|\alpha|^2$ and $|\beta|^2$ can be interpreted as probabilities. Unless further specified, matrices and vectors are specified with respect to this basis.
Since there are infinitely many possible states of a qubit, an intuitive understanding can be difficult. One way of representing a qubit state geometrically is with the use of the *Bloch sphere*. This is a three-dimensional sphere, where each point on the surface represents one pure qubit state. Since such a state is a unit vector, it can be rewritten as

$$|\psi\rangle = \mathrm{e}^{i\gamma}\Big(\cos\frac{\theta}{2}\,|0\rangle + \mathrm{e}^{i\phi}\sin\frac{\theta}{2}\,|1\rangle\Big) = \cos\frac{\theta}{2}\,|0\rangle + \mathrm{e}^{i\phi}\sin\frac{\theta}{2}\,|1\rangle, \tag{15}$$

where the term $\mathrm{e}^{i\gamma}$ is called a global phase factor. Due to the nature of measurements in quantum mechanics, such a global phase factor is unobservable and hence can be omitted. The Bloch sphere for a qubit state $|\psi\rangle$ with the two rotation angles θ and ϕ is illustrated in Figure 1. Some special qubit states are

$$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \tag{16}$$

$$|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle), \tag{17}$$

$$|+i\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i\,|1\rangle), \tag{18}$$

$$|-i\rangle = \frac{1}{\sqrt{2}}(|0\rangle - i\,|1\rangle), \tag{19}$$

which are often referred to in the literature and thus have their own label.
Similar to classical computing, multiple qubits are grouped together to perform computations. Such groups are called *quantum registers* and are mathematically defined as the tensor product of the Hilbert spaces of the individual qubits, i.e.,

$$|\psi\rangle = \alpha\,|0\rangle \otimes |0\rangle + \beta\,|0\rangle \otimes |1\rangle + \gamma\,|1\rangle \otimes |0\rangle + \delta\,|1\rangle \otimes |1\rangle \in H_2 \otimes H_2, \tag{20}$$

for a general two-qubit quantum register $|\psi\rangle$. The tensor product symbol $\otimes$ between the states is often omitted, e.g. $|1\rangle\otimes|1\rangle = |11\rangle$, and the bit-string is interpreted as the binary representation of the corresponding decimal number, i.e., $|00\rangle = |0\rangle$, $|01\rangle = |1\rangle$, $|2\rangle = |10\rangle$ and $|3\rangle = |11\rangle$. This can be extended to a group of $n \in \mathbb{N}$ qubits, which is called an *n-qubit quantum register*. The state of such a register can be generally written as

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i\,|i\rangle \in H_2^{\otimes n}, \tag{21}$$

with $\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$. The dimension of the product Hilbert space is 2^n and thus grows exponentially with the number of qubits.
Fundamental for quantum computing is the concept of *entanglement*. The state of a quantum register is said to be entangled if it cannot be written as the tensor product of all individual qubits. This leads to a correlation of the measurement outcomes of an entangled quantum register. It can be seen, e.g., for the so-called GHZ state:

$$|\mathrm{GHZ}\rangle = \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle). \tag{22}$$

If the first (leftmost) qubit is measured to be in the $|0\rangle$ state, the entire quantum register collapses to the $|000\rangle$ state and futher measurements of the remaining qubits will always result in the $|0\rangle$ state as well.
The core idea of quantum computing is to apply operations onto quantum registers. Such operations can be divided into *gates* and *measurements*. These two types of quantum operations are briefly explained in the following sections, starting with the quantum measurement.

2.2.1 Quantum Measurements

Generally, a quantum measurement can be defined as follows.

Definition 2.18. A quantum measurement is defined as a set $\{M_i\}$ of measurement operators, acting on the state space of a quantum register and satisfying the *completeness equation*

$$\sum_i {M_i}^\dagger M_i = I. \tag{23}$$

The indices i are referring to the possible measurement outcomes and the probability to measure i for a quantum register being in state $|\psi\rangle$ is given by

$$P(i) = \langle\psi|{M_i}^\dagger M_i|\psi\rangle, \tag{24}$$

and the state after the measurement is given by

$$|\psi'\rangle = \frac{M_i\,|\psi\rangle}{\sqrt{\langle\psi|{M_i}^\dagger M_i|\psi\rangle}}. \tag{25}$$

In contrast to the general definition of a quantum measurement above, a special class of measurements are so-called *projective measurements* that are easier to work with.

Definition 2.19. A projective measurement is described by an *Observable*, i.e., an Hermitian operator $M : H_2^{\otimes n} \to H_2^{\otimes n}$, with spectral decomposition

$$M = \sum_i i P_i, \tag{26}$$

where i are the eigenvalues of M and P_i the projectors onto the corresponding eigenspace. Possible measurement outcomes of M are the eigenvalues i. The probability to measure i for a register being in state $|\psi\rangle$ is given by

$$P(i) = \langle\psi|P_i|\psi\rangle\,, \tag{27}$$

while the post-measurement state is

$$|\psi'\rangle = \frac{P_i\,|\psi\rangle}{\sqrt{P(i)}}. \tag{28}$$

One useful property of projective measurements is the ability to calculate expectation values, which is summarized in the following example.

Example 2.1. Let M be a projective measurement and $|\psi\rangle \in H_2^{\otimes n}$. The expectation value of M for a quantum state $|\psi\rangle$ is then given by

$$\mathbb{E}_{|\psi\rangle}(M) = \sum_i iP(i) = \sum_i i\,\langle\psi|P_i|\psi\rangle = \langle\psi|\sum_i iP_i|\psi\rangle = \langle\psi|M|\psi\rangle\,. \tag{29}$$

A special class of projective measurements is the *measurement in the computational basis.*

Definition 2.20. The measurement associated with $M : H_2^{\otimes n} \to H_2^{\otimes n}, M = \sum_{i=0}^{2^n-1} i\,|i\rangle\,\langle i|$ is called measurement in the computational basis.

The outcome of an n-qubit quantum register measured in the computational basis is a bit-string of length n. While measurements with respect to different bases exist, unless otherwise specified, throughout this work only measurements in the computational base are considered. In contrast to quantum measurements, quantum gates can be defined a bit more straight forward, which is done next.

2.2.2 Quantum Gates

Formally, a quantum gate is also an operator acting on the state space of a quantum register.

Definition 2.21. A quantum gate is a unitary operator $U : H_2^{\otimes n} \to H_2^{\otimes n}$ acting on the state space of an n-qubit quantum register. The application of a gate transforms the state according to the unitary, i.e., $U\,|\psi\rangle = |\phi\rangle$ with $|\psi\rangle\,, |\phi\rangle \in H_2^{\otimes n}$.

Multiple gates can be applied one after another and since each gate is unitary, all operations together form a unitary as well. Quantum gates acting only on a single qubit are called *single-qubit* gates. These gates can be described by a complex 2×2 matrix. If such a gate U is applied to one qubit of a quantum register, it means that the operator $I \otimes \cdots \otimes U \otimes \ldots I$ is applied to the entire register, where I denotes the identity operator on H_2. Important single-qubit gates are the Pauli gates X, Y and Z:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},\; Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix},\; Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{30}$$

The X gate acts on the states $|0\rangle$ and $|1\rangle$ like the classical "NOT" operation, while the Z gate flips the relative phase in the computational basis of a qubit in superposition. Y is a combination of both, the negation and a phase flip. Moreover, the Hadamard gate H, the phase gate S and the T gate play an important role in quantum computation:

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix},\ S = \begin{bmatrix}1 & 0\\ 0 & i\end{bmatrix},\ T = \begin{bmatrix}1 & 0\\ 0 & e^{i\frac{\pi}{4}}\end{bmatrix}. \tag{31}$$

The Hadmard gate H creates uniform superpositions when acting on the computational basis states, i.e.,

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle, \tag{32}$$
$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle. \tag{33}$$

Both the S and the T gate shift the relative phase of state by i and $e^{i\frac{\pi}{4}}$, respectively. Based on the Pauli gates, the (Pauli) *rotation gates RX, RY, RZ* are defined as

$$RX(\theta) = \exp\left(-i\frac{\theta}{2}X\right) = \begin{bmatrix}\cos\frac{\theta}{2} & -i\sin\frac{\theta}{2}\\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2}\end{bmatrix}, \tag{34}$$
$$RY(\theta) = \exp\left(-i\frac{\theta}{2}Y\right) = \begin{bmatrix}\cos\frac{\theta}{2} & -\sin\frac{\theta}{2}\\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2}\end{bmatrix}, \tag{35}$$
$$RZ(\theta) = \exp\left(-i\frac{\theta}{2}Z\right) = \begin{bmatrix}e^{-i\frac{\theta}{2}} & 0\\ 0 & e^{i\frac{\theta}{2}}\end{bmatrix}. \tag{36}$$

These gates perform a rotation of θ around the corresponding axis on the Bloch sphere, see Figure 1. The inverse of the rotation gates can be easily obtained via rotating in the opposite direction, e.g., $RX(\theta)\cdot RX(-\theta) = I$. Compared to the single-qubit gates X, Y, Z, H, S and T, the rotation gates offer a degree of freedom – the parameter θ. This degree of freedom allows to fine-tune a quantum operation, and it is these parameters in particular that are trained as part of a variational quantum circuit, as described later in Section 2.2.4.
Quantum gates can also be defined on multiple qubits, which are then called *multi-qubit* gates. One important two-qubit gate is the "Controlled-NOT" (*CNOT*) gate

$$CNOT = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{bmatrix}, \tag{37}$$

which flips the second qubit if the first qubit is in state $|1\rangle$, e.g., $CNOT|10\rangle = |11\rangle$. Therefore, the first (left) qubit is called the *control* and the second (right) qubit the *target* qubit. The *CNOT* gate is also called *CX* gate, since it can be written as

$$CNOT = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X, \tag{38}$$

which is interpreted as applying the X gate to the target qubit if the control qubit is in state $|1\rangle$, i.e., a controlled-X operation. Similarly, controlled-U operations for arbitrary single-qubit gates U can be defined via the block-matrix

$$CU = \begin{bmatrix}I & 0\\ 0 & U\end{bmatrix}. \tag{39}$$

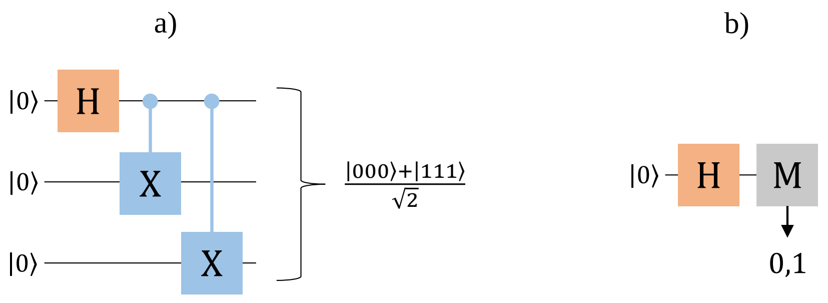

Figure 2: a) A quantum circuit for preparing the GHZ state. The circuit has three wires, corresponding to three qubits, together with three gates, the Hadamard gate H and two CX gates. b) A one-qubit quantum circuit with Hadamard gate and measurement operation. The output is either 0 or 1 with equal probability.

Quantum gates can also be controlled by qubits being in the $|0\rangle$ state. We denote these gates as $\widetilde{CU}$ and call them $|0\rangle$-controlled gates. They can be generally written as

$$\widetilde{CU} = |0\rangle\langle 0| \otimes U + |1\rangle\langle 1| \otimes I. \tag{40}$$

Another widely used two-qubit gate is the *SWAP* gate

$$SWAP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{41}$$

which simply swaps the states of the two qubits, i.e., $SWAP|10\rangle = |01\rangle$.

For classical computing, a small set of logical operations can be used to compute any classical function [25]. These sets are called *universal*. In the quantum computing realm, a similar result for universality exists. It is shown that one- and two-qubit gates can be used to construct unitary operators of arbitrary dimension [25]. Moreover, single-qubit gates together with, e.g., the *CNOT* gate, can be used to construct any two-qubit gate [25]. Thus, any unitary operator acting on n qubits can be implemented using single-qubit and *CNOT* gates. However, while this construction is exact, it is also highly inefficient since it requires $O(n^2 4^n)$ single qubit and *CNOT* gates [25]. Therefore, *universality* in the quantum computing realm is defined slightly differently: A set of quantum gates is said to be universal if any unitary operation can be approximated to arbitrary accuracy by a sequence of gates from this set [25]. It is shown that the set $\{H, S, T, CNOT\}$ is universal. To be more precise, any unitary U containing m single-qubit and *CNOT* gates can be approximated up to accuracy $\epsilon > 0$ with $O(m \cdot \log^c(m/\epsilon))$ gates from this set [25]. Here, the accuracy is with respect to an operator norm and $c \approx 2$ is a constant.

2.2.3 Quantum Circuits

The sequences of gates and measurements acting on qubits can be graphically represented with *quantum circuits*. Here, each wire corresponds to a qubit and the quantum gates are represented as rectangles. The time goes from left to right and the initial states of the qubits are written on the left-hand side of each wire. If a rectangle is connected to another wire with a dot, it means that the corresponding gate is controlled by the connected qubit. An example quantum circuit that prepares the GHZ state from Equation (22) is presented in Figure 2 a). Here, one Hadamard gate is applied on the first qubit followed by two *CX* gates acting on the second and third qubit, controlled by the first one.

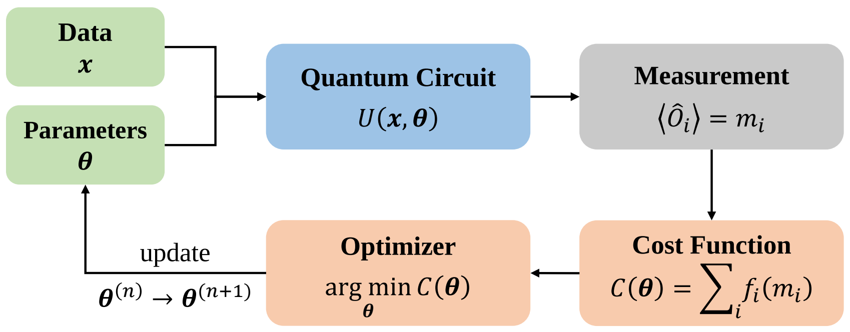
Data
$\boldsymbol{x}$
Parameters
$\boldsymbol{\theta}$
Quantum Circuit
$U(\boldsymbol{x},\boldsymbol{\theta})$
Measurement
$\langle \hat{O}_i \rangle = m_i$
Cost Function
$C(\boldsymbol{\theta}) = \sum_i f_i(m_i)$
Optimizer
$\arg\min_{\boldsymbol{\theta}} C(\boldsymbol{\theta})$
update
$\boldsymbol{\theta}^{(n)} \to \boldsymbol{\theta}^{(n+1)}$

Figure 3: A schematic illustration of a variational quantum algorithm. The quantum circuit $U(\boldsymbol{x},\boldsymbol{\theta})$ is initialized with training data $\boldsymbol{x}$ and free parameters $\boldsymbol{\theta}$. After the execution, some observables $\hat{O}_i$ are measured and a cost function $C(\boldsymbol{\theta})$ is calculated based on the measurement outputs m_i. A classical optimizer uses the cost function and updates the parameters, such that another iteration can be performed with the goal to minimize the cost.

A quantum measurement is often denoted as a rectangle including a "meter" symbol. Another way to illustrate measurements is to simply let the qubit wire end after the rectangle and write down the measurement outcome, see Figure 2 b) for an example circuit that measures a qubit in the $H|0\rangle = |+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ state. Here, the measurement outcome is either 0 or 1 with equal probability.
Two important properties of a quantum circuit are its *depth* and its *width*. The depth of a circuit can be obtained by simply moving all gates in the circuit to the left without changing their order for acting on the same qubits. Then, the number of gates for the qubit with the largest number of gates involved is defined as the depth of the circuit. In contrast, the width is simply the number of qubits that are manipulated by the quantum gates from the circuit.
The field of quantum computing is currently entering the area of *noisy intermediate-scale quantum computing* (NISQ) [26]. This name was introduced by J. Preskill in 2018 and the term "intermediate-scale" refers to quantum devices with less than a few hundred qubits, while "noisy" emphasizes that the physically implemented gates offer only imperfect control over the qubits [26]. Therefore, quantum circuits must have a low depth in order to be executed with reasonable accuracy on current quantum devices. One category of quantum algorithms that take these considerations into account are variational quantum algorithms (VQAs), which are introduced next.

2.2.4 Variational Quantum Algorithms

The core idea of a VQA lies in the fact that quantum circuits can offer free parameters with the use of rotation gates, which can be utilized in an optimization procedure. These quantum circuits are called *parameterized quantum circuits* (PQCs), which emphasizes the parameterization of the circuit. Often, PQCs are also referred to as *variational quantum circuits* (VQCs) to highlight that the parameters are varied iteratively. VQAs are hybrid quantum-classical algorithms, where the classical part is taken by a classical optimization algorithm. The execution of the quantum circuit is thus a quantum subroutine within a classical optimization procedure. Variational quantum circuits have been proposed for many fields and tasks such as approximating the solution of combinatorial optimization problems [27], calculating the energy ground states of molecules [13] and machine learning [28].
The general structure of a VQA is illustrated in Figure 3. A quantum circuit, denoted as a unitary U, is initialized with some training data $\boldsymbol{x} \in \mathbb{R}^d$ and an initial set of parameters $\boldsymbol{\theta}^{(0)} \in \mathbb{R}^n$. The circuit is then executed multiple times and some observables $\hat{O}_i$ are measured to approx-

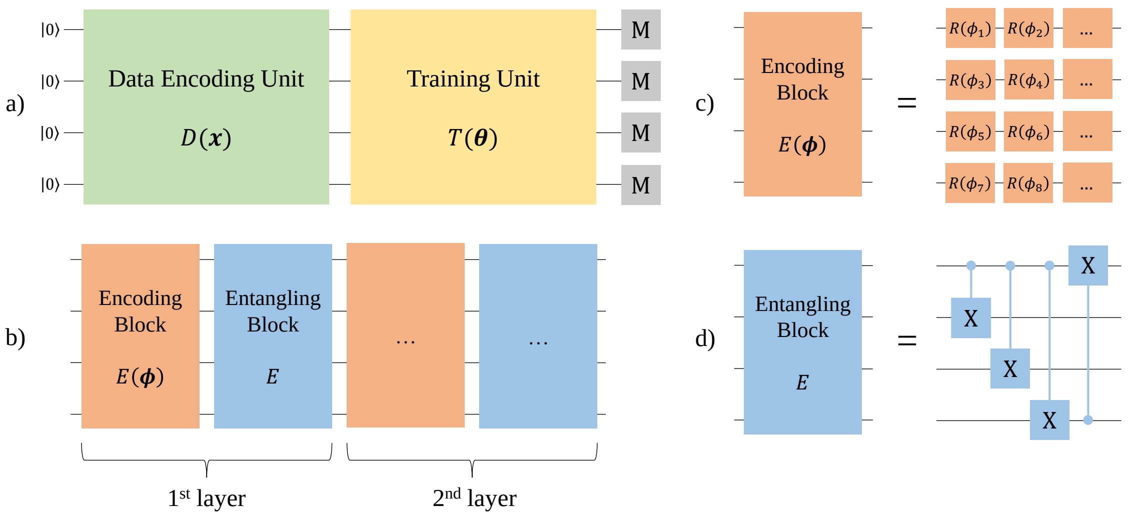

Figure 4: Illustration of a problem-inspired ansatz for a VQC. a) The circuit is made up by two units, one data encoding unit $D(\boldsymbol{x})$ and one training unit $T(\boldsymbol{\theta})$. b) Such units contain several layers of alternating encoding and entangling blocks. c) One encoding block $E(\boldsymbol{\phi})$ applies a fixed number of Pauli rotation gates $R \in \{RX, RY, RZ\}$. d) The entangling block E applies CX gates between each neighboring qubits.

imate the expectation values $\langle \hat{O}_i \rangle = m_i$. This information is used to construct a cost function, which is mostly of the form of $C(\boldsymbol{\theta}) = \sum_i f_i(m_i)$, where f_i can be some classical post-processing of the measurement outcome. A classical optimizer is then used to update the parameters, $\boldsymbol{\theta}^{(n)} \rightarrow \boldsymbol{\theta}^{(n+1)}$, with the goal to minimize the cost. The updated parameters are used to perform the next iteration until a convergence criterion is satisfied or until a maximal number of iterations is reached.
The choice of the observables $\hat{O}_i$ depends on the cost function that is used. The structure of U, i.e., the choice of gates and their arrangement, is called the *ansatz*. Different ansätze have been proposed in the literature, some of which are tailored to specific optimization problems and thus are called *problem-inspired* ansätze. On the other hand, *problem-agnostic* ansätze have been proposed to offer generic quantum circuits that can be used when no relevant information about the problem is available. One example of an ansatz for a VQC consisting of four qubits is shown in Figure 4. This ansatz is a problem-inspired ansatz and has been used in the field of supervised machine learning [11, 17, 18]. Here, the VQC is split into two units, one data encoding unit $D(\boldsymbol{x})$ that encodes some data $\boldsymbol{x}$, and one training unit $T(\boldsymbol{\theta})$ with the free parameters $\boldsymbol{\theta}$, see Figure 4 a). Such a unit is build up by $l \in \mathbb{N}$ layers, where each layer is a combination of an encoding block $E(\boldsymbol{\phi})$ and an entangling block E. The encoding block contains a fixed number of Pauli rotation gates $R \in \{RX, RY, RZ\}$, whereas the entangling block connects each qubit with its neighboring qubit via a CX gate, see Figure 4 d). A VQC following this ansatz can be written as a unitary $U(\boldsymbol{x}, \boldsymbol{\theta})$ and contains only Pauli rotation gates and CX gates.
To formally work with a VQC, it can be understood as a function, mapping the parameters to an expectation value.

Definition 2.22. Given an observable $\hat{O}$, a VQC is a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$, mapping $n \in \mathbb{N}$ real-valued parameters to the expectation value of the observable, i.e.,

$$f(\boldsymbol{\theta}) = \langle \hat{O} \rangle = \langle 0 | U(\boldsymbol{\theta})^\dagger \hat{O} U(\boldsymbol{\theta}) | 0 \rangle \,, \tag{42}$$

with $U(\boldsymbol{\theta})$ being the unitary representing the VQC.

It is important to note that in practice, a quantum circuit can only be executed finitely many times and the expectation value $\langle \hat{O} \rangle$ is thus approximated.

Within this work, a quantum circuit will serve as a quantum predictive model, that samples future trajectories of a stochastic process. The measurement outcomes of such a circuit are bit-strings, i.e., a collection of zeros and ones, and will be interpreted as the future samples. Therefore, the probability to measure qubits in a certain state is of interest. Based on Definition 2.22, these probabilities can be obtained as follows:

Example 2.2. Consider the observable $\hat{O}_i = |i\rangle\langle i|$ for a fixed $|i\rangle$, i.e., the projector onto one state of the computational basis $|i\rangle \in \{|0\rangle, |1\rangle\}^{\otimes d}$ built up by d qubits. Applying Definition 2.22 then yields

$$f_i(\boldsymbol{\theta}) = \langle \hat{O}_i \rangle = \langle 0|U(\boldsymbol{\theta})^\dagger |i\rangle\langle i| U(\boldsymbol{\theta})|0\rangle = \langle \psi(\boldsymbol{\theta})|i\rangle\langle i|\psi(\boldsymbol{\theta})\rangle = |\langle \psi(\boldsymbol{\theta})|i\rangle|^2, \tag{43}$$

with $|\psi(\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})|0\rangle$ being the state prepared by the circuit $U(\boldsymbol{\theta})$. $|\langle\psi(\boldsymbol{\theta})|i\rangle|^2$ represents the probability to measure $i \in \{0,1\}^d$ in the computational basis. Therefore, one can simply write

$$f_i(\boldsymbol{\theta}) = P_{\boldsymbol{\theta}}(i). \tag{44}$$

The above example shows that executing and measuring the VQC many times results in an approximation of the probability distribution of the VQC. Within this work, we propose a VQC that uses two quantum registers A and B, where we are only interested in the probability distribution of measuring B. For such a VQC, a similar result holds.

Example 2.3. Consider a VQC acting on the quantum registers $A \otimes B$, $A = H_2^{\otimes N_A}$, $B = H_2^{\otimes N_B}$, i.e., the first register consists of $N_A \in \mathbb{N}$ and the second of $N_B \in \mathbb{N}$ qubits. Next, define the observable $\hat{O}_i = \sum_{j=0}^{2^{N_A}-1} |ji\rangle\langle ji|$ for fixed $|i\rangle \in \{|0\rangle, |1\rangle\}^{\otimes N_B}$. Applying Definition 2.22 then yields

$$\begin{aligned}
f_i(\boldsymbol{\theta}) &= \langle \hat{O}_i \rangle & (45)\\
&= \langle 00|U(\boldsymbol{\theta})^\dagger \sum_{j=0}^{2^{N_A}-1} |ji\rangle\langle ji| U(\boldsymbol{\theta})|00\rangle & (46)\\
&= \sum_{j=0}^{2^{N_A}-1} \langle 00|U(\boldsymbol{\theta})^\dagger|ji\rangle\langle ji|U(\boldsymbol{\theta})|00\rangle & (47)\\
&= \sum_{j=0}^{2^{N_A}-1} \langle \psi(\boldsymbol{\theta})|ji\rangle\langle ji|\psi(\boldsymbol{\theta})\rangle & (48)\\
&= \sum_{j=0}^{2^{N_A}-1} |\langle \psi(\boldsymbol{\theta})|ji\rangle|^2 & (49)\\
&= \sum_{j=0}^{2^{N_A}-1} P_{\boldsymbol{\theta}}(j,i) & (50)\\
&= P_{\boldsymbol{\theta}}(i), & (51)
\end{aligned}$$

with $|\psi(\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})|00\rangle$. $P_{\boldsymbol{\theta}}(i,j)$ is the probability to measure register A in j and register B in i, while $P_{\boldsymbol{\theta}}(i)$ is the probability to measure register B in i, regardless of the measurement outcome of register A.

The success of a VQC depends on the efficiency of the optimization procedure, and the design of the ansatz and the cost function. In principle, any classical optimizer can be used for the hybrid quantum-classical loop. However, specifically *gradient-based* optimizers have been proposed in the literature, since they are known to be robust against sampling noise [18, 29]. Therefore, the gradient of the cost function with respect to the free parameters $\boldsymbol{\theta}$ needs to be calculated. Thus, we need to calculate the partial derivatives $\partial_{\theta_i} f(\boldsymbol{\theta})$. While several ways exist to calculate these expressions, one method has shown promise, known as the *parameter-shift* *rule* [21]. Here, the derivative of the expectation value f can be obtained via calculating expectation values of a “similar” circuit. To be more precise, consider a VQC as shown in Figure 4 with only *CX* and Pauli rotation gates. Moreover, assume that the parameter θ_i is affected only by one Pauli rotation gate $R(\theta_i)$. The partial derivative of f with respect to θ_i is then given by

$$\partial_{\theta_i} f(\boldsymbol{\theta}) = \frac{1}{2}\left[f(\boldsymbol{\theta} + \frac{\pi}{2}\cdot \boldsymbol{e}_i) - f(\boldsymbol{\theta} - \frac{\pi}{2}\cdot \boldsymbol{e}_i)\right], \tag{52}$$

with $\boldsymbol{e}_i$ being the i-th unit vector in $\mathbb{R}^n$ [21]. Therefore, the gradient can be obtained by evaluating the same circuit twice, shifting the parameter θ_i $\pm\pi/2$, and aggregating the values on a classical computer. Note that while the expression for the gradient is exact, the expectation values are approximated based on finite measurements.

VQCs have a number of useful properties in conjunction with NISQ devices. On the one hand, it is shown that there exist classes of low-depth VQCs that can produce highly non-trivial outputs. One example are so-called *instantaneous quantum polynomial* (IQP) circuits that produce probability distributions which cannot be classically simulated efficiently [30]. On the other hand, most ansätze lead to scalable circuits in depth and width, such that VQCs can scale up with future improvements of the devices. While a VQC can only offer an estimation of the cost function and its gradient, it is shown that these estimators together with a gradient-based optimizer essentially implement a stochastic gradient descent optimization scheme, which offers fast convergence rates in many settings [31]. Since the training is performed directly on the imperfect hardware, errors are implicitly taken into account within the optimization procedure.

Using quantum resources for predictive modeling has already been studied in the literature [11, 12, 32]. However, before listing these findings, predictive models are first formally introduced in the next section.

2.3 Predictive Models

In general, the goal of predictive models can be formulated as follows. Let $\{X_t, t \in \mathbb{Z}\}$ be a stationary stochastic process. Based on a given past $\overleftarrow{x}$, the predictive model samples a future trajectory $\overrightarrow{x}$ with the use of a physical memory. To this end, the memory stores suitable configurations such that after emitting one time step x_1, the state is updated based on x_1, allowing the model to generate the next time step x_2. This process can be applied sequentially to produce arbitrarily long future trajectories $x_1, x_2, x_3, \ldots$ [11]. Within this work, a distinction of predictive models is made based on their accuracy, i.e., the aim of *exact* or *approximate* modeling, as well as based on the technology used (classical or quantum). This distinction is illustrated in Figure 5. Exact predictive models produce trajectories governed by the same conditional probability distribution $P(\overrightarrow{X}|\overleftarrow{X})$ as the underlying stationary stochastic process. By contrast, approximate predictive models produce trajectories based on an approximation $\hat{P}(\overrightarrow{X}|\overleftarrow{X})$ of the distribution of the process. Classical predictive models use only classical resources for information processing, whereas quantum models can use quantum and classical resources. Approximate models can arise in different situations, such as when a model is learned based on a finite sample set or when the memory, either classical or quantum, is constrained. The “difference” between the approximate distribution $\hat{P}$ and the true distribution P is called *distortion* and reflects statistical errors in sampling the trajectories. Distance measures for probability distributions such as the KL divergence can be used to quantify these errors.

The use of a quantum resources for information processing can lead to an advantage in either the accuracy for approximate models [11] or lower memory requirements for exact models [12]. How this advantage is quantified and in which situations it occurs is investigated in detail in the following sections. The focus of this comparison lies on the different technologies, i.e., classical or quantum resources, while the comparison of exact and approximate models is not considered within this work.

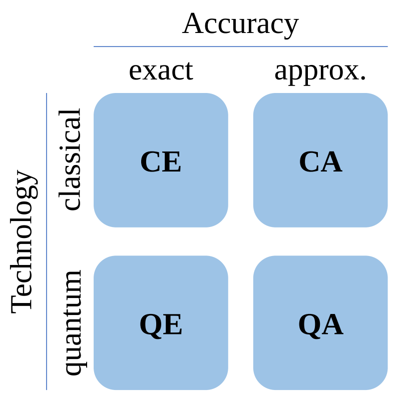

Figure 5: Illustration of the distinction of predictive models based on the accuracy (exact vs. approximate) and the technology (classical vs. quantum).

2.3.1 Exact Models

Before a comparison can be done, the predictive models are formally defined first, starting with an exact predictive model.

Definition 2.23. Let $\{X_t, t \in \mathbb{Z}\}$ be a stationary stochastic process. A classical *exact predictive model* for $\{X_t\}$ is a 3-tuple (E, M_c, P), where E is a deterministic map that encodes each past $\overleftarrow{x}$ to a suitable state $E(\overleftarrow{x})$ within a classical physical memory M_c, such that the same systematic action P on M_c at each subsequent time step sequentially outputs $x_1, x_2, x_3, \dots$ with probability $P(\overrightarrow{X} = \overrightarrow{x} | \overleftarrow{x})$.

For such models, knowing the state $E(\overleftarrow{x})$ of the memory M_c is as useful as knowing the particular past $\overleftarrow{x}$ for the purpose of generating future samples. Therefore, we write $P(\overrightarrow{X} = \overrightarrow{x} | E(\overleftarrow{x}))$ and read it as the probability of generating the future $\overrightarrow{x}$ given the memory M_c is in state $E(\overleftarrow{x})$. One important property of predictive models is that they are *unifilar*, which means that if a predictive model is in state $E(\overleftarrow{x})$ and emits some output x_1, the model will transition with certainty to the state $E(\overleftarrow{x}, x_1)$. It is this property in particular that allows predictive models to sample faithful future trajectories of arbitrary length. All the states $\{E(\overleftarrow{x})\}$ can be collected and indexed with some $i \in \mathbb{N}$, referring to the i-th state, which is then denoted as S_i.
One example of a classical exact predictive model is the so-called ϵ-machine, which has been studied in diverse contexts such as spike trains [3], stock markets [4, 5], and complex systems [6]. These models exploit the fact that if two pasts $\overleftarrow{x}$ and $\overleftarrow{x}'$ offer the same future statistics, they can be grouped together to generate future samples. This is formally done via dividing all possible pasts into equivalence classes according to

$$\overleftarrow{x} \sim \overleftarrow{x}' \text{ iff } P(\overrightarrow{X} | \overleftarrow{x}) = P(\overrightarrow{X} | \overleftarrow{x}'). \tag{53}$$

For each class, the ϵ-machine allocates one state $E(\overleftarrow{x}) = S_i$ in the memory. These states are called *causal states* and it has been shown that the ϵ-machine is the classical exact predictive model with provable minimal memory dimension [1]. Moreover, the causal states are a unique property of the process itself [1]. Therefore, properties of the ϵ-machine can be translated to properties of the underlying stationary stochastic process. Two important properties are the *topological* and the *statistical complexity*.

Definition 2.24. Let $\{X_t\}$ be a stationary stochastic process. Moreover, let $\{S_i\}_{i=1}^n$ be the $n \in \mathbb{N}$ causal states of $\{X_t\}$. The classical topological complexity of the process is defined as

$$d_c = \log_2 n. \tag{54}$$

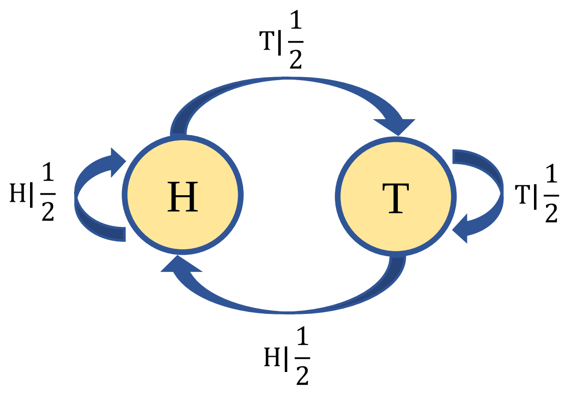

Figure 6: Graphical representation of the ϵ-machine of a fair coin toss process. The circles denote the causal states and the arrows the possible transitions between the states. Each label $x|P(x)$ indicates a transition with probability $P(x)$ while emitting x.

Definition 2.25. Let $\{X_t\}$ be a stationary stochastic process. The classical statistical complexity of the process is defined as

$$C_c = -\sum_i \pi_i \log_2 \pi_i, \tag{55}$$

with π_i being the stationary probability distribution of the i-th causal state.

The stationary distributions are understood as the asymptotic probability to be in a certain causal state for many i.i.d. simulations, i.e., $\pi_i = P(S_i)$. Similarly, the statistical complexity can be seen as the mean asymptotic memory requirement, measured in bits, of many parallel and i.i.d. processes [32]. By contrast, the topological complexity is the minimal memory requirement in bits that is needed to store all the causal states for a single instance of the process.
ϵ-machines can often be calculated analytically, given that a mathematical description of the underlying process is available. Moreover, methods have been proposed to reconstruct an ϵ-machine from empirical estimates of the joint distribution $P(\overleftarrow{X}, \overrightarrow{X})$ [8, 9]. Another way to construct an ϵ-machine directly from data is proposed by Shalizi et al. in Ref. [10] and named *causal state splitting reconstruction*. The core idea of this algorithm is to assume that one causal sate is already sufficient, but new states are added if necessary. This is the case if the model defined by the current causal states is not Markovian. Once the states define a Markov model, the algorithm is finished and the model is unifilar by construction. Whether or not the model is Markovian is determined via hypothesis testing.
An ϵ-machine offers a natural graphical representation. Figure 6 shows the ϵ-machine of a fair coin toss. In each step of the process, a fair coin is tossed with probability 1/2. The ϵ-machine thus has two causal states H, T with equal transition probabilities 1/2. Causal states are represented by circles and transitions by arrows labeled with their transition probability $P(x)$ and the emitting output symbol x.
The graphical representation of ϵ-machines is very similar to the notation of hidden Markov models (HMM). These models are widely used in the area of probabilistic modeling, and thus their relation to each other is briefly pointed out. Formally, an HMM is a pair of stochastic processes $(\{X_t\}, \{Y_t\})$ with $\{X_t\}$ being a Markov chain and $P(Y_t|\overleftarrow{X}) = P(Y_t|X_t)$. The states X_t are not directly observable, i.e., hidden, and the second process $\{Y_t\}$ is used to model the emissions of the system, whose probabilities are only dependent on the current hidden state X_t. From this perspective, an ϵ-machine can be seen as a unifilar hidden Markov model [1]. This is because due to the unifilarity of the model, one does not need to distinguish between emission and transition probabilities.
Within this work, we are interested in utilizing quantum devices for predictive modeling and thus define a *quantum* exact predictive model.

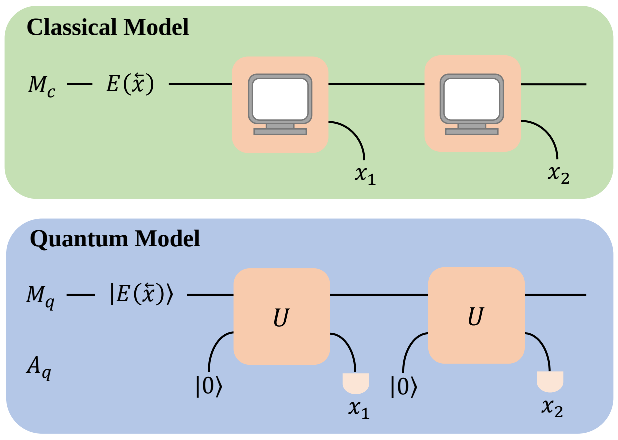

Figure 7: An illustration of the comparison between classical and quantum predictive models. Classical models start from a classical state $E(\overleftarrow{x})$ and act on a classical memory M_c. Quantum models start from a quantum state $|E(\overleftarrow{x})\rangle$, acting on both a memory and auxiliary quantum register and a measurement of the latter one generating future samples $x_1, x_2, ...$ The figure is based on Figure 2 from Ref. [11].

Definition 2.26. Let $\{X_t\}$ be a stationary stochastic process. A quantum exact predictive model for $\{X_t\}$ is a 4-tuple (E, M_q, A_q, U). E is a deterministic map that encodes each past $\overleftarrow{x}$ to a quantum state $|E(\overleftarrow{x})\rangle$ within a quantum register M_q. A_q is an ancillary quantum register that is initialized to the same fixed quantum state $|p\rangle$ before each time step. U is a time step independent unitary that acts sequentially on $M_q \otimes A_q$, such that at each subsequent time step, a measurement of A_q outputs $x_0, x_1, x_2, ...$ with probability $P(\overrightarrow{X} = \overrightarrow{x} | \overleftarrow{x})$.

For convenience, we set $|p\rangle = |0\rangle$, but it should be noted that the fixed quantum state can be arbitrarily complex, as long as this is respected in the structure of the unitary U. Just like classical predictive models, quantum predictive models are also unifilar. A comparison of classical and quantum predictive models is illustrated in Figure 7. The classical models store their causal states in their classical memory. By contrast, quantum models use a quantum register M_q for this purpose but also have another register A_q that is measured to generate future samples.
The quantum analogue to the ϵ-machine was theoretically introduced by Gu et al. [33]. Later, Binder et al. named these models q-simulators and showed a practical way to construct these models on a classical computer, given as input the ϵ-machine of a stationary stochastic process [12]. A q-simulator is a quantum exact predictive model in the sense of Definition 2.26 and features a set of internal states $|\sigma_i\rangle$, each corresponding to one causal state S_i from the stochastic process. The core of a q-simulator is the unitary U, which acts on $M_q \otimes A_q$ according to

$$|1_i\rangle = U|\sigma_i\rangle|0\rangle = \sum_{x_1 \in A} \sqrt{P(x_1|S_i)}\,|\sigma_{\lambda(S_i,x_1)}\rangle|x_1\rangle, \tag{56}$$

with A being the alphabet of the stochastic process, $P(x_1|S_i)$ the probability to emit x_1 when being in state S_i and $\lambda(S_i, x_1)$ the transition function that outputs the next causal state when emitting x_1 from S_i. Equation (56) shows the application of U for one single time step and denotes the resulting state with $|1_i\rangle$. This can be written in a more general way for the case of applying U L times, $L \in \mathbb{N}$, onto the memory and L auxiliary registers, i.e.,

$$|L_i\rangle = U^L|\sigma_i\rangle|0\rangle^{\otimes L} = \sum_{x_{1:L} \in A^L} \sqrt{P(x_{1:L}|S_i)}\,|\sigma_{\lambda(S_i,x_{1:L})}\rangle|x_{1:L}\rangle, \tag{57}$$

with defining λ recursively as $\lambda(S_i, x_{1:L}) = \lambda(\lambda(S_i, x_{1:L-1}), x_L)$. From here, it can be seen that after preparing the state $|L_i\rangle$ one measures a length-L sequence $x_{1:L} \in A^L$ with probability

$P(x_{1:L}|S_i)$, while the memory register collapses to the state $|\sigma_{\lambda(S_i,x_{1:L})}\rangle$. Binder et al. showed that such a unitary exists and that it can be calculated classically based on the Gram-Schmidt procedure [12].
The optimality of the ϵ-machine led to the definition of the classical topological and statistical complexity and we would like to define its quantum counterparts for the purpose of comparison. While it has been shown that the q-simulator is the optimal quantum predictive model for some classes of stochastic processes [7, 32], it is not known whether this also holds in general for arbitrary stationary stochastic processes. As a result, we cannot derive the quantum analogues of the classical complexities from the q-simulator. We can, nevertheless, define these measures as the minimal complexity over all valid models.

Definition 2.27. Let $\{X_t\}$ be a stationary stochastic process. The *quantum topological complexity* of $\{X_t\}$ is defined as

$$d_q = \min_{M_q}\Big(\log_2\big(\dim M_q\big)\Big), \tag{58}$$

where the minimization is done over all valid quantum exact predictive models.

Definition 2.28. Let $\{X_t\}$ be a stationary stochastic process. The *quantum statistical complexity* of $\{X_t\}$ is defined as

$$C_q = \min_{\rho}\Big(-\operatorname{tr}\big(\rho\log_2\rho\big)\Big), \tag{59}$$

with $\rho = \sum_i p_i |\sigma_i\rangle\langle\sigma_i|$ being the quantum stationary state and the minimization is performed over all valid quantum exact predictive models.

With respect to these measures, an advantage for quantum predictive models over classical ones has been shown, which is summarized in the following paragraphs.

Gu et al. [33] showed that there exist stationary stochastic processes that have a strictly lower quantum statistical complexity, i.e.,

$$C_q < C_c. \tag{60}$$

In particular, this holds true for all stationary stochastic processes that have the so-called *irreversibility condition*. That is, given the ϵ-machine of the stochastic process, the existence of two causal states S_i, S_j and a non-zero probability to transition from these causal states to another state S_k while emitting the same output $x \in A$. This means that given the last output x and the current causal state S_k, one is not able to determine the previous causal state with confidence. One example of such a stochastic process is the *perturbed coin*, where at each time step a box including the coin is perturbed, such that the coin flips with probability $0 < p < 1$ and the state of the coin is observed [33].
A similar result was shown by Thompson et al. [7] for the quantum topological complexity. Here, d_q was analyzed for stochastic processes together with their time-inverted counterparts, i.e., the stochastic processes that run from the future to the past. Let $d_c^+, d_c^-, d_q^+, d_q^-$ be the classical/quantum topological complexity for a stochastic process (+) and the time-inverted counterparts (−), respectively. It was shown that

$$\max(d_q^+, d_q^-) \leq \min(d_c^+, d_c^-), \tag{61}$$

holds for any stationary stochastic process. This means in particular that the quantum topological complexity never exceeds the classical one, or in other words, a quantum model has at worst the same topological complexity as its classical counterpart. Moreover, Thompson et al. showed that

$$d_q < d_c \tag{62}$$

holds true for a concrete example, the *heralding coin*, which is a simple extension to the perturbed coin with some post-processing of the observed states. Additionally, it was shown that there exists a family of processes, parameterized by some $n \in \mathbb{N}$, such that the advantage is unbounded, i.e.,

$$|C_q(n) - C_c(n)| \to \infty \text{ and } |d_q(n) - d_c(n)| \text{ for } n \to \infty. \tag{63}$$

The findings above indicate that quantum predictive models can have an advantage over classical ones in terms of the topological and statistical complexity. d_c and d_q are defined for stochastic processes and reflect the minimum dimension of the memory for any predictive model in order to *faithfully* sample future trajectories, i.e., without distortion. If processes become more and more non-Markovian, the amount of past information that is needed for exact simulation grows and thus the topological complexity grows as well [2]. Therefore, finite memory of classical/quantum predictive models necessarily lead to distortions. In this case, predictive models can only approximate the true stochastic behavior and thus are called approximate predictive models.

2.3.2 Approximate Models

Approximate predictive models can be defined analogously to Definition 2.23, respectively Definition 2.26 for the quantum case. The only difference is that the model outputs future samples $x_0, x_1, \ldots$ according to the distribution $\hat{P}(\overrightarrow{X} = \overrightarrow{x} | \overleftarrow{x})$, which is an approximation to the true probability distribution of the underlying stationary stochastic process. Such models arise in the setting when predictive models are learned based on a finite sample set or when imperfect hardware is used, such as NISQ devices. Moreover, one defines the following two properties of approximate predictive models.

Definition 2.29. Let (E, M_c, P) be an approximate classical predictive model with causal states $\{S_i\}_{i=1}^n$, $n \in \mathbb{N}$. The *classical memory size* of the model is defined as

$$\hat{d}_c = \log_2 n. \tag{64}$$

Definition 2.30. Let (E, M_q, A_q, U) be an approximate quantum predictive model. The *quantum memory size* of the model is defined as

$$\hat{d}_q = \log_2 \dim M_q. \tag{65}$$

Note that unlike the causal states of exact models, the states of approximate models are not called causal, but simply memory states or configurations of the memory. Also note that even if predictive models satisfy $\hat{d}_c = d_c$ or $\hat{d}_q = d_q$, they can be approximate. This can occur in situations when a model is learned based on a finite sample set or when imperfect hardware is used, such as NISQ devices.

Yang et al. [11] showed that approximate quantum predictive models can have an advantage over classical ones in terms of accuracy. To be more precise, they showed that there exist quantum approximate predictive models with quantum memory dimension $\hat{d}_q$ such that the distortion is provably lower compared to any classical approximate predictive model with classical memory dimension $\hat{d}_c = \hat{d}_q$. In order to show that, two major contributions where made. The first one is a classical discovery algorithm that only takes as input the desired model memory dimension $\hat{d}_q$ together with a length L sequence from the process $x_{1:L}$. It outputs a classical representation of a quantum predictive model for the stationary stochastic process that governs the distribution of $x_{1:L}$. The second contribution is a systematic way of finding a lower bound on the distortion of any classical predictive model, given a stochastic process with Markov order κ and the classical memory dimension $\hat{d}_c$. The distortion here is expressed with the KL divergence. In the following, the idea of the discovery algorithm and the systematic way of

finding lower bounds on the distortion are sketched.
The goal of the algorithm is to output a unitary U together with the encoding function E, which encodes the pasts $\overleftarrow{x}$ to suitable quantum states $|\sigma_{\overleftarrow{x}}\rangle$. In order to utilize machine learning techniques for this task, the discovery algorithm is formulated as a learning algorithm, with the goal of minimizing some cost function over a parameter set $\boldsymbol{\theta}$. It is shown that the model's output behavior can be defined by some operators $A = \{A_x\}$, where $A_x = \langle x|U|0\rangle$ reflects the action of the memory when interacting with the auxiliary system. Instead of optimizing over A, Yang et al. devise a way to optimize over some set of $\hat{d}_q \times \hat{d}_q$ complex matrices $B = \{B_x\}$, with the degrees of freedom in B being the trainable parameters $\boldsymbol{\theta}$. This is done since the operators $\{A_x\}$ are constrained to satisfy the completeness equation (cf. Definition 2.18). For the matrices B, the negative log-likelihood $-\log_2 P_B(x_{1:L})$ of producing the input sample $x_{1:L}$ can be efficiently computed and serves as the cost function that drives the learning. From the optimized matrices $\{B_x\}$, the operators $\{A_x\}$ and thus the unitary U as well as the encoding map E can be reconstructed.
The idea of finding lower bounds on the distortion of any classical predictive model is based on merging causal states and then optimizing over the next κ future samples. Importantly, the algorithm does not assume that the models have access only to a finite sample set. Thus, the search for the lowest distortion becomes an infinite dimensional problem. Yang et al. overcame this issue by defining a class of models for which a lower bound on the distortion turns out to be a finite dimensional search problem. Moreover, they showed that classical predictive models are part of this class and thus satisfy the same lower bound [11].
While the work of Yang et al. proves a superior accuracy of quantum approximate predictive models, it has the caveat that the learning algorithm remains classical. Since the memory requirement for a classical description of a quantum system scales exponentially in the number of qubits, the limit of a few tens of qubits that can be described classically is reached rapidly. Therefore, the approach is insufficient for practical applications. Learning a quantum model directly on a quantum computer, however, could overcome this issue, since no classical representation of the model is needed. This motivates the work at hand, namely to develop a quantum learning algorithm for quantum approximate predictive models. However, quantum circuits cannot provide direct access to the sample distribution, which leads to challenges in training such models. This issue has been studied in the context of so-called *implicit generative models*, which will be briefly discussed next.

2.4 Implicit Generative Models

In general, the task of generative modeling can be defined as follows. Given samples x_i from some unknown probability distribution P, output with high probability an efficient algorithm for generating new samples $\hat{x}_j$ from a good approximation $\hat{P}$ of the original distribution [17]. Generative models can be classified into *prescribed* and *implicit* models [34]. Prescribed generative models offer an *explicit* parameterization $\hat{P}_{\boldsymbol{\theta}}(x)$ of the distribution for generating new samples. This has the advantage that the distribution can be used to construct a cost function for the purpose of learning such models. Therefore, distance measures for probability distributions such as the KL divergence or the TV distance can be used for training and validation. By contrast, implicit generative models only define a stochastic procedure that directly generates new samples. That raises the question of which cost functions can be used to learn such models. Different approaches have been proposed in the literature, and some of them are presented next.
Liu et al. proposed a variational quantum circuit approach for generative models, the so-called quantum circuit Born machine (QCBM) [18]. The name QCBM derives from Born's rule, which refers to identifying the squared amplitude of a wave function as probability distribution [35]. Thus, the variational circuit generates samples via projective measurements, i.e., it generates a

sample x with probability

$$P_{\boldsymbol{\theta}}(x) = |\langle x|U(\boldsymbol{\theta})|0\rangle|^2, \tag{66}$$

where $U(\boldsymbol{\theta})$ represents the parameterized unitary of the VQC. In order to train the variational circuit, a cost function is needed. Liu et al. used the so-called *maximum mean discrepancy* (MMD) for this task, which is a kernel-based two-sample test [19], i.e., a statistical hypothesis testing technique that utilizes kernel methods. The MMD plays a crucial role in the present work and is thus described in detail in the next section, together with a brief overview about kernel methods. The quantum circuit for the QCBM consists of layers of arbitrary rotation gates $U(\theta_1,\theta_2,\theta_3) = R_z(\theta_1)R_x(\theta_2)R_z(\theta_3)$ together with layers of *CX* gates to induce correlations between qubits. For this ansatz, the gradient of the MMD can be calculated and gradient-based optimizers such as ADAM can be employed [36]. While non-gradient-based optimizers can also be used in principle, they failed to scale up to use a larger number of parameters [18]. Thus, gradient-based optimizers are preferred. As a proof of concept, the QCBM was tested for the Bars-and-Stripes dataset as well as for Gaussian mixture distributions [18].
Coyle et al. defined a QCBM with a special circuit structure, leading to an Ising Hamiltonian for the circuit [20]. For this model, they extended the methodology of the QCBM in two ways. First, the kernel in the MMD for the cost function is replaced by a quantum kernel, which is evaluated on quantum hardware. Second, the cost function itself is replaced by the Sinkhorn divergence [37], which is another method to compare probability distributions, similar to the MMD. Numerical results show that these two alterations can outperform previous approaches that are based only on the MMD [20].
While the works of Coyle et al. offer promising alternatives to the MMD, they are outside of the scope of this work. Yet, it is of great interest how these alternatives would perform in the domain of predictive modeling.

2.4.1 Maximum Mean Discrepancy

Proposed by Gretton et al., the MMD is a measure on the space of probability distributions [19], similar to the KL divergence or the TV distance from Section 2.1.3. However, a major difference lies in the fact that the MMD can be calculated based only on samples, and without direct access to the probability distributions. It is therefore particularly interesting for the area of implicit generative modeling. The following definitions and theorems are taken from Ref. [19].
The problem the MMD tries to solve can be formulated as follows. Let X and Y be random variables with respective probability distributions P and $\hat{P}$. Moreover, let $x = x_1,...,x_m$ and $y = y_1,...,y_n$ be independently and identically distributed observations from P and $\hat{P}$, respectively. Based on that, can it be decided whether $P \neq \hat{P}$? Theoretically, this can be answered with the following criterion.

Lemma 2.3. Let $(\mathscr{X}, d)$ be a metric space. Then $P = \hat{P}$ if and only if

$$\mathbb{E}_{x\sim P}[f(x)] = \mathbb{E}_{y\sim\hat{P}}[f(y)] \quad \text{for all } f \in C(\mathscr{X}), \tag{67}$$

where $C(\mathscr{X})$ is the space of all bounded continuous functions on $\mathscr{X}$.

While this lemma can be used theoretically to check whether $P = \hat{P}$, it can not be used practically since $C(\mathscr{X})$ has infinite dimension. However, a more general condition can be defined that nevertheless allows to uniquely determine whether $P = \hat{P}$, which is the maximum mean discrepancy.

Definition 2.31. Let $F(\mathscr{X})$ be a class of functions $f : \mathscr{X} \to \mathbb{R}$. The maximum mean discrepancy (MMD) is defined as

$$MMD[F,P,\hat{P}] = \sup_{f\in F}\Big(\mathbb{E}_{x\sim P}[f(x)] - \mathbb{E}_{y\sim\hat{P}}[f(y)]\Big). \tag{68}$$

For practical applications, the expectation values in the MMD can be approximated, which gives an estimate

$$MMD_e[F,P,\hat{P}] = \sup_{f\in F}\Big(\frac{1}{m}\sum_{i=1}^{m} f(x_i) - \frac{1}{n}\sum_{i=1}^{n} f(y_i)\Big). \tag{69}$$

If one chooses $F(\mathscr{X}) = C(\mathscr{X})$, one can use Lemma 2.3 to get $MMD[C(\mathscr{X}),P,\hat{P}] = 0$ if and only if $P = \hat{P}$. However, the key lies in the fact that a more handy function space F can be chosen that yields the same outcome. In particular, the unit ball of a so-called *universal reproducing kernel Hilbert space* (RKHS) is used [19]. The theory about RKHS and kernel functions is an independent discipline within mathematics and thus not discussed here in detail. An introduction to RKHSs can be found in textbooks such as in Ref. [38]. In essence, a reproducing kernel Hilbert space $F(\mathscr{X})$ is a Hilbert space of functions $f : \mathscr{X} \to \mathbb{R}$, where the point evaluation functionals are continuous. For such spaces, a function $k : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$ exists, such that

$$k(\cdot,\cdot) \text{ is symmetric and positive semi-definite and} \tag{70}$$

$$k(x,y) = \langle k(\cdot,x), k(\cdot,y)\rangle_F \text{ for all } x,y \in \mathscr{X}. \tag{71}$$

k is called *reproducing kernel*. A RKHS $F(\mathscr{X})$ is called *universal*, if it is dense in $C(\mathscr{X})$. For these spaces, Gretton et al. have shown that the MMD has the desired property to decide whether $P \neq \hat{P}$ [19]:

Lemma 2.4. Let $F(\mathscr{X})$ be the unit ball in a universal RKHS, defined on the compact metric space $\mathscr{X}$. Then

$$MMD[F,P,\hat{P}] = 0 \text{ if and only if } P = \hat{P}. \tag{72}$$

Having a suitable RKHS, one can obtain the squared MMD via kernel function evaluations [19]:

Theorem 2.1. Let X, X' be independent random variables with probability distribution P and Y, Y' be independent random variables with probability distribution $\hat{P}$, respectively. Then

$$MMD^2[F,P,\hat{P}] = \mathbb{E}_{x\sim P,x'\sim P}[k(x,x')] - 2\cdot\mathbb{E}_{x\sim P,y'\sim\hat{P}}[k(x,y)] + \mathbb{E}_{y\sim\hat{P},y'\sim\hat{P}}[k(y,y')]. \tag{73}$$

The random variables X', Y' in Theorem 2.1 are understood as independent copies of X, respectively of Y, with the same probability distribution. For practical applications, an estimator for the *MMD* is obtained via

$$MMD_e^2[F,P,\hat{P}] = \frac{1}{m(m-1)}\sum_{i=1}^{m}\sum_{j\neq i}^{m} k(x_i,x_j) - \frac{2}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} k(x_i,y_j) + \frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{j\neq i}^{n} k(y_i,y_j). \tag{74}$$

The proper choice of the RKHS, i.e., the choice of the kernel functions, can often be non-trivial and problem-dependent. It was shown that Gaussian RKHSs are universal [39] and can thus be used for the MMD. These RKHSs are induced by Gaussian kernel functions

$$k(x,y) = \frac{1}{n_\alpha}\sum_{i=1}^{n_\alpha}\exp\Big(-\frac{|x-y|^2}{2\alpha_i}\Big), \tag{75}$$

with $n_\alpha \in \mathbb{N}$ and so-called bandwidths $\alpha_i \in \mathbb{R}$. Gaussian kernels have been successfully used in practice for the QCBM [18] which is why we also rely on them for calculating the MMD for predictive models.

The areas of implicit generative and predictive modeling have been introduced separately from each other within this work. We did this because the two fields are also separately discussed in the literature, using different notations and semantics. However, both areas do relate to each other in several ways. To avoid confusion, the relationship between the two will be discussed in more detail in the next section.

2.4.2 Relation to Predictive Models

We start our discussion by noting that both fields aim to learn models where samples can be drawn according to some probability distribution. In the typical setting of implicit generative models, as introduced by Shakir and Balaji [34], this probability distribution is fixed, i.e., the output of one sample does not change the behavior of the model for outputting the next sample. Moreover, the framework of implicit generative modeling is embedded in the area of machine learning and therefore shares notations and semantics that is focused on learning patterns from data. By contrast, predictive models heavily rely on updating some internal states based on the output of previous samples. The generated samples thus follow a conditional probability distribution instead and can be arbitrarily long. A central quantity in predictive modeling is the memory, which is reflected in discussions about the topological and statistical complexity. The semantic and notation of predictive models is influenced by the field of information theory. In this field, a core concept is entropy, which is a measure for the average information or uncertainty of the outcome of a random variable. Thus, the classical statistical complexity is simply the Shannon entropy applied to stochastic processes [32]. Similarly, the quantum statistical complexity is the Von Neumann entropy [32]. The information-centric approach of predictive models is often motivated by Occam's razor; the internal states should encapsulate all the necessary information about the future but nothing more.
Within this work, we understand predictive models as a generalization of implicit generative models. This is motivated as follows. If a predictive model is iteratively applied to produce an output of length $n \in \mathbb{N}$, while being always initialized with the same fixed input, we essentially define a stochastic procedure that generates new samples of size n from a fixed probability distribution. This is precisely the definition of implicit generative models that we are referring to within this work. Therefore, we *extend* several concepts of the domain of implicit generative modeling to the domain of predictive modeling, which includes the KL divergence, the MMD and the QCBM.
To the end of this chapter, the different methods from the field of predictive modeling are summarized and discussed.

2.5 Discussion

In summary, the existing approaches for quantum predictive models either assume the availability of an ϵ-machine and derive quantum models from it (q-simulators) or train a quantum model on classical resources (discovery algorithm). The ϵ-machine of a particular stochastic process is mostly unknown and needs to be constructed with, e.g., the causal state splitting reconstruction algorithm [10]. However, since it has been shown that quantum predictive models perform at worst as well as the ϵ-machine, one could essentially skip learning the ϵ-machine and directly learn a quantum predictive model instead. For this purpose, the discovery algorithm by Yang et al. can be used [11]. This, however, leads to the problem that only models with a relatively small-sized quantum memory can be learned.
In contrast to this, we propose a way to learn a quantum predictive model directly on quantum devices, that, nevertheless, only takes data as input. This approach is inspired by the quantum circuit Born machine [18], where a VQC is trained iteratively with a cost function based on the MMD. For learning predictive models, however, the MMD needs to be extended to take the conditional behavior of the probability distributions into account. Moreover, the VQC of our learning algorithm features two quantum registers M_q and A_q, where the first one is used to store the quantum memory states and the second is used for measuring future samples. The learned model can thus be seen as an approximated q-simulator, which follows the same idea of repeatedly measuring A_q, letting M_q collapse to a memory state. Furthermore, we employ the same validation metrics as Yang et al. [11], allowing a direct comparison between classical and quantum learned predictive models.

3 Methodology

The overall goal of this work is to develop a quantum learning algorithm for quantum approximate predictive models. After training these models, they should be able to simulate the corresponding stationary stochastic process with high accuracy. This chapter explains how this issue is investigated in detail, after a rough overview is given first.
The learning algorithm follows the idea of a variational quantum algorithm, where a VQC is trained based on a cost function in an iterative hybrid quantum-classical optimization procedure. We are aiming for learning a model based only on data, without further knowledge of the process. Therefore, the algorithm is given access only to a sample $x_{1:L} \in A^L$, drawn from the stationary stochastic process. In order to achieve a reasonable performance of the model, the input sample should be sufficiently long. Consequently, it is infeasible to load the entire sample into the VQC at once. Therefore, the learning algorithm features a *classical pre-processing step* that splits $x_{1:L}$ into a training data set consisting of past-future-pairs $\{(\overleftarrow{x}, \overrightarrow{x})\}$, where the length of the past and future samples are l_p and l_f, respectively. The VQC is initialized alternately with pasts from this set and the generated outputs $\overrightarrow{y}$ are compared with the known futures $\overrightarrow{x}$. This comparison is implicitly done by calculating the maximum mean discrepancy, which serves as a distance measure for the two underlying probability distributions and is used as a cost function for the learning algorithm.
The classical pre-processing step raises the question of how long the samples $\overrightarrow{y}$ and $\overrightarrow{x}$ should be. In the domain of predictive modeling, we are aiming to have models that can simulate arbitrarily many future time steps. Yet, we only have access to finite samples and thus can only perform a two-sample test based on a finite length. This means that we are essentially minimizing $|P(X_{1:l_f}|\overleftarrow{x}) - \hat{P}(X_{1:l_f}|\overleftarrow{x})|$, for some distance measure $|\cdot|$. However, there is no reason to assume that a model trained based on such a cost function will perform well for time steps larger than l_f. The ability of exact predictive models to accurately simulate arbitrarily many time steps relies on the use of causal states in the classical setting and memory states in the quantum setting. We therefore add a regularization term to the cost function that penalizes models with a rich set of internal states. This is heuristically motivated since the causal states of an ϵ-machine are the minimal set of states that is needed to faithfully sample future trajectories. Within this chapter, we show that such a regularization term can be obtained via performing a *quantum post-processing step*.
To utilize gradient-based optimizers, the cost function needs to be differentiable and the gradient efficiently computable. We show that our proposed cost function is differentiable and define an ansatz, for which gradients can be efficiently calculated. This ansatz is created based on the following steps. We start with defining a class of stochastic processes that are considered in detail throughout this work and that we use to validate the learned models. Based on these stochastic processes, the next step is then to construct a *static* quantum circuit that corresponds to the q-simulator of these processes, i.e., a quantum exact predictive model. The circuit is called *static* because no rotation gates are used here. Next, we decompose these gates step by step into rotation and *CX* gates, such that the output is a parameterized quantum circuit, which serves as the ansatz for the learning algorithm. While this ansatz can be used in general, the approach described above has the advantage that the models can easily be validated for the stochastic processes that we consider, since the ansatz contains an optimal solution by construction. Moreover, we can use the parameter-shift rule to calculate gradients with respect to the trainable parameters.

The chapter is arranged as follows. Section 3.1 starts with the introduction of the model problem that is used throughout this work. Next, in Section 3.2, the KL divergence and the TV distance are extended to the domain of predictive modeling such that they can be used to validate the learned models. Afterward, in Section 3.3, the quantum learning algorithm is presented in more detail, which includes the choice of the cost function, the classical pre-processing step,

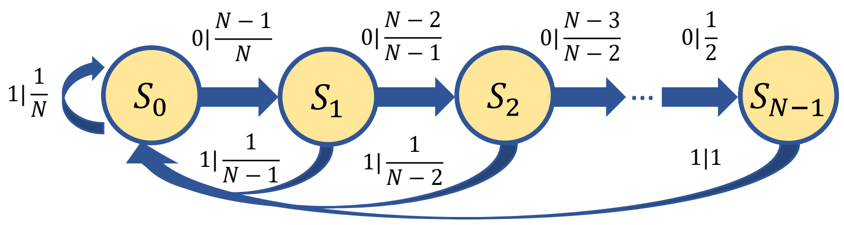

Figure 8: Illustration of the ϵ-machine of the period-N uniform renewal process. The nodes indicate the causal states $S_0, S_1, \ldots, S_{N-1}$ and the labels $x|P(x)$ indicate that the output x is produced with probability $P(x)$ when transitioning to the next causal state. The figure is based on Figure 6 from Ref. [11].

and the quantum post-processing step. The ansatz for the VQC is derived in Section 3.4, where we construct the circuit for the q-simulator first and gradually decompose the gates yielding a parameterized quantum circuit. Subsequently, Section 3.5 covers the computation of the gradient for the cost function. This includes in particular the common parameter-shift rule as well as an extension thereof. At the end of this chapter, the quantum learning algorithm is summarized in Section 3.6.

3.1 Model Problem

An iconic non-Markovian process will serve as the model problem, namely the *period-N uniform renewal process* [11]. This process uses the binary alphabet $A = \{0,1\}$ and produces bitstrings as output, where the number of zeros between two adjacent ones is uniformly distributed between 0 and $N-1$. The corresponding ϵ-machine is illustrated in Figure 8. It shows N causal states, where past observations ending with a 1 are mapped to S_0 and past observations ending with n-times a 0, $n \in \{1,2,\ldots,N-1\}$, are mapped to S_n.
In the following paragraphs, some important properties of the period-N uniform renewal process are presented. This includes the Markov order, transition probabilities, stationary probabilities, and conditional probabilities for future samples based on past observations. Moreover, the topological and classical complexities are calculated. The analysis of the period-N uniform renewal process is limited to the periods $N=2$ and $N=3$ since we consider these in practice.
We start with the Markov order of the process.

Lemma 3.1. The period-N uniform renewal process has Markov order $N-1$.

Proof. The longest past observation that is needed to identify the corresponding causal state has length $n = N-1$. This observation is n times a 0, and it is mapped to the state S_{N-1}. Thus we get

$$P(X_1|\overleftarrow{X}) = P\big(X_1|E(\overleftarrow{X})\big) = P\big(X_1|E(X_{N-1:0})\big) = P(X_1|X_{N-1:0}), \tag{76}$$

i.e., the next time step depends at most on the past $N-1$ time steps. □

Since the ϵ-machine has the structure of a hidden Markov model, it is based on transition probabilities $T_{i,j}$ to transition from causal state j to i. We can read out these probabilities from Figure 8 and organize them in a matrix, leading to

$$\boldsymbol{T}^{(2)} = \begin{bmatrix} 1/2 & 1 \\ 1/2 & 0 \end{bmatrix} \quad \text{and} \quad \boldsymbol{T}^{(3)} = \begin{bmatrix} 1/3 & 1/2 & 1 \\ 2/3 & 0 & 0 \\ 0 & 1/2 & 0 \end{bmatrix}, \tag{77}$$

for the period-2 and the period-3 uniform renewal process, respectively. Note that the superscript indicates the period of the process. Based on the transition probabilities, the stationary distributions of the process can be calculated.

Lemma 3.2. The period-2 and period-3 uniform renewal processes have the following stationary distributions:

$$\boldsymbol{\pi}^{(2)} = \begin{bmatrix} 2/3 \\ 1/3 \end{bmatrix}, \quad \boldsymbol{\pi}^{(3)} = \begin{bmatrix} 1/2 \\ 1/3 \\ 1/6 \end{bmatrix}. \tag{78}$$

Proof. The stationary distribution is defined as the fixed-point of the transition matrix, i.e., $\boldsymbol{T\pi} = \boldsymbol{\pi}$. Thus, the distribution can be obtained by calculating the eigenvector for the eigenvalue $\lambda = 1$, which is

$$\boldsymbol{\pi}^{(2)} = x^{(2)} \cdot \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \quad \boldsymbol{\pi}^{(3)} = x^{(3)} \cdot \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}, \tag{79}$$

where $x^{(2)}, x^{(3)} \in \mathbb{R}$ can be chosen arbitrarily. Since the vectors are interpreted as a probability distribution, the normalization condition $\sum_i \pi_i = 1$ yields $x^{(2)} = 1/3$ and $x^{(3)} = 1/6$. □

These distributions are understood as the asymptotic probability to be in a certain causal state for many i.i.d. simulations, i.e., $\pi_i = P(S_i)$. Given the distributions for the causal states, the stationary distribution of past observations $P(\overleftarrow{X})$ can be calculated.

Lemma 3.3. The period-2 and period-3 uniform renewal processes have the following stationary distribution of past observations:

$$P^{(2)}(X_0 = 0) = 1/3, \quad P^{(2)}(X_0 = 1) = 2/3, \tag{80}$$

and

$$P^{(3)}(X_{-1} = 0, X_0 = 0) = 1/6, \quad P^{(3)}(X_{-1} = 0, X_0 = 1) = 1/3, \tag{81}$$

$$P^{(3)}(X_{-1} = 1, X_0 = 0) = 1/3, \quad P^{(3)}(X_{-1} = 1, X_0 = 1) = 1/6. \tag{82}$$

Proof. In order to calculate the stationary past distributions, we apply the law of the total probability, i.e.,

$$P(X) = \sum_i P(X|S_i)P(S_i), \tag{83}$$

with the stationary distributions $P(S_i) = \pi_i$. For the period-2 uniform renewal process, we get

$$P^{(2)}(X_0 = 0) = \sum_{i=0}^{1} P(X_0 = 0|S_i)\pi_i^{(2)} = 1/3, \tag{84}$$

$$P^{(2)}(X_0 = 0) = \sum_{i=0}^{1} P(X_0 = 0|S_i)\pi_i^{(2)} = 2/3, \tag{85}$$

and for the period-3 uniform renewal process

$$P^{(3)}(X_{-1} = 0, X_0 = 0) = \sum_{i=0}^{2} P(X_{-1} = 0, X_0 = 0|S_i)\pi_i^{(3)} = 1/6, \tag{86}$$

$$P^{(3)}(X_{-1} = 0, X_0 = 1) = \sum_{i=0}^{2} P(X_{-1} = 0, X_0 = 1|S_i)\pi_i^{(3)} = 1/3, \tag{87}$$

$$P^{(3)}(X_{-1} = 1, X_0 = 0) = \sum_{i=0}^{2} P(X_{-1} = 1, X_0 = 0|S_i)\pi_i^{(3)} = 1/3, \tag{88}$$

$$P^{(3)}(X_{-1} = 1, X_0 = 1) = \sum_{i=0}^{2} P(X_{-1} = 1, X_0 = 1|S_i)\pi_i^{(3)} = 1/6, \tag{89}$$

where the conditional probabilities can be read off from the ϵ-machine, see Figure 8. □

Period-2 uniform renewal process				
$\overleftarrow{x}$	State	$P(\overleftarrow{x})$	$P(0\|\overleftarrow{x})$	$P(1\|\overleftarrow{x})$
0	S_1	1/3	0	1
1	S_0	2/3	1/2	1/2

Period-3 uniform renewal process				
$\overleftarrow{x}$	State	$P(\overleftarrow{x})$	$P(0\|\overleftarrow{x})$	$P(1\|\overleftarrow{x})$
00	S_2	1/6	0	1
01	S_0	1/3	2/3	1/3
10	S_1	1/3	1/2	1/2
11	S_0	1/6	2/3	1/3

Table 1: Conditional probabilities and stationary past distributions of the period-2 (left) and period-3 (right) uniform renewal process. Additionally, the mappings from past observations to the causal states are listed.

The conditional probabilities, stationary past distributions and causal state mappings are summarized in Table 1. These properties are used for validating the learned models.
Besides the probability distributions, we also calculate the classical statistical and topological complexity.

Lemma 3.4. The period-2 and period-3 uniform renewal process have the following classical topological and statistical complexities:

$$d_c^{(2)} = 1, \quad C_c^{(2)} \approx 0.92, \tag{90}$$

$$d_c^{(3)} \approx 1.59, \quad C_c^{(3)} \approx 1.46. \tag{91}$$

Proof. The topological complexities can be calculated by counting the number of causal states of the corresponding ϵ-machine and taking the base-2 logarithm (see Definition 2.24). This yields

$$d_c^{(2)} = \log_2 2 = 1 \text{ and } d_c^{(3)} = \log_2 3 \approx 1.59. \tag{92}$$

For the statistical complexities, we apply Definition 2.25 which results in

$$C_c^{(2)} = -\sum_{i=0}^{1} \pi_i^{(2)} \log\left(\pi_i^{(2)}\right) = -\frac{2}{3}\log_2\left(\frac{2}{3}\right) - \frac{1}{3}\log_2\left(\frac{1}{3}\right) \approx 0.92 \tag{93}$$

and

$$C_c^{(3)} = -\sum_{i=0}^{2} \pi_i^{(3)} \log\left(\pi_i^{(3)}\right) = -\frac{1}{2}\log_2\left(\frac{1}{2}\right) - \frac{1}{3}\log_2\left(\frac{1}{3}\right) - \frac{1}{6}\log_2\left(\frac{1}{6}\right) \approx 1.46, \tag{94}$$

respectively. □

Lemma 3.4 shows that one bit is sufficient for classical models to faithfully sample future trajectories for the period-2 uniform renewal process. Therefore, a quantum model using only one qubit as a memory register is sufficient as well. By contrast, for the period-3 uniform renewal process, at least two bits are required. If the memory was restricted to only one bit, no classical predictive model would be able to exactly simulate the period-3 uniform renewal process. However, since the quantum topological complexity can be strictly lower than its classical counterpart, a quantum model with only one qubit might perform better, i.e., with fewer statistical errors. In order to quantify these errors, some metrics are needed. In Section 2.1.3, we introduced the KL divergence and the TV distance, which are distance measures for probability distributions. However, stochastic processes offer a conditional probability distribution and include uncountably many random variables $\overrightarrow{X} = X_1, X_2, \ldots$. We thus extend these measures in the next section to be applicable for comparing the distributions of stochastic processes with the distributions of predictive models.

3.2 Validation Metrics

The KL divergence compares two probability distributions P and $\hat{P}$ according to

$$\mathscr{D}_{KL}(P,\hat{P}) = \sum_{x\in A} P(x)\log_2\left(\frac{P(x)}{\hat{P}(x)}\right), \tag{95}$$

cf. Definition 2.16. We extend this formula in two ways. On the one hand, we introduce the conditional behavior and average over the pasts $\overleftarrow{x}$. On the other hand, we include multiple future time steps $X_{1:L}$ and take the limit $L \to \infty$. This extension to the KL divergence has already been done by Yang et al. [11], which we follow here, albeit using a slightly different notation. In the following, the probability distribution P refers to a stationary stochastic process $\{X_t\}$, whereas $\hat{P}$ denotes the distribution of an approximate predictive model for $\{X_t\}$.

Definition 3.1. The *normalized conditional* KL divergence is defined as

$$\mathscr{D}_{KL}(L,P,\hat{P}|\overleftarrow{x}) = \frac{1}{L}\mathscr{D}_{KL}\left(P(X_{1:L}|\overleftarrow{x}),\hat{P}(X_{1:L}|\overleftarrow{x})\right) \tag{96}$$

for any $L \in \mathbb{N}$ and possible past $\overleftarrow{x}$.

This measure can be taken to compare models for one particular past $\overleftarrow{x}$ and a fixed number of time steps L. However, we are interested in the overall performance and thus average over all possible pasts.

Definition 3.2. The *normalized* KL divergence is defined as

$$D_{KL}(L,P,\hat{P}) = \sum_{\overleftarrow{x}} P(\overleftarrow{x})\mathscr{D}_{KL}(L,P,\hat{P}|\overleftarrow{x}) \tag{97}$$

for any $L \in \mathbb{N}$.

The use of $\overleftarrow{x}$ indicates an infinitely long past sample. However, for stochastic processes of finite Markov order, the sum in Equation (97) is finite as well. The last step now is to take the limit $L \to \infty$ to get the mean value over all future time steps.

Definition 3.3. The *mean* KL divergence is defined as

$$D_{KL}(P,\hat{P}) = \lim_{L\to\infty} D_{KL}(L,P,\hat{P}). \tag{98}$$

This measure is able to capture all the important properties, i.e., the statistics for all future time steps based on all possible pasts. We can fully write it down as

$$D_{KL}(P,\hat{P}) = \lim_{L\to\infty}\sum_{x_{1:L}\in A^L}\sum_{\overleftarrow{x}}\frac{P(\overleftarrow{x})\cdot P(x_{1:L}|\overleftarrow{x})}{L}\log_2\left(\frac{P(x_{1:L}|\overleftarrow{x})}{\hat{P}(x_{1:L}|\overleftarrow{x})}\right). \tag{99}$$

An important question at this point is if the mean KL divergence is well-defined, or in other words, if the limit in Equation (99) exists. Yang et al. [11] showed that this is the case. Furthermore, they showed that the mean KL divergence can already be obtained by considering only a single future time step:

Lemma 3.5. Let P and $\hat{P}$ be the probability distributions of two stationary stochastic processes $\{X_t\}$ and $\{\hat{X}_t\}$, respectively. Moreover, let $L \in \mathbb{N}$ be arbitrary but fixed. Then

$$D_{KL}(P,\hat{P}) = D_{KL}(L,P,\hat{P}) = D_{KL}(1,P,\hat{P}). \tag{100}$$

The proof is based on Bayes' theorem and exploiting the property of stationarity. [1]
A word of caution: In general, Lemma 3.5 cannot be applied to a predictive model and its underlying stationary stochastic process directly. This is the case since the probability distribution $\hat{P}$ defined by an approximate predictive model does not need to be stationary. The general case is that we can only upper bound the normalized KL divergence, as described in the following lemma.

Lemma 3.6. Let P and $\hat{P}$ be probability distributions and $L_1, L_2 \in \mathbb{N}$. Then

$$D_{KL}(L_1, P, \hat{P}) \leq \frac{L_2}{L_1} D_{KL}(L_2, P, \hat{P}) \tag{101}$$

for $L_1 < L_2$.

Proof. Let $L \in \mathbb{N}$ be arbitrary but fixed. Then for any $n \in \mathbb{N}$ we have

$$\begin{aligned}
L D_{KL}(L, P, \hat{P}) &= \sum_{x_{1:L}} \sum_{\overleftarrow{x}} P(\overleftarrow{x}) \cdot P(x_{1:L}|\overleftarrow{x}) \log_2 \left(\frac{P(x_{1:L}|\overleftarrow{x})}{\hat{P}(x_{1:L}|\overleftarrow{x})} \right) && (102) \\
&= \sum_{\overleftarrow{x}} \sum_{x_{1:L}} P(\overleftarrow{x}) \cdot P(x_{1:L}|\overleftarrow{x}) \log_2 \left(\frac{P(x_{1:L}|\overleftarrow{x})}{\hat{P}(x_{1:L}|\overleftarrow{x})} \right) && (103) \\
&= \sum_{\overleftarrow{x}} \sum_{x_{1:L}} P(\overleftarrow{x}) \Big(\sum_{x_{L:L+n}} P(x_{1:L+n}|\overleftarrow{x}) \Big) \log_2 \left(\frac{\sum_{x_{L:L+n}} P(x_{1:L+n}|\overleftarrow{x})}{\sum_{x_{L:L+n}} \hat{P}(x_{1:L+n}|\overleftarrow{x})} \right) && (104) \\
&\leq \sum_{\overleftarrow{x}} \sum_{x_{1:L}} P(\overleftarrow{x}) \sum_{x_{L:L+n}} P(x_{1:L+n}|\overleftarrow{x}) \log_2 \left(\frac{P(x_{1:L+n}|\overleftarrow{x})}{\hat{P}(x_{1:L+n}|\overleftarrow{x})} \right) && (105) \\
&= \sum_{\overleftarrow{x}} \sum_{x_{1:L+n}} P(\overleftarrow{x}) \cdot P(x_{1:L+n}|\overleftarrow{x}) \log_2 \left(\frac{P(x_{1:L+n}|\overleftarrow{x})}{\hat{P}(x_{1:L+n}|\overleftarrow{x})} \right) && (106) \\
&= (L+n) D_{KL}(L+n, P, \hat{P}), && (107)
\end{aligned}$$

where we used the log sum inequality

$$\Big(\sum_i a_i \Big) \log \left(\frac{\sum_i a_i}{\sum_i b_i} \right) \leq \sum_i a_i \log \left(\frac{a_i}{b_i} \right). \tag{108}$$

Thus, we have

$$D_{KL}(L, P, \hat{P}) \leq \frac{L+n}{L} D_{KL}(L+n, P, \hat{P}), \tag{109}$$

which completes the proof. □

Therefore, if we calculate the mean KL divergence for approximate predictive models, we expect it to increase with increasing time steps i. Thus, calculating $D_{KL}(i, P, \hat{P})$ for $i = 1, 2, 3, \ldots$ can offer an indication of "how well" the approximate model is able to produce multiple future time steps.
Besides the KL divergence, we have also introduced the TV distance as a distance measure for probability distribution in Section 2.1.3. This measure can be extended to stochastic processes in a similar way.

Definition 3.4. The *conditional* TV distance is defined as

$$\mathscr{D}_{TV}(L, P, \hat{P}|\overleftarrow{x}) = \mathscr{D}_{TV}\big(P(X_{1:L}|\overleftarrow{x}), \hat{P}(X_{1:L}|\overleftarrow{x})\big), \tag{110}$$

for any $L \in \mathbb{N}$ and possible past $\overleftarrow{x}$.

[1]The full proof can be found in [11] as Lemma 2 in Appendix F.

To get an overall performance, we sum up all the values for different pasts.

Definition 3.5. The *full* TV distance is defined as

$$D_{TV}(L,P,\hat{P}) = \sum_{\overleftarrow{x}} \mathscr{D}_{TV}(L,P,\hat{P}|\overleftarrow{x}), \tag{111}$$

for any $L \in \mathbb{N}$.

We can write the full expression as

$$D_{TV}(L,P,\hat{P}) = \frac{1}{2}\sum_{x_{1:L}\in A^L}\sum_{\overleftarrow{x}} |P(x_{1:L}|\overleftarrow{x}) - \hat{P}(x_{1:L}|\overleftarrow{x})|. \tag{112}$$

Note that while we include multiple future time steps $X_{1:L}$, we do not divide by L. Similarly, we sum over all pasts but do not weigh each past with its likelihood $P(\overleftarrow{x})$. This is because for the TV distance, we are interested in the absolute point-wise difference between the two distributions. At this point, we took the limit $L \to \infty$ for the normalized KL divergence to get the mean value. However, this can not be done for the full TV distance since the limit does not need to exist in general. Yet, we can upper bound the full TV distance similarly to the normalized KL divergence as follows.

Lemma 3.7. Let P and $\hat{P}$ be probability distributions and $L_1, L_2 \in \mathbb{N}$. Then

$$D_{TV}(L_1,P,\hat{P}) \leq D_{TV}(L_2,P,\hat{P}) \tag{113}$$

for $L_1 < L_2$.

Proof. The result is a direct application of the triangle inequality, i.e.,

$$\begin{aligned}
D_{TV}(L_1,P,\hat{P}) &= \frac{1}{2}\sum_{x_{1:L_1}}\sum_{\overleftarrow{x}} |P(x_{1:L_1}|\overleftarrow{x}) - \hat{P}(x_{1:L_1}|\overleftarrow{x})| && (114)\\
&= \frac{1}{2}\sum_{\overleftarrow{x}}\sum_{x_{1:L_1}} |P(x_{1:L_1}|\overleftarrow{x}) - \hat{P}(x_{1:L_1}|\overleftarrow{x})| && (115)\\
&= \frac{1}{2}\sum_{\overleftarrow{x}}\sum_{x_{1:L_1}} \Big|\sum_{x_{L_1:L_2}} P(x_{1:L_2}|\overleftarrow{x}) - \sum_{x_{L_1:L_2}} \hat{P}(x_{1:L_2}|\overleftarrow{x})\Big| && (116)\\
&= \frac{1}{2}\sum_{\overleftarrow{x}}\sum_{x_{1:L_1}} \Big|\sum_{x_{L_1:L_2}} \big(P(x_{1:L_2}|\overleftarrow{x}) - \hat{P}(x_{1:L_2}|\overleftarrow{x})\big)\Big| && (117)\\
&\leq \frac{1}{2}\sum_{\overleftarrow{x}}\sum_{x_{1:L_1}}\sum_{x_{L_1:L_2}} |P(x_{1:L_2}|\overleftarrow{x}) - \hat{P}(x_{1:L_2}|\overleftarrow{x})| && (118)\\
&= \frac{1}{2}\sum_{\overleftarrow{x}}\sum_{x_{1:L_2}} |P(x_{1:L_2}|\overleftarrow{x}) - \hat{P}(x_{1:L_2}|\overleftarrow{x})| && (119)\\
&= D_{TV}(L_2,P,\hat{P}). && (120)
\end{aligned}$$

□

Therefore, we also take the full TV distance as a validation metric for fixed numbers $i = 1,2,3,\ldots$ of time steps. Similar to the normalized KL divergence, we expect the full TV distance to increase for multiple future time steps.

Now that we have discussed the validation metrics, we present the learning algorithm in more detail in the next section.

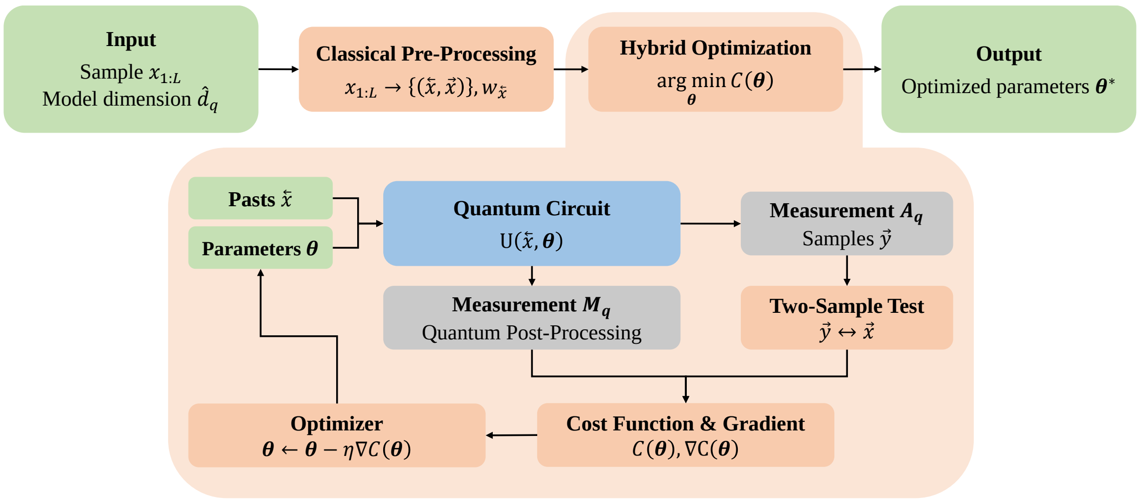
Input
Sample $x_{1:L}$
Model dimension $\hat{d}_q$
Classical Pre-Processing
$x_{1:L} \to \{(\overleftarrow{x}, \overrightarrow{x})\}, w_{\overleftarrow{x}}$
Hybrid Optimization
$\arg\min_{\boldsymbol{\theta}} C(\boldsymbol{\theta})$
Output
Optimized parameters $\boldsymbol{\theta}^*$
Pasts $\overleftarrow{x}$
Parameters $\boldsymbol{\theta}$
Quantum Circuit
$U(\overleftarrow{x}, \boldsymbol{\theta})$
Measurement A_q
Samples $\overrightarrow{y}$
Measurement M_q
Quantum Post-Processing
Two-Sample Test
$\overrightarrow{y} \leftrightarrow \overrightarrow{x}$
Optimizer
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla C(\boldsymbol{\theta})$
Cost Function & Gradient
$C(\boldsymbol{\theta}), \nabla C(\boldsymbol{\theta})$

Figure 9: Schema of the quantum learning algorithm. The algorithm takes as input one sample $x_{1:L}$, together with the desired model memory size $\hat{d}_q$. At first, a classical pre-processing step is done, which outputs past-future-pairs $\{(\overleftarrow{x}, \overrightarrow{x})\}$ and some weights $w_{\overleftarrow{x}}$. Afterward, the hybrid optimization is performed. Here, a VQC is initialized with some past observations, executed, and measured. The measurement of the auxiliary register A_q leads to samples of future trajectories $\overrightarrow{y}$, that are used to perform a two-sample test. Measuring the memory register M_q corresponds to a quantum post-processing step that yields a regularization term which penalizes models with a rich set of internal states. Both pieces of information together are used to calculate the cost function $C(\boldsymbol{\theta})$ and its gradient $\nabla C(\boldsymbol{\theta})$, and a classical optimizer updates the trainable parameters based on that. The output of the learning algorithm are the trained parameters $\boldsymbol{\theta}^*$.

3.3 Quantum Learning Algorithm

Within this section, the proposed quantum learning algorithm for predictive models is explained in detail. The overall procedure is illustrated in Figure 9. As input, the learning algorithm is given access to a sample $x_{1:L} \in A^L$ of length $L \in \mathbb{N}$, and the desired model quantum memory size $\hat{d}_q$. A classical pre-processing step takes the sample and splits it into pairs of pasts and futures $\{(\overleftarrow{x}, \overrightarrow{x})\}$ of a certain length. Additionally, some weights $w_{\overleftarrow{x}}$ are calculated which will be part of the cost function and take the frequencies of different pasts into account. This pre-processing step is done only once, as an initialization step of the learning algorithm. Afterward, the hybrid optimization loop is performed. Here, the VQC is initialized with a given past $\overleftarrow{x}$ and the trainable parameters $\boldsymbol{\theta}$, and both registers, the auxiliary register A_q as well as the memory register M_q are measured. On the one hand, the measurement outcomes of A_q are interpreted as future trajectories and are used to perform a two-sample test via calculating the MMD. The samples thus follow the probability distribution $\hat{P}_{\boldsymbol{\theta}}$ defined by the VQC. On the other hand, the measurement of M_q corresponds to the outcome of a quantum post-processing step yielding a regularization term that penalizes models with a rich set of internal states. Both outcomes are used together to compute the cost function $C(\boldsymbol{\theta})$ and its gradient $\nabla C(\boldsymbol{\theta})$. The value of the cost function determines whether a convergence criterion is satisfied and the gradient is used to update the trainable parameters according to a classical gradient-based optimizer. The output of the learning algorithm is an optimized parameter set $\boldsymbol{\theta}^*$, which can be used together with the quantum circuit to simulate the underlying stationary stochastic process.
One important part of a learning algorithm is the cost function, which should reflect “how well” a model candidate solves the given problem. In principle, the KL divergence and the TV distance can be used for this task. However, both measures take as input the probability distri-

bution of the model, which is not directly accessible in the quantum setting. We therefore only take these measures as validation metrics, i.e., to measure their simulation performance, and propose a cost function that can be used as a computable proxy instead. Our cost function is strongly related to the structure of the VQC. Thus, an overview of the VQC is given first, and the cost function is derived based on that afterward.

3.3.1 General Structure of the VQC

In general, we split the VQC into two parts, one data encoding unit E and one unitary T, both of which are trained within an optimization loop. In the setting of predictive models, E corresponds to the deterministic map that encodes pasts $\overleftarrow{x}$ to a memory state, i.e.,

$$E(\overleftarrow{x})\,|0\rangle = |\sigma_{\overleftarrow{x}}\rangle\,, \tag{121}$$

with $|\sigma_{\overleftarrow{x}}\rangle$ being the memory state for a past $\overleftarrow{x}$ and $|0\rangle$ the initial state of the memory register. Following the idea of the q-simulator, the unitary T is acting on these states together with an auxiliary register A_q, such that a measurement of the latter generates a future sample while the memory register M_q collapses to another memory state. We can write this formally as

$$T\,|\sigma_{\overleftarrow{x}}\rangle\,|0\rangle = \sum_{x\in A}\sqrt{P(x|\overleftarrow{x})}\,|\sigma_{\overleftarrow{x},x}\rangle\,|x\rangle \xrightarrow{I\otimes M} |\sigma_{\overleftarrow{x},x^*}\rangle\,|x^*\rangle\,, \tag{122}$$

with $|\sigma_{\overleftarrow{x},x}\rangle$ being the memory state associated with the observation of $(\overleftarrow{x},x)$ and x^* being the measurement outcome of A_q. Note that Equation (122) exactly describes the property of a q-simulator, cf. Equation (56).
Within this work, we consider VQCs with a fixed ansatz, i.e., a fixed structure of gates, but parameterized by some real values $\boldsymbol{\theta}\in\mathbb{R}^n$. Thus, the unitary T as well as the data encoding unit become dependent on these parameters, i.e., $T = T(\boldsymbol{\theta})$ and $E(\overleftarrow{x},\boldsymbol{\theta})$. The goal of the learning algorithm is to train these parameters, such that the unitaries $E(\overleftarrow{x},\boldsymbol{\theta}^*)$ and $T(\boldsymbol{\theta}^*)$ for an optimized parameter set $\boldsymbol{\theta}^*$ act according to the above formulas. However, depending on the ansatz and the stochastic process, it is possible that no optimal solution exists, or in other words, that no choice of $\boldsymbol{\theta}$ leads to the unitaries described above. In this case, the best we can hope for is to have unitaries that offer a good approximation $\hat{P}_{\boldsymbol{\theta}}$ of the true underlying probability distribution, i.e.,

$$T(\theta)\,|\sigma_{\overleftarrow{x}}\rangle\,|0\rangle = \sum_{x\in A}\sqrt{\hat{P}_{\boldsymbol{\theta}}(x|\overleftarrow{x})}\,|\sigma_{\overleftarrow{x},x}\rangle\,|x\rangle\,. \tag{123}$$

This requires that we are able to successfully learn two unitaries with different targets; the data encoding unit $E(\overleftarrow{x},\boldsymbol{\theta})$ to generate memory states and the unitary $T(\boldsymbol{\theta})$ to approximate the process' probability distribution while acting on these states. In general, the output of $T(\boldsymbol{\theta})$ might not be a memory state, but simply one internal state of the model. However, it is desirable to have memory states since we can only train the VQC to approximate finitely many future time steps. If the output states of the unitary T matches the memory states of the data encoding unit E, we call these states a *valid set of memory states*. For the case that we can ensure both, $T(\boldsymbol{\theta})$ approximates finitely many time steps accurately and outputs memory states, we know that $T(\boldsymbol{\theta})$ satisfies Equation (123) and thus accurately approximate arbitrarily many time steps. These two goals are treated separately within the proposed cost function, which is introduced next.

3.3.2 Cost Function

Formally, the two requirements are introduced by splitting the cost function $C(\boldsymbol{\theta})$ into two parts, one as a measure $D(\boldsymbol{\theta})$ of the distortion and one as a measure $R(\boldsymbol{\theta})$ for the ability of

$T(\boldsymbol{\theta})$ to output memory states, i.e.,

$$C(\boldsymbol{\theta}) = D(\boldsymbol{\theta}) + R(\boldsymbol{\theta}). \tag{124}$$

From this form, $R(\boldsymbol{\theta})$ can be seen as a regularization term which introduces our knowledge about an optimal solution to the problem. Since the model needs to be initialized based on a given past $\overleftarrow{x}$, the two measures depend on $\overleftarrow{x}$ and the overall cost function is split per past, but summed up together such that

$$C(\boldsymbol{\theta}) = \sum_{\overleftarrow{x}} D_{\overleftarrow{x}}(\boldsymbol{\theta}) + R_{\overleftarrow{x}}(\boldsymbol{\theta}). \tag{125}$$

So far, the future statistics of each past are treated equally. However, if a past $\overleftarrow{x}$ occurs relatively rarely in the stochastic process, it should be given a lower priority, since its impact on the overall performance is also relatively small. Imagine a past $\overleftarrow{x}'$ which occurs only a few times. Even if a model fails to faithfully sample future trajectories for $\overleftarrow{x}'$, it will only happen a few times and thus will not significantly influence the overall performance. Therefore, weights $w_{\overleftarrow{x}}$ are introduced with the aim to prioritize frequent pasts. In order to balance the two measures, a hyper-parameter $c \in \mathbb{R}^+$ is added, such that the overall cost function can be read as

$$C(\boldsymbol{\theta}) = \sum_{\overleftarrow{x}} w_{\overleftarrow{x}} D_{\overleftarrow{x}}(\boldsymbol{\theta}) + cR_{\overleftarrow{x}}(\boldsymbol{\theta}). \tag{126}$$

Measuring the distortion in the sense of the KL divergence or the TV distance is not possible, since the probability distribution of the VQC is not accessible. However, the MMD can be used for the same task but can also be estimated based only on samples drawn from the training data set and the model. For repetition, given a universal RKHS F with kernel k, the MMD can be calculated as

$$MMD^2[F, P, \hat{P}] = \mathbb{E}_{x,x'}[k(x,x')] - 2\,\mathbb{E}_{x,y}[k(x,y)] + \mathbb{E}_{y,y'}[k(y,y')], \tag{127}$$

where x, x' are drawn from P and y, y' from $\hat{P}$, respectively, cf. Theorem 2.1. The MMD is based on probability distributions $P(x)$, but can be straightforwardly extended to take conditional probabilities into account. Thus, P and $\hat{P}$ are simply replaced by $P(\cdot|\overleftarrow{x})$ and $\hat{P}_{\boldsymbol{\theta}}(\cdot|\overleftarrow{x})$, respectively, and the MMD becomes dependent on a particular past $\overleftarrow{x}$, which we denote as $MMD^2[F, P, \hat{P}_{\boldsymbol{\theta}}|\overleftarrow{x}]$. Inserting this into the cost function yields

$$C(\boldsymbol{\theta}) = \sum_{\overleftarrow{x}} w_{\overleftarrow{x}}\, MMD^2[F, P, \hat{P}_{\boldsymbol{\theta}}|\overleftarrow{x}] + cR_{\overleftarrow{x}}(\boldsymbol{\theta}). \tag{128}$$

The term $R_{\overleftarrow{x}}(\boldsymbol{\theta})$ will be calculated based on a quantum post-processing step, which is investigated in detail in Section 3.3.4. For now, we continue with the classical pre-processing step and how the weights $w_{\overleftarrow{x}}$ are calculated.

3.3.3 Classical Pre-Processing

Since it is infeasible to load the entire input sample $x_{1:L} \in A^L$ into the quantum circuit at once, a classical pre-processing step is needed. This step is only applied once in the initialization phase of the algorithm. Note that without loss of generality, we assume that $A \subset \mathbb{N}_0$. Thus, the input sample is an *int-string*, i.e., a string consisting of integers. The goal of the pre-processing step is to split the input sample into pairs of sequences $\{(\overleftarrow{x}, \overrightarrow{x})\}$, and to calculate the weights $w_{\overleftarrow{x}}$. This is done as follows. First, we choose lengths $l_p, l_f \in \mathbb{N}$ for past and future sequences, respectively. Next, we partition the int-string $x_{1:L}$ into sequences of length $l_p + l_f$, where the first l_p integers are interpreted as the past $\overleftarrow{x}$ and the remaining l_f integers are interpreted as the future $\overrightarrow{x}$. This interpretation can be done since the underlying stochastic process is stationary, i.e., $P(X_{t+\tau}, \ldots X_{t+\tau+n}) = P(X_t, \ldots, P(X_{t+n})$ for all $n, \tau \in \mathbb{N}$. Afterward, these sequences are grouped if their pasts coincide. This yields a training data set, where for each unique observed past $\overleftarrow{x}$ of length l_p, a set of future outcomes $\{\overrightarrow{x}\}$, each of length l_f, can be accessed. The future outcomes $\{\overrightarrow{x}\}$ are not unique. On the contrary, the number of possible futures offers a way to indicate which pasts occur often in the input sample and thus have a greater impact on the performance of the model. Hence, the last step is to count the number of futures per past, denoted as $f_{\overleftarrow{x}}$, and divide it by the total number of futures, $\frac{L}{l_f+l_p}$, to calculate the weights, i.e.,

$$w_{\overleftarrow{x}} = \frac{f_{\overleftarrow{x}}}{\frac{L}{l_f+l_p}} = f_{\overleftarrow{x}} \frac{l_f + l_p}{L}. \tag{129}$$

Due to the construction of the weights, we have $0 \leq w_{\overleftarrow{x}} \leq 1$ for all $\overleftarrow{x}$, $\sum_{\overleftarrow{x}} w_{\overleftarrow{x}} = 1$ and $w_{\overleftarrow{x}}$ is large if $\overleftarrow{x}$ occurs often in the input sample. Besides calculating the weights, the training data sets are also used to perform the two-sample test, i.e., to calculate the MMD dependent on a particular past. The classical pre-processing step additionally introduces the hyper-parameters l_p and l_f, which, in general, can be optimized by validating the learned model with different choices of l_p and l_f and taking the values that performed best. However, the length of the past samples should be at least the Markov order of the underlying stochastic process to ensure that all memory states can be learned in principle. Moreover, we choose $l_f = 1$ within this work, since learning to accurately sample one future time step should be sufficient as long as valid memory states are present. How this can be quantified and calculated with a quantum post-processing step is investigated in the next section.

3.3.4 Quantum Post-Processing

In general, the regularization term $R(\boldsymbol{\theta})$ should be small if the unitary $T(\boldsymbol{\theta})$ outputs memory states, and it should be large if this is not the case. Normally, a quantum register has to be prepared and measured several times to reconstruct the underlying state via quantum state tomography. However, we are not interested in the exact representation of the state, only whether it is a valid memory state. We therefore take a shortcut and measure how close the output of $T(\boldsymbol{\theta})$ is to the corresponding memory state as determined by the encoding unit E. This can be done based on the following observations. One the one hand, we know how to create valid memory states $|\sigma_{\overleftarrow{x}}\rangle$, which is simply applying the data encoding unit $E(\overleftarrow{x},\boldsymbol{\theta})$ for the corresponding past $\overleftarrow{x}$. On the other hand, we also know the memory states that the unitary $T(\boldsymbol{\theta})$ should output, which is $|\sigma_{\overleftarrow{x},x^*}\rangle$ for the case that the auxiliary register is measured to be x^*. Therefore, we can apply the appropriate inversion of the data encoding onto the memory register M_q which yields the $|0\rangle$ state if and only if M_q was in the correct memory state before, i.e.,

$$E^{-1}(\overleftarrow{x},x^*,\boldsymbol{\theta})\,|\sigma_{\overleftarrow{x},x^*}\rangle = |0\rangle\,. \tag{130}$$

Measuring the memory register M_q in the computational basis then yields an approximation for the probability of $T(\boldsymbol{\theta})$ outputting correct memory states. We denote this probability as $P_{\boldsymbol{\theta}}(M_q = 1)$, since we want to minimize this term, and identify it as the missing part $R_{\overleftarrow{x}}(\boldsymbol{\theta})$ in the cost function. Note that the probability distributions $\hat{P}_{\boldsymbol{\theta}}$ and $P_{\boldsymbol{\theta}}$ are not the same. The first one refers to the model's output distribution, i.e., the distribution for generating new samples. This is implicitly defined by measuring the auxiliary register A_q, whereas $P_{\boldsymbol{\theta}}$ is defined by measuring the memory register M_q.

This approach introduces the overhead of applying additional gates during training, which can be seen as a quantum post-processing step. Applying the correct inverse data encoding unit $E^{-1}(\overleftarrow{x},x^*,\boldsymbol{\theta})$ generally requires knowing the measured outcome x^* before the circuit is executed. However, this is not possible in the quantum setting. Therefore, we propose two ways to overcome this issue. The first one relies on performing a fixed inverse data encoding, irrespective of the measurement outcome of the auxiliary register, but post-selecting the events

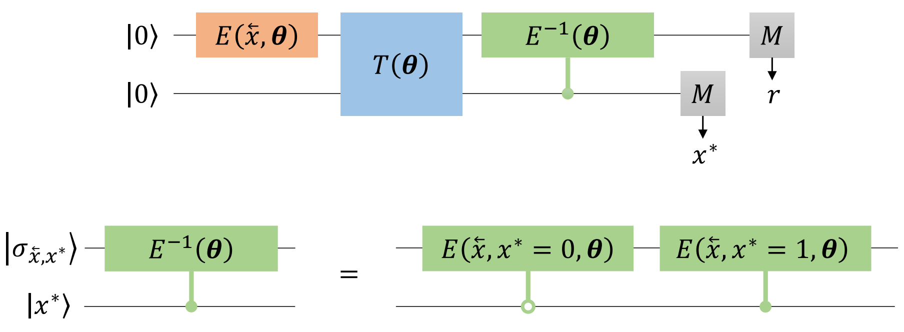
$|0\rangle$
$E(\overleftarrow{x}, \boldsymbol{\theta})$
$T(\boldsymbol{\theta})$
$E^{-1}(\boldsymbol{\theta})$
M
r
$|0\rangle$
M
x^*
$|\sigma_{\overleftarrow{x},x^*}\rangle$
$E^{-1}(\boldsymbol{\theta})$
$|x^*\rangle$
$=$
$E(\overleftarrow{x}, x^* = 0, \boldsymbol{\theta})$
$E(\overleftarrow{x}, x^* = 1, \boldsymbol{\theta})$

Figure 10: Illustration of the quantum post-processing step. The quantum circuit uses a single qubit as memory and auxiliary register and thus $x^* \in \{0, 1\}$. After the application of the data encoding unit $E(\overleftarrow{x}, \boldsymbol{\theta})$ and the unitary $T(\boldsymbol{\theta})$, the inverse of the data encoding unit $E^{-1}(\boldsymbol{\theta})$ is performed. This quantum post-processing step consists of a $|0\rangle$-controlled application of the inverse data encoding for an observation of $(\overleftarrow{x}, x^* = 0)$ and a controlled application of the data encoding for $(\overleftarrow{x}, x^* = 1)$, respectively.

that correspond to a valid run. To be more concrete, assume we have a stochastic process that uses the binary alphabet $A = \{0, 1\}$, e.g., the period-N uniform renewal process. Thus, the auxiliary register A_q is simply one qubit, and a measurement of A_q yields either 0 or 1. Moreover, assume we run the quantum circuit n times for one specific past $\overleftarrow{x}$ to collect n future trajectories $\overrightarrow{y}$. Given the fixed past $\overleftarrow{x}$, we apply $E^{-1}(\overleftarrow{x}, x^* = 0, \boldsymbol{\theta})$ for the first half of the runs, i.e., we assume the measurement outcome to be $\overrightarrow{y} = 0$. Then, we count how often we have measured the memory register to be in the $|0\rangle$ state, but we do not count a run if the generated output was $\overrightarrow{y} = 1$ instead. For the other half, we do the same, but apply $E^{-1}(\overleftarrow{x}, x^* = 1, \boldsymbol{\theta})$ and do not count a run if the generated output was $\overrightarrow{y} = 0$, respectively.
By contrast, the other way is to perform the inverse data encoding for all possible outcomes, but let this operation be controlled by the auxiliary register. We refer to this approach as controlled inverse data encoding. Thus, the state of A_q determines which inverse data encoding is applied. This way is illustrated in Figure 10, where the empty node is meant as a $|0\rangle$-controlled operation. If A_q is in state $|x^*\rangle = |0\rangle$, the first operator is applied, i.e., $E^{-1}(\overleftarrow{x}, x^* = 0, \boldsymbol{\theta})$ that corresponds to a measurement of $x^* = 0$. However, if $|x^*\rangle = |1\rangle$, the first operator is omitted and just the second is applied, which corresponds to $x^* = 1$.
In principle, the post-selection, as well as the controlled inverse data encoding, can be used to calculate $R_{\overleftarrow{x}}(\boldsymbol{\theta})$, with each having its advantages and disadvantages. On the one hand, post-selection leads to a fractional loss in the approximation of the probability distribution. On the other hand, the circuit is relatively shallow, since only one operation is applied. By contrast, the depth of the circuit for the controlled approach increases with the size of the alphabet, since all possible outcomes need to be taken into account. However, each run of the circuit contributes to the approximation of the desired probability distribution. Throughout this work, we use the latter approach, since the period-N uniform renewal processes use a binary alphabet and thus the resulting circuit is relatively shallow as well.
How the inverse data encodings look in detail is dependent on the ansatz of the VQC. Thus, the ansatz that we will use within this work is explained next.

3.4 Quantum Circuit Ansatz

While the general structure of the VQC was explained in Section 3.3.1, it remains open which gates should be used to construct the unitaries E, T and E^{-1}, respectively. The goal of this section is therefore to propose a fixed ansatz for the VQC, which offers trainable parameters $\boldsymbol{\theta}$. In general, different ansätze have been proposed in the literature, as explained in Section 2.2.4. In this work, we take the approach of starting with a known solution for our model problem and gradually decomposing the involving gates into rotation and *CX* gates to construct an ansatz. To be more precise, in the following we construct a quantum circuit that corresponds to the q-simulator for the period-2 uniform renewal process, i.e., a quantum exact predictive model. The gates used for this circuit are static, which means that they do not offer trainable parameters. The next step then is to derive decompositions for these gates, such that the circuit can be gradually decomposed into a parameterized quantum circuit. This PQC offers the same structure as the problem-inspired ansatz for supervised machine learning from Section 2.2.4, i.e., alternating blocks of encoding and entangling layers. This approach has the advantage that we know the optimal parameter set for our model problem and can thus use this information to validate the learned models. Note that we follow this approach only to validate our models, such that we can show the proposed learning algorithm works as expected. Deriving an ansatz from a known solution is, in general, not necessary and even not possible since the true underlying stochastic process is a priori unknown.

3.4.1 Construction of a q-simulator

The period-2 uniform renewal process has two causal states S_0 and S_1, where past observations ending with 1 are mapped to S_0 and observations ending with 0 are mapped to S_1, respectively, cf. Figure 8. We therefore want to have two memory states $|\sigma_0\rangle$ and $|\sigma_1\rangle$ that correspond to the two causal states. By contrast to the ϵ-machine, we would like to have the mapping $\overleftarrow{x} = 0 \to |\sigma_0\rangle$ and $\overleftarrow{x} = 1 \to |\sigma_1\rangle$, i.e., we swap the indices of the memory states. This makes the following calculations a bit easier to read since a memory state $|\sigma_i\rangle$ can be directly associated with the mapped observation $i \in A$. These two states together define the data encoding unit $E(\overleftarrow{x})$ according to

$$E(x_0 = 0)\,|0\rangle = |\sigma_0\rangle\,, \tag{131}$$

$$E(x_0 = 1)\,|0\rangle = |\sigma_1\rangle\,. \tag{132}$$

The unitary T will act on these memory states together with the auxiliary register, which is initially in state $|0\rangle$. Applying T once should lead to

$$T\,|\sigma_i\rangle\,|0\rangle = \sum_{x\in A} \sqrt{P(x|i)}\,|\sigma_x\rangle\,|x\rangle\,, \tag{133}$$

which is the formula of the q-simulator, cf. Equation (56). For a system prepared in this state, the second qubit will be measured to be $x \in A$ with probability $P(x|i)$, while the first qubit collapses to $|\sigma_x\rangle$. If we insert the values for the conditional probability distribution of the period-2 uniform renewal process (see Table 1), we get

$$T\,|\sigma_0\rangle\,|0\rangle = \sqrt{P(0|0)}\,|\sigma_0\rangle\,|0\rangle + \sqrt{P(1|0)}\,|\sigma_1\rangle\,|1\rangle \tag{134}$$

$$= |\sigma_1\rangle\,|1\rangle \tag{135}$$

and

$$T\,|\sigma_1\rangle\,|0\rangle = \sqrt{P(0|1)}\,|\sigma_0\rangle\,|0\rangle + \sqrt{P(1|1)}\,|\sigma_1\rangle\,|1\rangle \tag{136}$$

$$= \frac{1}{\sqrt{2}}\,|\sigma_0\rangle\,|0\rangle + \frac{1}{\sqrt{2}}\,|\sigma_1\rangle\,|1\rangle\,. \tag{137}$$

The question now is how to construct the two unitaries E and T. A natural approach would be to define the memory states $|\sigma_i\rangle$ and to derive the structure of E and T from there. However, the memory states cannot be chosen arbitrarily, which is reflected in the following lemma.

Lemma 3.8. The orthogonal states $|0\rangle$ and $|1\rangle$ can not be the memory states for the period-2 uniform renewal process.

Proof. Assume the mapping $0 \to |\sigma_0\rangle = |0\rangle$ and $1 \to |\sigma_1\rangle = |1\rangle$. Thus, the unitary operator T needs to satisfy

$$T|00\rangle = |11\rangle, \tag{138}$$

$$T|10\rangle = \frac{1}{\sqrt{2}}\Big(|00\rangle + |11\rangle\Big). \tag{139}$$

Writing T as a 4×4 matrix, the first two columns are determined by the two equations and can be calculated as

$$\boldsymbol{T} = \begin{bmatrix} 0 & \frac{1}{\sqrt{2}} & \# & \# \\ 0 & 0 & \# & \# \\ 0 & 0 & \# & \# \\ 1 & \frac{1}{\sqrt{2}} & \# & \# \end{bmatrix} \tag{140}$$

in the computational basis. Since these two columns are not orthogonal, the matrix can never be filled out to become unitary. The reversed mapping $0 \to |1\rangle$ and $1 \to |0\rangle$ can be done analogously, which concludes the proof. □

This result shows that the memory states must be chosen with caution. We therefore take a closer look at the ϵ-machine and see if we can find a better candidate for the memory states. If the ϵ-machine is in state S_1, i.e., a 0 was observed last, the next state will be S_0 with confidence. On the other side, if the ϵ-machine is in state S_0, the next state will be either S_0 or S_1 with equal probability. The quantum memory states can be chosen to reflect the same behavior. Thus, one way is to fix one state $|\sigma_i\rangle = |i\rangle$ for one $i \in \{0,1\}$, and define the other as an equal superposition of the first one. Following this, one possible choice is $0 \to |\sigma_0\rangle = |1\rangle$ and $1 \to |\sigma_1\rangle = |+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, which turns out to be a valid set of memory states. To prove this, we first derive the corresponding unitary T. For this set of memory states, the action of T onto the memory states needs to be

$$\begin{aligned} T|\sigma_0\rangle|0\rangle &= T|10\rangle = |{+}1\rangle, \\ T|\sigma_1\rangle|0\rangle &= T|{+}0\rangle = \frac{1}{\sqrt{2}}|10\rangle + \frac{1}{\sqrt{2}}|{+}1\rangle. \end{aligned} \tag{141}$$

Let us first focus on the action onto $|\sigma_0\rangle$. Here, T essentially changes the first qubit from state $|1\rangle$ to $|+\rangle$ and the second from $|0\rangle$ to $|1\rangle$. This can be done by applying a Hadamard gate onto the second qubit, controlled by the first one, and swapping the qubits afterward. More formally, this can be written as

$$|\sigma_0\rangle|0\rangle = |10\rangle \tag{142}$$

$$\xrightarrow{CH} |1+\rangle \tag{143}$$

$$= \frac{1}{\sqrt{2}}\Big(|10\rangle + |11\rangle\Big) \tag{144}$$

$$\xrightarrow{SWAP} \frac{1}{\sqrt{2}}\Big(|01\rangle + |11\rangle\Big) \tag{145}$$

$$= |{+}1\rangle, \tag{146}$$

and the corresponding unitary would be $T' = SWAP \cdot CH$. We write T' because this is not yet the final form of the operator. This can be seen by applying it onto $|\sigma_1\rangle$, i.e.,

$$|\sigma_1\rangle|0\rangle = |{+}0\rangle \tag{147}$$

$$= \frac{1}{\sqrt{2}}\Big(|00\rangle + |10\rangle\Big) \tag{148}$$

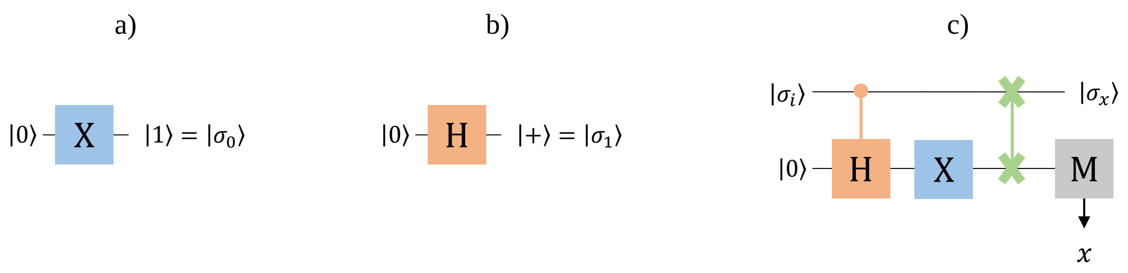

Figure 11: Components of the q-simulator for the period-2 uniform renewal process. a) The data encoding unit for $|\sigma_0\rangle$ which is the Pauli-X gate. b) Applying the Hadamard gate creates the second memory state $|+\rangle$. c) The circuit of the q-simulator that acts on $|\sigma_i\rangle \otimes |0\rangle$ and consists of a controlled Hadamard, a Pauli-X and a $SWAP$ gate.

$$\xrightarrow{CH} \frac{1}{\sqrt{2}}\Big(|00\rangle + |1+\rangle\Big) \tag{149}$$

$$= \frac{1}{\sqrt{2}}\Big(|00\rangle + \frac{1}{\sqrt{2}}\Big(|10\rangle + |11\rangle\Big)\Big) \tag{150}$$

$$= \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle \tag{151}$$

$$\xrightarrow{SWAP} \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|11\rangle \tag{152}$$

$$= \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|+1\rangle\,, \tag{153}$$

which includes $\frac{1}{\sqrt{2}}|00\rangle$ instead of $\frac{1}{\sqrt{2}}|10\rangle$, cf. Equation (141). However, this can be fixed by simply applying an X gate onto the first qubit,

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|+1\rangle \xrightarrow{X\otimes I} \frac{1}{\sqrt{2}}|10\rangle + \frac{1}{\sqrt{2}}|+1\rangle\,, \tag{154}$$

which works since $X|+\rangle = |+\rangle$. Due to the same reason, the additional gate does not change the effect onto the memory state $|\sigma_0\rangle = |+\rangle$. There is no difference if we swap the two qubits and apply the X gate onto the first one, or apply X to the second qubit first and swap afterward. The latter approach, however, has the advantage that the decomposition of the circuit has fewer gates, which is explained in the next section. Thus, we conclude that the operator T can be written as

$$T = SWAP \cdot (I \otimes X) \cdot CH. \tag{155}$$

The quantum circuit corresponding to the unitary T is presented in Figure 11 c). What is still missing is the data encoding unit E. However, the two memory states can be easily prepared with either an X or a Hadamard gate:

$$X|0\rangle = |1\rangle = |\sigma_0\rangle\,, \tag{156}$$

$$H|0\rangle = |+\rangle = |\sigma_1\rangle\,. \tag{157}$$

Thus, if we want to start the simulation based on a past observation ending with $\overleftarrow{x} = 0$, we apply the X gate first to the memory qubit and perform the unitary T afterward. Similarly, we can use the H gate to start the simulation based on past observations ending with $\overleftarrow{x} = 1$. The data encoding units are illustrated in Figure 11 a) and b), respectively.

While we have successfully constructed a q-simulator for the period-2 uniform renewal process, the resulting circuit is static and can only be used to simulate this particular process. Thus, we gradually decompose these static gates into rotation and CX gates in the following.

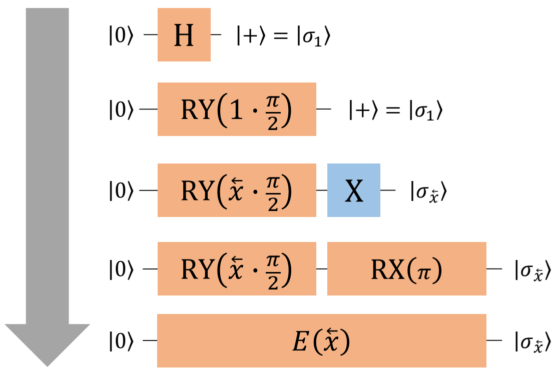
$|0\rangle$
H
$|+\rangle = |\sigma_1\rangle$
$|0\rangle$
$\mathrm{RY}(1 \cdot \frac{\pi}{2})$
$|+\rangle = |\sigma_1\rangle$
$|0\rangle$
$\mathrm{RY}(\overleftarrow{x} \cdot \frac{\pi}{2})$
X
$|\sigma_{\overleftarrow{x}}\rangle$
$|0\rangle$
$\mathrm{RY}(\overleftarrow{x} \cdot \frac{\pi}{2})$
$\mathrm{RX}(\pi)$
$|\sigma_{\overleftarrow{x}}\rangle$
$|0\rangle$
$E(\overleftarrow{x})$
$|\sigma_{\overleftarrow{x}}\rangle$

Figure 12: Illustration of the construction of the data encoding unit for the q-simulator. The different steps are read from top to bottom. First, the Hadamard gate is replaced by an RY gate, such that the observation of $\overleftarrow{x} = 1$ can be encoded via $RY(\overleftarrow{x} \cdot \pi/2)$. A Pauli-$X$ gate is then added such that past observations $\overleftarrow{x} = 0$ can also be encoded. Lastly, the X gate is replaced by $RX(\pi)$, leading to the data encoding unit $E(\overleftarrow{x})$.

3.4.2 Quantum Circuit Decomposition

We start the decomposition of the circuit with the data encoding unit $E(\overleftarrow{x})$. For the q-simulator, depending on the past observation, either the Pauli-X or Hadamard gate is applied, leading to $|\sigma_0\rangle = X|0\rangle = |1\rangle$ or $|\sigma_1\rangle = H|0\rangle = |+\rangle$, respectively. However, we are aiming for a fixed ansatz, i.e., a fixed set of gates, which does not need to be changed for encoding different past observations. Thus, we are looking for a sequence of rotation gates that can encode $0 \to |1\rangle$ and $1 \to |+\rangle$, while acting on the initial state $|0\rangle$. The procedure to construct such a circuit is illustrated in Figure 12 and will be explained step by step in the following.
We start with preparing the $|\sigma_1\rangle = |+\rangle$ state. Here, an RY gate can be used with a rotation angle of $\frac{\pi}{2}$, i.e.,

$$RY\left(\frac{\pi}{2}\right)|0\rangle = \cos\frac{\pi}{4}|0\rangle + \sin\frac{\pi}{4}|1\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle = |\sigma_1\rangle\,. \tag{158}$$

Therefore, we can encode an observation of $\overleftarrow{x} = 1$ by multiplying this value with the rotation angle $\frac{\pi}{2}$ leading to

$$E'(\overleftarrow{x}) = RY\left(\overleftarrow{x} \cdot \frac{\pi}{2}\right). \tag{159}$$

We write E' since this is not yet the final form of the decomposition. This can be seen via performing the encoding of an observation $\overleftarrow{x} = 0$:

$$E'(\overleftarrow{x} = 0)\,|0\rangle = RY\left(0 \cdot \frac{\pi}{2}\right)|0\rangle = |0\rangle \;\neq\; |1\rangle = |\sigma_0\rangle\,. \tag{160}$$

However, we can perform a Pauli-X operation afterward in order to transform the $|0\rangle$ state to $|1\rangle$. Thus,

$$E''(\overleftarrow{x} = 0)\,|0\rangle = X \cdot RY\left(0 \cdot \frac{\pi}{2}\right)|0\rangle = X|0\rangle = |1\rangle = |\sigma_0\rangle\,, \tag{161}$$

which does not change the encoding of $\overleftarrow{x} = 1$ since $X|+\rangle = |+\rangle = |\sigma_1\rangle$. However, we have now introduced the Pauli-X gate which is not parameterized. Yet, the Pauli gates can easily be written as rotation gates as follows.

Lemma 3.9. The Pauli gates $\{X, Y, Z\}$ can be written as rotation gates according to

$$X = RX(\pi), \quad Y = RY(\pi), \quad Z = RZ(\pi), \tag{162}$$

up to an unobservable global phase.

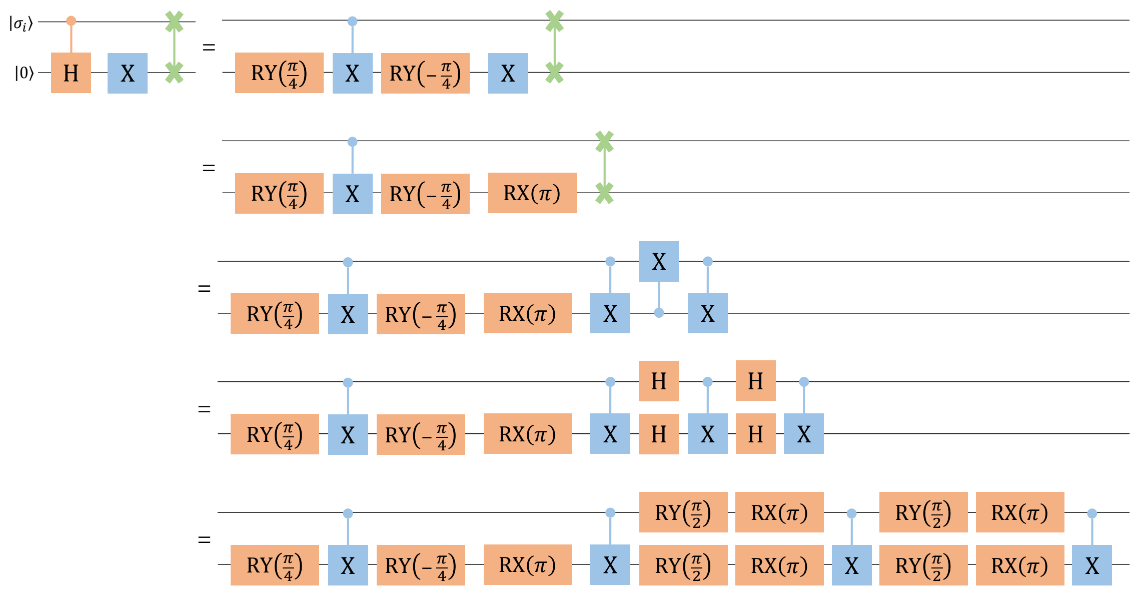

Figure 13: Decomposition of the unitary T of the q-simulator. The first step is to rewrite the controlled Hadamard gate as two rotation gates; *RY* and a *CX* gate. Next, the Pauli-X gate is replaced by its corresponding rotation gate *RX*. The last step is to decompose the *SWAP* gate. This is done in three steps, where the gate is first decomposed to three *CX* gates, where one gate is upside down. A transformation to the Hadamard base is applied to swap the upside-down gate and the Hadamard gates are subsequently rewritten as a sequence of two rotation gates *RX*· *RY*.

Proof. Using the definition of the rotation gates from Section 2.2.2 we get

$$RX(\pi) = \exp\left(-i\frac{\pi}{2}X\right) = \begin{bmatrix} \cos\pi/2 & -i\sin\pi/2 \\ -i\sin\pi/2 & \cos\pi/2 \end{bmatrix} = \begin{bmatrix} 0 & -i \\ -i & 0 \end{bmatrix} = -iX, \tag{163}$$

$$RY(\pi) = \exp\left(-i\frac{\pi}{2}Y\right) = \begin{bmatrix} \cos\pi/2 & -\sin\pi/2 \\ \sin\pi/2 & \cos\pi/2 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} = -iY, \tag{164}$$

$$RZ(\pi) = \exp\left(-i\frac{\pi}{2}Z\right) = \begin{bmatrix} e^{-i\pi/2} & 0 \\ 0 & e^{i\pi/2} \end{bmatrix} = \begin{bmatrix} -i & 0 \\ 0 & i \end{bmatrix} = -iZ. \tag{165}$$

□

Thus, we simply replace X with $RX(\pi)$ and get the data encoding unit E that only depends on rotation gates:

Theorem 3.1. The data encoding unit for the q-simulator of the period-2 uniform renewal process can be written as

$$E(\overleftarrow{x}) = RX(\pi)\cdot RY\left(\overleftarrow{x}\cdot\frac{\pi}{2}\right), \tag{166}$$

with $\overleftarrow{x} \in A$.

The next step is to decompose the unitary $T = SWAP\cdot(I\otimes X)\cdot CH$. The different steps are illustrated in Figure 13. We start with the controlled Hadamard gate, which can be decomposed into rotation gates as follows.

Lemma 3.10. The controlled Hadamard gate can be decomposed into

$$CH = \left(I\otimes RY\left(-\frac{\pi}{4}\right)\right)\cdot CX\cdot\left(I\otimes RY\left(\frac{\pi}{4}\right)\right). \tag{167}$$

Proof. In order to prove this decomposition, we analyze the effect onto the target qubit for both cases, the control qubit being in state $|0\rangle$ and $|1\rangle$. On the one hand, if the control qubit is in state $|0\rangle$, no X gate is applied onto the target qubit and the two rotation gates thus cancel out, leading to no effect. On the other hand, if the control qubit is in state $|1\rangle$, a Pauli-X gate is applied between the two rotation gates and we perform

$$RY\Big(-\frac{\pi}{4}\Big)\cdot X\cdot RY\Big(\frac{\pi}{4}\Big) \tag{168}$$

onto the target qubit. In order to finish the proof, we show that this sequence of gates acts as the Hadamard gate onto the computational basis. Thus, we get

$$\begin{aligned}
|0\rangle \xrightarrow{RY(\pi/4)} & \cos\frac{\pi}{8}|0\rangle+\sin\frac{\pi}{8}|1\rangle && (169)\\
\xrightarrow{X} & \sin\frac{\pi}{8}|0\rangle+\cos\frac{\pi}{8}|1\rangle && (170)\\
\xrightarrow{RY(-\pi/4)} & \sin\frac{\pi}{8}\Big(\cos-\frac{\pi}{8}|0\rangle+\sin-\frac{\pi}{8}|1\rangle\Big)+\cos\frac{\pi}{8}\Big(-\sin-\frac{\pi}{8}|0\rangle+\cos-\frac{\pi}{8}|1\rangle\Big) && (171)\\
=&\Big(\sin\frac{\pi}{8}\cos-\frac{\pi}{8}-\cos\frac{\pi}{8}\sin-\frac{\pi}{8}\Big)|0\rangle+\Big(\sin\frac{\pi}{8}\sin-\frac{\pi}{8}+\cos\frac{\pi}{8}\cos-\frac{\pi}{8}\Big)|1\rangle && (172)\\
=&\sin\Big(\frac{\pi}{8}-\Big(-\frac{\pi}{8}\Big)\Big)|0\rangle+\cos\Big(\frac{\pi}{8}-\Big(-\frac{\pi}{8}\Big)\Big)|1\rangle && (173)\\
=&\sin\frac{\pi}{4}|0\rangle+\cos\frac{\pi}{4}|1\rangle && (174)\\
=&\frac{|0\rangle+|1\rangle}{\sqrt{2}} && (175)\\
=&|+\rangle\,, && (176)
\end{aligned}$$

with using the trigonometric identities

$$\sin(x-y)=\sin(x)\cos(y)-\cos(x)\sin(y), \tag{177}$$
$$\cos(x-y)=\sin(x)\sin(y)+\cos(x)\cos(y). \tag{178}$$

Analogously we can calculate that

$$RY\Big(-\frac{\pi}{4}\Big)\cdot X\cdot RY\Big(\frac{\pi}{4}\Big)|1\rangle=|-\rangle\,. \tag{179}$$

which concludes the proof. □

We can use this decomposition for the unitary T and rewrite the controlled Hadamard gate as a sequence containing two rotation gates and one CX gate, see Figure 13. The next gate is the Pauli-X gate, for which we know already that it can be written as RX gate, cf. Lemma 3.9. Lastly, the $SWAP$ gate needs to be decomposed. We do that step-wise and decompose the gate first into a sequence of CX gates.

Lemma 3.11. The $SWAP$ gate can be expressed with only CX gates as

$$SWAP=CX_{c,t}\cdot CX_{t,c}\cdot CX_{c,t}, \tag{180}$$

with the subscript c,t indicating that the first qubit is the control and the second the target qubit.

Proof. Applying the sequence of CX gates onto the basis vectors $\{|00\rangle,|01\rangle,|10\rangle,|11\rangle\}$ yields

$$\begin{aligned}
CX_{c,t}\cdot CX_{t,c}\cdot CX_{c,t}|00\rangle &= |00\rangle\,, && (181)\\
CX_{c,t}\cdot CX_{t,c}\cdot CX_{c,t}|01\rangle &= CX_{c,t}\cdot CX_{t,c}|01\rangle=CX_{c,t}|11\rangle=|10\rangle\,, && (182)\\
CX_{c,t}\cdot CX_{t,c}\cdot CX_{c,t}|10\rangle &= CX_{c,t}\cdot CX_{t,c}|11\rangle=CX_{c,t}|01\rangle=|01\rangle\,, && (183)\\
CX_{c,t}\cdot CX_{t,c}\cdot CX_{c,t}|11\rangle &= CX_{c,t}\cdot CX_{t,c}|10\rangle=CX_{c,t}|10\rangle=|11\rangle\,, && (184)
\end{aligned}$$

i.e., the qubits have been swapped. □

While the circuit for T now only contains rotation and CX gates, we are still not finished yet. This is because we would like to have a quantum circuit that consists of alternating encoding and entangling blocks. However, one CX gate has a wrong orientation, i.e., the control and target qubits are swapped, see Figure 13. Yet, we can change this orientation with the use of Hadamard gates as follows.

Lemma 3.12. The position of the control and target qubit of the CX gate can be swapped via

$$CX_{t,c} = H^{\otimes 2} \cdot CX_{c,t} \cdot H^{\otimes 2}, \tag{185}$$

with $H^{\otimes 2} = H \otimes H$.

Proof. Applying the $CX_{c,t}$ gate onto the states of the Hadamard basis $\{|{+}{+}\rangle, |{+}{-}\rangle, |{-}{+}\rangle, |{-}{-}\rangle\}$ with $|+\rangle = H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|-\rangle = H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ yields

$$CX_{c,t}|{+}{+}\rangle = CX_{c,t}\,\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) = \frac{1}{2}(|00\rangle + |01\rangle + |11\rangle + |10\rangle) = |{+}{+}\rangle, \tag{186}$$

$$CX_{c,t}|{+}{-}\rangle = CX_{c,t}\,\frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle) = \frac{1}{2}(|00\rangle - |01\rangle + |11\rangle - |10\rangle) = |{-}{-}\rangle, \tag{187}$$

$$CX_{c,t}|{-}{+}\rangle = CX_{c,t}\,\frac{1}{2}(|00\rangle + |01\rangle - |10\rangle - |11\rangle) = \frac{1}{2}(|00\rangle + |01\rangle - |11\rangle - |10\rangle) = |{-}{+}\rangle, \tag{188}$$

$$CX_{c,t}|{-}{-}\rangle = CX_{c,t}\,\frac{1}{2}(|00\rangle - |01\rangle - |10\rangle + |11\rangle) = \frac{1}{2}(|00\rangle - |01\rangle - |11\rangle + |10\rangle) = |{+}{-}\rangle. \tag{189}$$

Therefore, in the Hadamard basis, the $CX_{c,t}$ gate acts as the $CX_{t,c}$ gate, but onto $|+\rangle$ instead of $|0\rangle$ and $|-\rangle$ instead of $|1\rangle$. Thus, one can simply change the basis before and after applying the $CX_{c,t}$ gate and get the effect of the $CX_{t,c}$ gate, i.e.,

$$H^{\otimes 2} \cdot CX_{c,t} \cdot H^{\otimes 2}|00\rangle = H^{\otimes 2} \cdot CX_{c,t}|{+}{+}\rangle = H^{\otimes 2}|{+}{+}\rangle = |00\rangle, \tag{190}$$

$$H^{\otimes 2} \cdot CX_{c,t} \cdot H^{\otimes 2}|01\rangle = H^{\otimes 2} \cdot CX_{c,t}|{+}{-}\rangle = H^{\otimes 2}|{-}{-}\rangle = |11\rangle, \tag{191}$$

$$H^{\otimes 2} \cdot CX_{c,t} \cdot H^{\otimes 2}|10\rangle = H^{\otimes 2} \cdot CX_{c,t}|{-}{+}\rangle = H^{\otimes 2}|{-}{+}\rangle = |10\rangle, \tag{192}$$

$$H^{\otimes 2} \cdot CX_{c,t} \cdot H^{\otimes 2}|11\rangle = H^{\otimes 2} \cdot CX_{c,t}|{-}{-}\rangle = H^{\otimes 2}|{+}{-}\rangle = |01\rangle, \tag{193}$$

and $H^{\otimes 2} \cdot CX_{c,t} \cdot H^{\otimes 2}$ can be identified as $CX_{t,c}$. □

After applying this decomposition onto the unitary T, all the CX gates within our circuit have the correct orientation. However, in order to achieve this, we have introduced some Hadamard gates. Yet, we can decompose these gates into rotation gates as well.

Lemma 3.13. The Hadamard gate can be written as

$$H = RX(\pi) \cdot RY\left(\frac{\pi}{2}\right), \tag{194}$$

up to a global phase.

Proof. A straightforward calculation yields

$$RX(\pi) \cdot RY\left(\frac{\pi}{2}\right) = \frac{1}{\sqrt{2}}\,RX(\pi)\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \tag{195}$$

$$= \frac{1}{\sqrt{2}} - iX \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \tag{196}$$

$$= \frac{-i}{\sqrt{2}}\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \tag{197}$$

$$= \frac{-i}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{198}$$

$$= -iH, \tag{199}$$

with using $RX(\pi) = -iX$ from Lemma 3.9. □

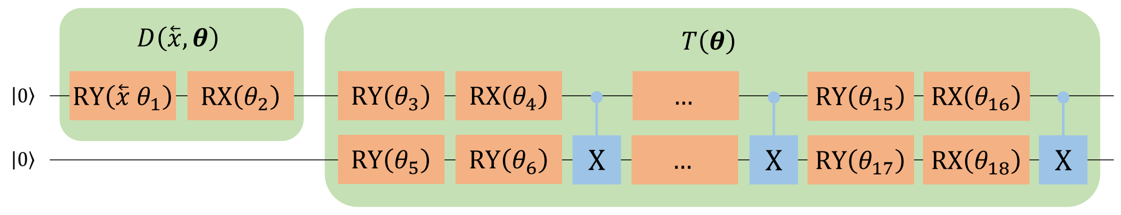

Figure 14: Illustration of the ansatz used for the VQC. The data encoding unit $D(\overleftarrow{x},\boldsymbol{\theta})$ has two parameters, θ_1,θ_2, where the first one is multiplied with the past observation $\overleftarrow{x}$. Four layers of alternating encoding and entangling blocks form the unitary $T(\boldsymbol{\theta})$ that offers 16 trainable parameters $\theta_3,\theta_4,\ldots,\theta_{18}$.

Thus, a Hadamard gate can be written as two rotation gates, which we apply to our quantum circuit for T and get the following final decomposition:

Theorem 3.2. The unitary T of the q-simulator for the period-2 uniform renewal process can be decomposed into the circuit given in Figure 13. This circuit consists of 11 rotation gates from the set $\{RX, RY\}$ and 4 CX gates.

Based on this decomposition, we can now derive the ansatz for the VQC. We want to have a circuit structure that can be written as alternating encoding and entangling layers. Since $RX(0) = RY(0) = I$, we can add the missing rotation gates RX and RY such that the circuit contains 4 encoding blocks, each of which being a sequence $RX \cdot RY$. In total, we have 4 encoding layers, each offering 4 rotation gates, and thus we have 16 rotation angles for T. Together with the two angles from the data encoding unit $E(\overleftarrow{x})$ we have 18 rotation angles in total. Lastly, we replace these fixed angles with free parameters $\boldsymbol{\theta} \in \mathbb{R}^{18}$ and get a parameterized quantum circuit $U(\overleftarrow{x},\boldsymbol{\theta})$. This circuit is illustrated in Figure 14 and will be used as the ansatz within this work. Since the ansatz only contains CX and Pauli rotation gates, gradients can be computed using the parameter-shift rule.
In order to drive the learning towards quantum memory states, we have introduced a quantum post-processing step in Section 3.3.4. This step makes use of the inverse of the data encoding unit. Given the two parameters for the data encoding, we have

$$E(x^*,\theta_1,\theta_2) = RX(\theta_2)\cdot RY(x^*\cdot\theta_1) \tag{200}$$

for $x^* \in \{0,1\}$, and the post-processing step can be written as

$$V(\theta_1,\theta_2) = CE^{-1}(x^*=1,\theta_1,\theta_2)\cdot\widetilde{CE}^{-1}(x^*=0,\theta_1,\theta_2), \tag{201}$$

with $\widetilde{CE}^{-1}$ being the inverse data encoding, but controlled by the $|0\rangle$ state. Note that we deviate from the usual notation here, as the inverse encoding is applied onto the memory (upper) register and controlled by the auxiliary (lower) register, cf. Figure 10. Combining Equations (200) and (201) leads to

$$V(\theta_1,\theta_2) = CRY^{-1}(1\cdot\theta_1)\cdot CRX^{-1}(\theta_2)\cdot\widetilde{CRY}^{-1}(0\cdot\theta_1)\cdot\widetilde{CRX}^{-1}(\theta_2), \tag{202}$$

with using $(A\cdot B)^{-1} = (A\cdot B)^\dagger = B^\dagger\cdot A^\dagger = B^{-1}\cdot A^{-1}$, for two unitaries A and B. Note that we explicitly keep the values $x^*=0$ and $x^*=1$ in Equation (202) to show that the trainable parameters are multiplied with all possible observations x^*. The post-processing step $V(\theta_1,\theta_2)$ will not be used for simulating the stochastic process, but it is part of the circuit during the training. Thus, we need to calculate the gradient of such circuits with respect to the trainable parameters $\boldsymbol{\theta}$. For this purpose, we would like to apply the parameter-shift rule, which is given

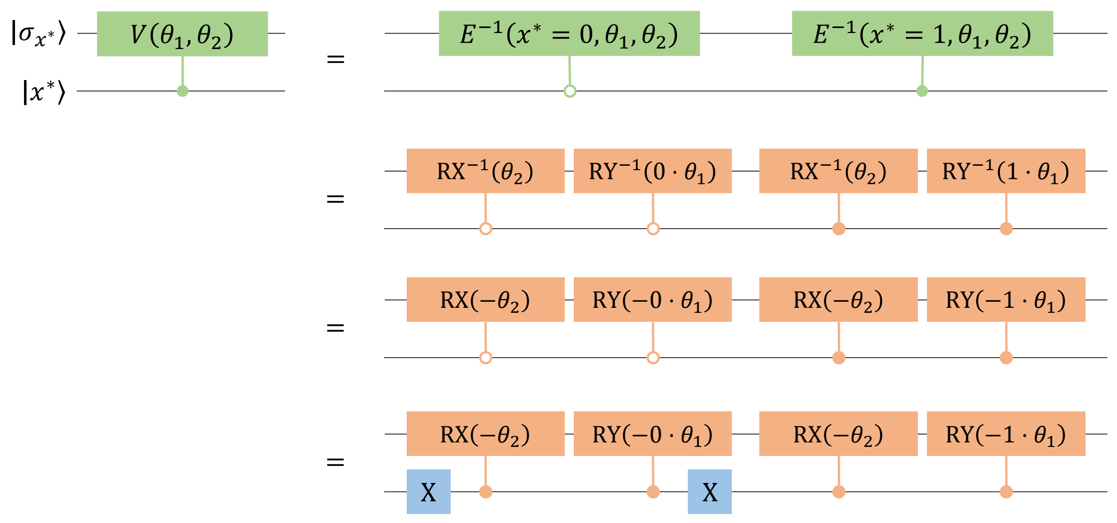

Figure 15: Illustration of the decomposition of the quantum post-processing step. First, the controlled inverse data encoding is decomposed into controlled rotation gates. Second, the inverse rotation gates are replaced by normal rotation gates but rotate in the opposite direction. The last step is to decompose the $|0\rangle$-controlled gates into normal controlled gates with the use of Pauli-X gates before and after the application.

for circuits consisting of Pauli rotation gates. However, it does not work out of the box for their controlled counterparts. Therefore, we also gradually decompose $V(\overleftarrow{x}, \theta_1, \theta_2)$, such that the controlled rotation gates are replaced by simple rotation and CX gates. The different steps are illustrated in Figure 15. We start with replacing the inverse controlled rotation gates with their controlled version via rotating in the opposite direction, e.g., $CRX^{-1}(\theta) = CRX(-\theta)$. The next step is to rewrite the $|0\rangle$-controlled rotation gates, such that they become normal controlled rotation gates. This can be done as follows.

Lemma 3.14. A $|0\rangle$-controlled gate $\widetilde{CU}$ can be written as

$$\widetilde{CU} = (X \otimes I) \cdot CU \cdot (X \otimes I). \tag{203}$$

Proof. A general controlled-U gate can be written as

$$\widetilde{CU} = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes U, \tag{204}$$

cf. Equation (38). Thus, we have

$$\begin{aligned}
(X \otimes I) \cdot CU \cdot (X \otimes I) &= (X \otimes I) \cdot \big(|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes U\big) \cdot (X \otimes I) && (205)\\
&= X|0\rangle\langle 0|X \otimes I + X|1\rangle\langle 1|X \otimes U && (206)\\
&= |1\rangle\langle 1| \otimes I + |0\rangle\langle 0| \otimes U && (207)\\
&= |0\rangle\langle 0| \otimes U + |1\rangle\langle 1| \otimes I && (208)\\
&= \widetilde{CU}. && (209)
\end{aligned}$$

□

Therefore, we can simply add a Pauli-X gate before and after the application of the $|0\rangle$-controlled rotation gates, see Figure 15. The next step now is to decompose the normal controlled rotation gates into rotation and CX gates. This can be done in a way similar to the decomposition of the controlled Hadamard gate, cf. Lemma 3.10.

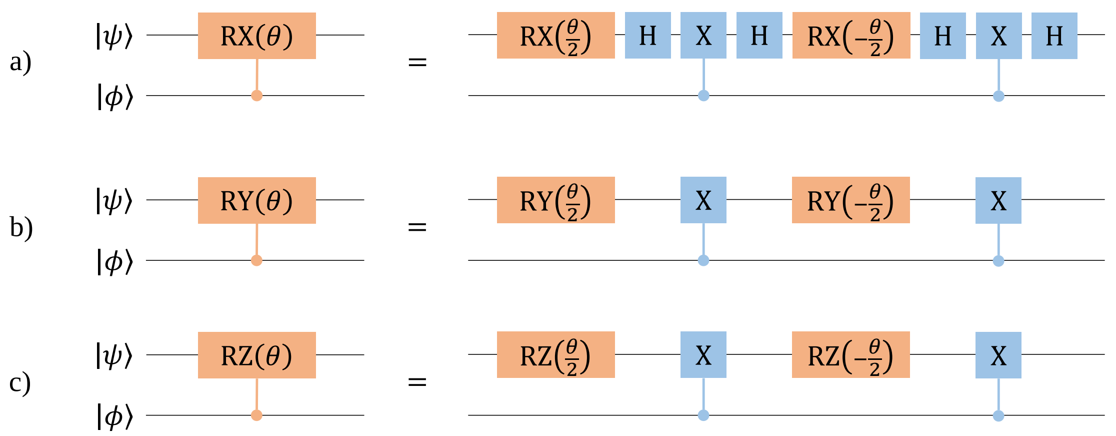

Figure 16: Decomposition of the controlled *RX* gate a), the controlled *RY* gate b), and the controlled *RZ* gate c).

Lemma 3.15. The controlled rotation gates can be decomposed to

$$CRX(\theta) = H_t \cdot CX \cdot H_t \cdot RX_t\left(-\frac{\theta}{2}\right) \cdot H_t \cdot CX \cdot H_t \cdot RX_t\left(\frac{\theta}{2}\right), \tag{210}$$

$$CRY(\theta) = CX \cdot RY_t\left(-\frac{\theta}{2}\right) \cdot CX \cdot RY_t\left(\frac{\theta}{2}\right), \tag{211}$$

$$CRZ(\theta) = CX \cdot RZ_t\left(-\frac{\theta}{2}\right) \cdot CX \cdot RZ_t\left(\frac{\theta}{2}\right), \tag{212}$$

with the subscript t indicating that the gate is applied only onto the target qubit.

Proof. In order to prove these decompositions, we analyze the effect onto the target qubit for both cases, the controlled qubit being in state $|0\rangle$ and $|1\rangle$. First, consider the *CRY* gate. If the controlled qubit is in state $|0\rangle$, the target qubit is rotated $\frac{\theta}{2}$, immediately followed by a rotation of $-\frac{\theta}{2}$, ending up in no effect. With the controlled qubit being in state $|1\rangle$, a Pauli-X gate is applied after the two rotations. A straightforward calculation shows that

$$X \cdot RY\left(-\frac{\theta}{2}\right) \cdot X \ RY\left(\frac{\theta}{2}\right) = RY(\theta), \tag{213}$$

i.e., a single rotation of θ around the Y axis is performed. The same holds true for the controlled *RZ* gate. Consider now the *CRX* gate. For the *RX* gate, Equation 213 does not hold. However, it can be shown that

$$H \cdot X \cdot H \cdot RX\left(-\frac{\theta}{2}\right) \cdot H \cdot X \cdot H \cdot RX\left(\frac{\theta}{2}\right) = RX(\theta),$$

which is applied onto the target qubit if the control qubit is in state $|1\rangle$. If the target qubit is in state $|0\rangle$, the Hadamard gates are applied one after the other. Since $H^2 = I$ we get

$$RX\left(-\frac{\theta}{2}\right) \ RX\left(\frac{\theta}{2}\right) = I,$$

i.e., no effect onto the target qubit. □

The decomposition of the controlled rotation gates are illustrated in Figure 16. Since we make use of the *CRY* gate, we now have additional Hadamard gates within our circuit. Moreover, we inserted Pauli-X gates in order to get rid of the $|0\rangle$-controlled gates. However, these gates do not depend on the trainable parameters $\boldsymbol{\theta}$ and are therefore no obstacle for the parameter-shift rule.

At the end of this section, we want to stress a few aspects of the ansatz. The first one is that we have rotation gates within our circuit where the trainable parameter is multiplied by some factor, e.g., $RY(\overleftarrow{x}\theta_1)$. This is the case for the data encoding unit $E(\overleftarrow{x},\theta_1,\theta_2)$ as well as for the post-processing step $V(\theta_1,\theta_2)$. Moreover, these two parts of the circuit use the same parameters θ_1 and θ_2. Furthermore, the decomposition of one controlled rotation gate contains two rotation gates depending on the same parameter but also multiplied by a constant factor, e.g., $RX(\theta_2/2)$. We refer to this kind of ansatz as one that has *shared weights* and *constant factors*. This observation is crucial since the common parameter-shift rule does not take this aspect into account. We therefore extend the parameter-shift rule in the following to be applicable for these circuit architectures as well.

3.5 Quantum Gradients

In order to train a quantum predictive model, we want to utilize gradient-based optimizers. Therefore, the gradient of the cost function needs to be calculated, which is the goal of this section. Recall that the proposed cost function from Section 3.3.2 reads as

$$C(\boldsymbol{\theta}) = \sum_{\overleftarrow{x}} w_{\overleftarrow{x}} MMD^2[F,P,\hat{P}_{\boldsymbol{\theta}}|\overleftarrow{x}] + cP_{\boldsymbol{\theta}}(M_q = 1|\overleftarrow{x}) \tag{214}$$

with

$$MMD^2[F,P,\hat{P}_{\boldsymbol{\theta}}|\overleftarrow{x}] = \mathbb{E}_{x,x'}[k(x,x')] - 2\,\mathbb{E}_{x,y}[k(x,y)] + \mathbb{E}_{y,y'}[k(y,y')] \tag{215}$$

being the conditional maximum mean discrepancy, where x, x' are sampled from P and y, y' are sampled from $\hat{P}_{\boldsymbol{\theta}}$, respectively. Within this work, we make use of the Gaussian kernel

$$k(x,y) = \frac{1}{n_\gamma}\sum_{i=1}^{n_\gamma} \exp\left(-\frac{|x-y|^2}{2\gamma_i}\right), \tag{216}$$

and thus omit the parameter F from the MMD in the following. P denotes the probability distribution of the underlying stochastic process and $\hat{P}_{\boldsymbol{\theta}}$, $P_{\boldsymbol{\theta}}$ the distribution associated with measuring the auxiliary and the memory register of the VQC, respectively. While $\hat{P}_{\boldsymbol{\theta}}$ is understood as the probability distribution of the corresponding quantum predictive model, $P_{\boldsymbol{\theta}}$ is used to determine if the model has a valid set of memory states. Both distributions are defined by the VQC and can be written as expectation values, i.e.,

$$\hat{P}_{\boldsymbol{\theta}}(x|\overleftarrow{x}) = \langle 00|U(\overleftarrow{x},\boldsymbol{\theta})^\dagger \hat{O}_x U(\overleftarrow{x},\boldsymbol{\theta})|00\rangle\,, \tag{217}$$

with $U(\overleftarrow{x},\boldsymbol{\theta})$ being the unitary of the VQC and $\hat{O}_x = \sum_y |yx\rangle\langle yx|$ being the observable for measuring the auxiliary register in x, cf. Example 2.3. Similarly, the distribution $P_{\boldsymbol{\theta}}$ can also be written as an expectation value according to

$$P_{\boldsymbol{\theta}}(M_q = 1|\overleftarrow{x}) = \langle 00|U(\overleftarrow{x},\boldsymbol{\theta})^\dagger \hat{O}_1 U(\overleftarrow{x},\boldsymbol{\theta})|00\rangle\,, \tag{218}$$

with $\hat{O}_1 = \sum_y |1y\rangle\langle 1y|$ being the observable for measuring the memory register as 1. Taking the derivative of the cost function therefore leads to taking the derivatives of these expectation values, for which the parameter-shift rule can be used. However, our ansatz uses shared weights and constant factors but the common parameter-shift rule does not include these aspects out of the box. We therefore extend the parameter-shift rule such that it can be applied onto Equations (217) and (218). Since the proofs for the extension are closely connected to the proof of the common parameter-shift rule, we start with looking at those proofs in detail, which can also be found in Ref. [21].

3.5.1 Common Parameter-Shift Rule

Within this section, we are interested in taking the derivatives of functions $f : \mathbb{R}^n \to \mathbb{R}$ which are of the form of

$$f(\boldsymbol{\theta}) = \langle 0|U(\boldsymbol{\theta})^\dagger \hat{O} U(\boldsymbol{\theta})|0\rangle , \tag{219}$$

with an observable $\hat{O}$ and a unitary $U(\boldsymbol{\theta})$. We start by defining the gates which the common parameter-shift rule can be applied to.

Definition 3.6. A *single qubit rotation gate* is a unitary $G(\mu) = e^{-ia\mu\hat{G}}$ with $a, \mu \in \mathbb{R}$ and $\hat{G} : \mathbb{C}^2 \to \mathbb{C}^2$ some Hermitian generator with two distinct eigenvalues $\pm\lambda$.

These gates have a useful property, which is summarized in the next Lemma.

Lemma 3.16. Let G be a single qubit rotation gate. Then

$$G(\mu) = e^{-ia\mu\hat{G}} = I\cos(r \cdot \mu) - i\frac{a}{r}\hat{G}\sin(r \cdot \mu), \tag{220}$$

with $r = a\lambda$.

Proof. Since $\hat{G}$ is Hermitian, we can diagonalize the operator and get

$$\hat{G} = U^{-1}DU = U^\dagger DU, \tag{221}$$

with some unitary $U \in \mathbb{C}^{2\times 2}$ and a diagonal matrix $D \in \mathbb{R}^{2\times 2}$ with entries $\pm\lambda$. Thus, we can rewrite the single qubit rotation gate as

$$\begin{aligned} G(\mu) &= e^{-ia\mu\hat{G}} && (222)\\ &= e^{-ia\mu U^\dagger DU} && (223)\\ &= U^\dagger e^{-ia\mu D} U && (224)\\ &= U^\dagger G'(\mu) U, && (225)\end{aligned}$$

where we defined $G'(\mu) = e^{-ia\mu D}$. Note that for the diagonal matrix D we have

$$D^2 = \lambda^2 I. \tag{226}$$

Applying the definition of the matrix exponential together with Equation (226) yields

$$\begin{aligned} G'(\mu) &= e^{-ia\mu D} && (227)\\ &= \sum_{k=0}^{\infty} \frac{(-ia\mu)^k D^k}{k!} && (228)\\ &= \sum_{k=0}^{\infty} \frac{(-ia\mu)^{2k} D^{2k}}{(2k)!} + \sum_{k=0}^{\infty} \frac{(-ia\mu)^{2k+1} D^{2k+1}}{(2k+1)!} && (229)\\ &= I\sum_{k=0}^{\infty} \frac{(-1)^k (r\mu)^{2k}}{(2k)!} - i\frac{a}{r}D\sum_{k=0}^{\infty} \frac{(-1)^k (r\mu)^{2k+1}}{(2k+1)!} && (230)\\ &= I\cos(r\mu) - i\frac{a}{r}D\sin(r\mu), && (231)\end{aligned}$$

with the definition of $r = a\lambda$. The last step is to multiply this expression with $U^\dagger$ from the left and with U from the right side, such that

$$U^\dagger U\cos(r\mu) - i\frac{a}{r}U^\dagger DU\sin(r\mu) = I\cos(r\mu) - i\frac{a}{r}\hat{G}\sin(r\mu) = G(\mu). \tag{232}$$

□

Thus, single qubit rotation gates can be written as a sum of sin and cos terms. We get a special case of Lemma 3.16 if we consider a rotation of $\pm\frac{\pi}{4r}$.

Lemma 3.17. Under the same assumptions of Lemma 3.16 we get

$$G\Big(\pm\frac{\pi}{4r}\Big)=\frac{1}{\sqrt{2}}\Big(I\mp i\frac{a}{r}\hat{G}\Big) \tag{233}$$

respectively

$$I\mp i\frac{a}{r}\hat{G}=\sqrt{2}G\Big(\pm\frac{\pi}{4r}\Big). \tag{234}$$

This shows that we can write a single qubit rotation gate G as the identity I plus the generator $\hat{G}$ for a specific rotation. Additionally, we show the following lemma:

Lemma 3.18. Let B, C be unitary, $\hat{O}$ some observable and $|\psi\rangle$ some state. Then

$$\langle\psi|B^\dagger\hat{O}C|\psi\rangle+\langle\psi|C^\dagger\hat{O}B|\psi\rangle=\frac{1}{2}\Big[\langle\psi|(B+C)^\dagger\hat{O}(B+C)|\psi\rangle-\langle\psi|(B-C)^\dagger\hat{O}(B-C)|\psi\rangle\Big]. \tag{235}$$

Proof. We have

$$\frac{1}{2}\Big[\langle\psi|(B+C)^\dagger\hat{O}(B+C)|\psi\rangle-\langle\psi|(B-C)^\dagger\hat{O}(B-C)|\psi\rangle\Big] \tag{236}$$

$$=\frac{1}{2}\Big[\langle\psi|B^\dagger\hat{O}B|\psi\rangle+\langle\psi|B^\dagger\hat{O}C|\psi\rangle+\langle\psi|C^\dagger\hat{O}B|\psi\rangle+\langle\psi|C^\dagger\hat{O}C|\psi\rangle \tag{237}$$

$$-\langle\psi|B^\dagger\hat{O}B|\psi\rangle+\langle\psi|B^\dagger\hat{O}C|\psi\rangle+\langle\psi|C^\dagger\hat{O}B|\psi\rangle-\langle\psi|C^\dagger\hat{O}C|\psi\rangle\Big] \tag{238}$$

$$=\langle\psi|B^\dagger\hat{O}C|\psi\rangle+\langle\psi|C^\dagger\hat{O}B|\psi\rangle. \tag{239}$$

□

Based on these lemmas, we can now prove the common parameter-shift rule:

Theorem 3.3. Let $U(\boldsymbol{\theta})$ be unitary and $\hat{O}$ some observable. Define $\theta_i=\mu$ and let the parameter μ only affect one single qubit rotation gate $G(\mu)$. The derivative of f with respect to μ is then given by

$$\frac{\partial f(\boldsymbol{\theta})}{\partial\mu}=r\Big[f(\mu+s)-f(\mu-s)\Big], \tag{240}$$

with $s=\frac{\pi}{4r}$, $r=a\lambda$ and $\mu\pm s=(\theta_1,\dots,\mu\pm s,\dots,\theta_N)^T$.

Proof. Since the parameter μ only affects one single gate, we can write the circuit as

$$U(\boldsymbol{\theta})=VG(\mu)W, \tag{241}$$

with some unitary V and W that are independent of μ. Deriving f thus yields

$$\frac{\partial f(\boldsymbol{\theta})}{\partial\mu}=\partial_\mu\langle\psi|G(\mu)^\dagger\hat{Q}G(\mu)|\psi\rangle \tag{242}$$

$$=\langle\psi|(\partial_\mu G(\mu))^\dagger\hat{Q}G(\mu)|\psi\rangle+\langle\psi|G(\mu)^\dagger\hat{Q}(\partial_\mu G(\mu))|\psi\rangle, \tag{243}$$

where we defined $\hat{Q}=V^\dagger\hat{O}V$ and $|\psi\rangle=W|0\rangle$. With the known form of $G(\mu)$, the derivatives can be calculated according to

$$\partial_\mu G(\mu)=\partial_\mu\Big(e^{-ia\mu\hat{G}}\Big)=-ia\hat{G}e^{-ia\mu\hat{G}}=-ia\hat{G}G(\mu) \tag{244}$$

and

$$(\partial_\mu G(\mu))^\dagger = (-ia\hat{G}G(\mu))^\dagger = G(\mu)^\dagger(+ia\hat{G}). \tag{245}$$

Substituting these derivatives in Equation (242) yields

$$\frac{\partial f(\boldsymbol{\theta})}{\partial\mu} = \langle\psi'|(+iaG)\hat{Q}|\psi'\rangle + \langle\psi'|\hat{Q}(-iaG)|\psi'\rangle, \tag{246}$$

with $|\psi'\rangle = G(\mu)|\psi\rangle$. Next, Lemma 3.18 can be applied, which leads to

$$\frac{\partial f(\boldsymbol{\theta})}{\partial\mu} = \frac{r}{2}\Big[\langle\psi'|\Big(I - i\frac{a}{r}\hat{G}\Big)^\dagger\hat{Q}\Big(I - i\frac{a}{r}\hat{G}\Big)|\psi'\rangle - \langle\psi'|\Big(I + i\frac{a}{r}\hat{G}\Big)^\dagger\hat{Q}\Big(I + i\frac{a}{r}\hat{G}\Big)|\psi'\rangle\Big], \tag{247}$$

where we used $B = I$, $C = -i\frac{a}{r}\hat{G}$ and $r = a\lambda$. With Lemma 3.17 we now get

$$\frac{\partial f(\boldsymbol{\theta})}{\partial\mu} = r\Big[\langle\psi'|G\Big(+\frac{\pi}{4r}\Big)^\dagger\hat{Q}G\Big(+\frac{\pi}{4r}\Big)|\psi'\rangle - \langle\psi'|G\Big(-\frac{\pi}{4r}\Big)^\dagger\hat{Q}G\Big(-\frac{\pi}{4r}\Big)|\psi'\rangle\Big]. \tag{248}$$

Next, we remember $|\psi'\rangle = G(\mu)|\psi\rangle$ and exploit the fact that $G(\mu)G(s) = G(\mu+s)$ for any $s \in \mathbb{R}$. Thus

$$\begin{aligned}\frac{\partial f(\boldsymbol{\theta})}{\partial\mu} &= r\Big[\langle\psi|(G(\mu+s))^\dagger\hat{Q}G(\mu+s)|\psi\rangle - \langle\psi|(G(\mu-s))^\dagger\hat{Q}G(\mu-s)|\psi\rangle\Big] &(249)\\ &= r\Big[f(\mu+s) - f(\mu-s)\Big], &(250)\end{aligned}$$

with $s = \frac{\pi}{4r}$ and $r = a\lambda$. □

We conclude that we can evaluate the gradient for the VQC for a single parameter μ when shifting μ with $+s$ and $-s$ and summing up the two expectation values. Since s and thus r depend on the factor a, the parameter-shift rule needs to take the concrete parameterized gates into account. For the case of Pauli rotation gates, $a = \frac{1}{2}$ and thus we get the following Lemma.

Lemma 3.19. The common parameter-shift rule for the Pauli rotation gates reads as

$$\frac{\partial f(\boldsymbol{\theta})}{\partial\mu} = \frac{1}{2}\Big[f\Big(\mu + \frac{\pi}{2}\Big) - f\Big(\mu - \frac{\pi}{2}\Big)\Big]. \tag{251}$$

Proof. The Pauli gates X, Y and Z are unitary and Hermitian and have the eigenvalues ± 1. Thus, the Pauli rotation gates RX, RY, RZ are single qubit rotation gates with $a = \frac{1}{2}$ and generators $\hat{G} \in \{X, Y, Z\}$. □

The common parameter-shift rule allows calculating gradients for VQCs where each parameter affects only one single qubit rotation gate. However, if we have shared weights or constant factors, the parameter-shift rule can not be applied. We therefore generalize the parameter-shift rule in the following to be applicable for such circuit architectures.

3.5.2 Generalized Parameter-Shift Rule

In particular, we now consider VQCs where a parameter $\mu = \theta_i$ is transformed under a continuously differentiable map $\alpha : \mathbb{R} \to \mathbb{R}$ and affects at most $m \in \mathbb{N}$ single qubit rotation gates. We denote the gate that is affected by the j-th occurence of parameter θ_i with $G_{i,j}$, i.e., we have $G_{i,j}(\alpha_{i,j}(\theta_i))$. First, we consider the Pauli rotation gates $G \in \{RX, RY, RZ\}$ and the case when θ_i affects only one gate but is transformed via a map α.

Lemma 3.20. Let $U(\boldsymbol{\theta})$ be a unitary, $\hat{O}$ some observable and $\alpha : \mathbb{R} \to \mathbb{R}$ continuously differentiable. Define $\theta_i = \mu$ and let the parameter μ only affect one Pauli rotation gate according to $G(\alpha(\mu))$. The derivative of f with respect to μ is then given by

$$\frac{\partial f(\boldsymbol{\theta})}{\partial \mu} = \frac{1}{2}\alpha'(\mu)\Big[f\Big(\alpha(\mu)+\frac{\pi}{2}\Big) - f\Big(\alpha(\mu)-\frac{\pi}{2}\Big)\Big], \tag{252}$$

with $\alpha(\mu) \pm \frac{\pi}{2} = (\theta_1, \dots, \alpha(\mu) \pm \frac{\pi}{2}, \dots, \theta_N)^T$.

Proof. Following Theorem 3.3, the circuit can be written as $U(\boldsymbol{\theta}) = VG(\alpha(\mu))W$. Applying the chain-rule thus leads to

$$\frac{\partial f(\boldsymbol{\theta})}{\partial \mu} = \partial_\mu \langle\psi|G(\alpha(\mu))^\dagger \hat{Q} G(\alpha(\mu))|\psi\rangle \tag{253}$$

$$= \alpha'(\mu)\Big[\langle\psi|(\partial_{\hat{\mu}}G(\hat{\mu}))^\dagger \hat{Q} G(\hat{\mu})|\psi\rangle + \langle\psi|G(\hat{\mu})^\dagger \hat{Q}(\partial_{\hat{\mu}}G(\hat{\mu}))|\psi\rangle\Big], \tag{254}$$

with $\hat{\mu} = \alpha(\mu)$, $\hat{Q} = V^\dagger \hat{O} V$ and $|\psi\rangle = W|0\rangle$. From here, the proof can be done analogously with $\hat{\mu}$ instead of μ leading to

$$\frac{\partial f(\boldsymbol{\theta})}{\partial \mu} = \frac{1}{2}\alpha'(\mu)\Big[f\Big(\hat{\mu}+\frac{\pi}{2}\Big) - f\Big(\hat{\mu}-\frac{\pi}{2}\Big)\Big] \tag{255}$$

$$= \frac{1}{2}\alpha'(\mu)\Big[f\Big(\alpha(\mu)+\frac{\pi}{2}\Big) - f\Big(\alpha(\mu)-\frac{\pi}{2}\Big)\Big] \tag{256}$$

□

Next, we consider the case when multiple gates are affected by θ_i, but no function α is applied onto the parameters.

Lemma 3.21. Let $U(\boldsymbol{\theta})$ be a unitary and $\hat{O}$ some observable. Define $\theta_i = \mu$ and let the parameter μ affect two Pauli rotation gates $G_1(\mu), G_2(\mu)$. The derivative of f with respect to μ is then given by

$$\frac{\partial f(\boldsymbol{\theta})}{\partial \mu} = \frac{1}{2}\Big[f_1\Big(\mu,+\frac{\pi}{2}\Big) - f_1\Big(\mu,-\frac{\pi}{2}\Big)\Big] \tag{257}$$

$$+ \frac{1}{2}\Big[f_2\Big(\mu,+\frac{\pi}{2}\Big) - f_2\Big(\mu,-\frac{\pi}{2}\Big)\Big], \tag{258}$$

with $f_i(\mu, \eta)$ indicating that the parameter for $G_i(\mu)$ is shifted by η but the other is fixed to μ.

Proof. The circuit can be written as $U(\boldsymbol{\theta}) = FG_1(\mu)VG_2(\mu)W$ where $G_1(\mu)$ is the first and $G_2(\mu)$ is the second gate in the circuit containing the parameter μ, and F, V, W some unitaries. Following Equation (242) we can derive the expectation value as follows:

$$\frac{\partial f(\boldsymbol{\theta})}{\partial \mu} = \partial_\mu \langle 0|(FG_1(\mu)VG_2(\mu)W)^\dagger \hat{O} FG_1(\mu)VG_2(\mu)W|0\rangle \tag{259}$$

$$= \langle\psi|(\partial_\mu G_2(\mu))^\dagger V^\dagger G_1(\mu)^\dagger \hat{Q} G_1(\mu) V G_2(\mu)|\psi\rangle \tag{260}$$

$$+ \langle\psi|G_2(\mu)^\dagger V^\dagger (\partial_\mu G_1(\mu))^\dagger \hat{Q} G_1(\mu) V G_2(\mu)|\psi\rangle \tag{261}$$

$$+ \langle\psi|G_2(\mu)^\dagger V^\dagger G_1(\mu)^\dagger \hat{Q}(\partial_\mu G_1(\mu)) V G_2(\mu)|\psi\rangle \tag{262}$$

$$+ \langle\psi|G_2(\mu)^\dagger V^\dagger G_1(\mu)^\dagger \hat{Q} G_1(\mu) V(\partial_\mu G_2(\mu))|\psi\rangle, \tag{263}$$

where we used the product rule and defined $|\psi\rangle = W|0\rangle$ and $\hat{Q} = F^\dagger \hat{O} U$. Next, we define $\hat{R}(\mu) = V^\dagger G_1(\mu)^\dagger \hat{Q} G_1(\mu) V$ and $|\psi'(\mu)\rangle = VG_2(\mu)|\psi\rangle$ and rewrite the sum as

$$\frac{\partial f(\boldsymbol{\theta})}{\partial \mu} = \langle\psi|(\partial_\mu G_2(\mu))^\dagger \hat{R}(\mu) G_2(\mu)|\psi\rangle \tag{264}$$

$$+ \langle\psi|G_2(\mu)^\dagger \hat{R}(\mu)(\partial_\mu G_2(\mu))|\psi\rangle \tag{265}$$

$$+ \langle\psi'(\mu)|(\partial_\mu G_1(\mu))^\dagger \hat{Q} G_1(\mu)|\psi'(\mu)\rangle \tag{266}$$

$$+ \langle\psi'(\mu)|G_1(\mu)^\dagger \hat{Q}(\partial_\mu G_1(\mu))|\psi'(\mu)\rangle\,. \tag{267}$$

Taking a closer look at the four different expectation values, we see that we are in the same situation as in Equation (242). Thus we can apply Theorem 3.3 twice and get

$$\frac{\partial f(\boldsymbol{\theta})}{\partial\mu} = \frac{1}{2}\left[f_1\left(\mu,+\frac{\pi}{2}\right) - f_1\left(\mu,-\frac{\pi}{2}\right)\right] \tag{268}$$

$$+ \frac{1}{2}\left[f_2\left(\mu,+\frac{\pi}{2}\right) - f_2\left(\mu,-\frac{\pi}{2}\right)\right], \tag{269}$$

with

$$f_1(\mu,\eta) = \langle 0|(FG_1(\mu+\eta)VG_2(\mu)W)^\dagger \hat{O} G_1(\mu+\eta)VG_2(\mu)W|0\rangle\,, \tag{270}$$

$$f_2(\mu,\eta) = \langle 0|(FG_1(\mu)VG_2(\mu+\eta)W)^\dagger \hat{O} G_1(\mu)VG_2(\mu+\eta)W|0\rangle\,, \tag{271}$$

i.e., for f_j, the j-th occurrence of the parameter μ is shifted by η but the other is fixed to μ. □

Lemma 3.21 can be extended to multiple occurrences as follows.

Lemma 3.22. Let $U(\boldsymbol{\theta})$ be a VQC and $\hat{O}$ some observable. Define $\theta_i = \mu$ and let the parameter μ affect at most m Pauli rotation gates $G_1(\mu), G_2(\mu), \ldots, G_m(\mu)$. The derivative of f with respect to μ is then given by

$$\frac{\partial f(\boldsymbol{\theta})}{\partial\mu} = \frac{1}{2}\sum_{j=1}^{m}\left[f_j\left(\mu,+\frac{\pi}{2}\right) - f_j\left(\mu,-\frac{\pi}{2}\right)\right], \tag{272}$$

with $f_j(\mu,\eta)$ indicating that the parameter for $G_j(\mu)$ is shifted by η but all others are fixed to μ.

Proof. The proof follows the one from Lemma 3.21 but uses the product rule for m products instead. Deriving the expectation value leads to a sum of terms of the following form:

$$\langle 0|F_m^\dagger G_m(\mu)^\dagger \ldots F_j^\dagger(\partial_\mu G_j(\mu))^\dagger \ldots F_1^\dagger G_1(\mu)^\dagger F_0^\dagger \hat{O} F_0 G_1(\mu) F_1 \ldots G_m(\mu) F_m|0\rangle \tag{273}$$

together with their Hermitian conjugates

$$\langle 0|F_m^\dagger G_m(\mu)^\dagger \ldots F_1^\dagger G_1(\mu)^\dagger F_0^\dagger \hat{O} F_0 G_1(\mu) F_1 \ldots (\partial_\mu G_j(\mu)) F_j \ldots G_m(\mu) F_m|0\rangle\,. \tag{274}$$

Thus, the parts that are independent of the derivative operator ∂_μ can always be absorbed in the state

$$|\psi'\rangle = F_j \ldots G_m(\mu) F_m\,|0\rangle \tag{275}$$

and the observable

$$\hat{Q} = F_{j-1}^\dagger G_{j-1}(\mu)^\dagger \ldots \hat{O} \ldots G_{j-1}(\mu) F_{j-1}, \tag{276}$$

yielding a sum of the form of

$$\langle\psi'|(\partial_\mu G_j(\mu))^\dagger \hat{Q} G_j(\mu)|\psi'\rangle + \langle\psi'|G_j(\mu)^\dagger \hat{Q}(\partial_\mu G_j(\mu))|\psi'\rangle + \ldots. \tag{277}$$

For each such a pair in the sum, Theorem 3.3 can be applied leading to

$$\frac{\partial f(\boldsymbol{\theta})}{\partial\mu} = \frac{1}{2}\sum_{j=1}^{m}\left[f_j\left(\mu,+\frac{\pi}{2}\right) - f_j\left(\mu,-\frac{\pi}{2}\right)\right]. \tag{278}$$

□

Next, we combine Lemma 3.20 with Lemma 3.22 to get the final result of the generalized parameter-shift rule for shared parameters with constant factors.

Theorem 3.4. Let $U(\boldsymbol{\theta})$ be a VQC, $\hat{O}$ some observable and $\{\alpha_j, \alpha_j : \mathbb{R} \to \mathbb{R}\}_{j=1}^{m}$ a set of m continuously differentiable functions. Define $\theta_i = \mu$ and let the parameter μ affect at most m Pauli rotation gates according to $G_j(\alpha_j(\mu))$. The derivative of f with respect to μ is then given by

$$\frac{\partial f(\boldsymbol{\theta})}{\partial \mu} = \frac{1}{2}\sum_{j=1}^{m} \alpha'_j(\mu)\Big[f_j\Big(\alpha_j(\mu), +\frac{\pi}{2}\Big) - f_j\Big(\alpha_j(\mu), -\frac{\pi}{2}\Big)\Big], \tag{279}$$

with $f_j(\alpha_j(\mu), \eta)$ indicating that the parameter $\alpha_j(\mu)$ for G_j is shifted by η, i.e., $G_j(\alpha_j(\mu) \pm \eta)$, but all others are fixed to $\alpha_j(\mu)$.

Proof. The proof follows Lemma 3.22 but applies the chain-rule to each pair in Equation (277) leading to

$$\alpha'_j(\mu)\Big[\langle\psi'|(\partial_{\hat{\mu}_j} G_j(\hat{\mu}_j))^\dagger \hat{Q} G_j(\hat{\mu}_j)|\psi'\rangle + \langle\psi'|G_j(\hat{\mu}_j)^\dagger \hat{Q}(\partial_{\hat{\mu}_j} G_j(\hat{\mu}_j))|\psi'\rangle\Big] + \ldots, \tag{280}$$

with $\hat{\mu}_j = \alpha_j(\mu)$. The rest of the proof follows Lemma 3.22 but with $\hat{\mu}_j$, leading to

$$\frac{\partial f(\boldsymbol{\theta})}{\partial \mu} = \frac{1}{2}\sum_{j=1}^{m} \alpha'_j(\mu)\Big[f_j\Big(\hat{\mu}_j, +\frac{\pi}{2}\Big) - f_j\Big(\hat{\mu}_j, -\frac{\pi}{2}\Big)\Big]. \tag{281}$$

Substituting $\hat{\mu}_j = \alpha_j(\mu)$ completes the proof. □

The generalized parameter-shift rule in Theorem 3.4 can be applied onto VQCs with Pauli rotation gates. As a remark, the result does also hold for general single qubit rotation gates as defined in Definition 3.6. For these gates, the generalized parameter-shift rule reads as

$$\frac{\partial f(\boldsymbol{\theta})}{\partial \mu} = \sum_{j=1}^{m} r_j \alpha'_j(\mu)\Big[f_j(\alpha_j(\mu), +s_j) - f_j(\alpha_j(\mu), -s_j)\Big], \tag{282}$$

with $s_j = \frac{\pi}{4r_j}$ and $r_j = a_j\lambda_j$.
With this rule in place, we can now take the derivative of the cost function, which is done next.

3.5.3 Gradient of the Cost Function

Taking the derivative of the cost function from Equation (214) leads to

$$\frac{\partial C(\boldsymbol{\theta})}{\partial \theta_i} = \sum_{\overleftarrow{x}} w_{\overleftarrow{x}} \frac{\partial MMD^2[P, \hat{P}_{\boldsymbol{\theta}}|\overleftarrow{x}]}{\partial \theta_i} + c\frac{\partial P_{\boldsymbol{\theta}}(M_q = 1|\overleftarrow{x})}{\partial \theta_i}. \tag{283}$$

Our overall goal is to get an expression for the two partial derivatives that we can use in order to compute the gradient. In the following, we discuss the two terms separately and start with the conditional maximum mean discrepancy. Here, we first neglect the conditional past and simply derive the MMD from Equation (73) which leads to

$$\frac{\partial MMD^2[P, \hat{P}_{\boldsymbol{\theta}}]}{\partial \theta_i} = \sum_{x,y} k(x,y)\Big(\hat{P}_{\boldsymbol{\theta}}(y)\frac{\partial \hat{P}_{\boldsymbol{\theta}}(x)}{\partial \theta_i} + \hat{P}_{\boldsymbol{\theta}}(x)\frac{\partial \hat{P}_{\boldsymbol{\theta}}(y)}{\partial \theta_i}\Big) - 2\sum_{x,y} k(x,y)P(y)\frac{\partial \hat{P}_{\boldsymbol{\theta}}(x)}{\partial \theta_i}. \tag{284}$$

We will include the conditional behavior again later, but for now, we focus on how to further derive the partial derivative of the probability distribution, i.e., $\partial \hat{P}_{\boldsymbol{\theta}}(x)/\partial \theta_i$. We will see that the calculation of this quantity depends on the ansatz, i.e., whether shared weights or constant factors are used. Starting with a VQC that contains neither and subsequently introducing constant factors and shared weights, we combine the results to get a formula for the conditional

MMD. All of the following proofs are based on Equation (284) and apply the corresponding parameter-shift rule to get an expression for the partial derivative of the probability distribution. We start with the simple case, in which the VQC does neither include shared weights nor constant factors. This case has already been studied by Liu et al. for the quantum circuit Born machine [18]. Here, the gradient of the MMD can be expressed as a sum of expectation values over the probability distributions of the VQC with shifted parameters.

Lemma 3.23. Let $U(\boldsymbol{\theta})$ be a VQC where each θ_i affects only one Pauli rotation gate. The gradient of the MMD is then given by

$$\frac{\partial MMD^2[P,\hat{P}_\theta]}{\partial\theta_i} = \mathop{\mathbb{E}}_{\substack{x\sim\hat{P}_{\boldsymbol{\theta}_i^+}\\ y\sim\hat{P}_{\boldsymbol{\theta}}}}[k(x,y)] - \mathop{\mathbb{E}}_{\substack{x\sim\hat{P}_{\boldsymbol{\theta}_i^-}\\ y\sim\hat{P}_{\boldsymbol{\theta}}}}[k(x,y)] - \mathop{\mathbb{E}}_{\substack{x\sim\hat{P}_{\boldsymbol{\theta}_i^+}\\ y\sim P}}[k(x,y)] + \mathop{\mathbb{E}}_{\substack{x\sim\hat{P}_{\boldsymbol{\theta}_i^-}\\ y\sim P}}[k(x,y)], \tag{285}$$

with $\hat{P}_{\boldsymbol{\theta}_i^\pm}(x)$ being the probability to measure x from the VQC with parameters $\boldsymbol{\theta}_i^\pm = (\theta_1,\theta_2,\ldots,\theta_i\pm\frac{\pi}{2},\ldots,\theta_n)^T$.

Proof. The probability to measure x can be expressed as expectation value via

$$\hat{P}_{\boldsymbol{\theta}}(x) = \langle 0|U(\boldsymbol{\theta})^\dagger\hat{O}U(\boldsymbol{\theta})|0\rangle \tag{286}$$

with the observable $\hat{O} = |x\rangle\langle x|$. Thus, we can apply the common parameter-shift rule (Lemma 3.19) and get

$$\frac{\partial\hat{P}_{\boldsymbol{\theta}}(x)}{\partial\theta_i} = \frac{1}{2}\Big(\hat{P}_{\boldsymbol{\theta}_i^+}(x) - \hat{P}_{\boldsymbol{\theta}_i^-}(x)\Big), \tag{287}$$

i.e., the value of the probability distribution is the mean of two shifted probability distributions. Therefore, we can substitute the partial derivative in Equation (284) which yields

$$\begin{aligned}\frac{\partial MMD^2[P,P_{\boldsymbol{\theta}}]}{\partial\theta_i} &= \frac{1}{2}\Big(\sum_{x,y}k(x,y)\hat{P}_{\boldsymbol{\theta}}(y)\hat{P}_{\boldsymbol{\theta}_i^+}(x) - \sum_{x,y}k(x,y)\hat{P}_{\boldsymbol{\theta}}(y)\hat{P}_{\boldsymbol{\theta}_i^-}(x)\Big) &(288)\\ &+ \frac{1}{2}\Big(\sum_{x,y}k(x,y)\hat{P}_{\boldsymbol{\theta}}(x)\hat{P}_{\boldsymbol{\theta}_i^+}(y) - \sum_{x,y}k(x,y)\hat{P}_{\boldsymbol{\theta}}(x)\hat{P}_{\boldsymbol{\theta}_i^-}(y)\Big) &(289)\\ &- \Big(\sum_{x,y}k(x,y)\hat{P}_{\boldsymbol{\theta}_i^+}(x)P(y) - \sum_{x,y}k(x,y)\hat{P}_{\boldsymbol{\theta}_i^-}(x)P(y)\Big). &(290)\end{aligned}$$

Since the kernel function k is symmetric, i.e., $k(x,y) = k(y,x)$, we get

$$\begin{aligned}\frac{\partial MMD^2[P,P_{\boldsymbol{\theta}}]}{\partial\theta_i} &= \sum_{x,y}k(x,y)\hat{P}_{\boldsymbol{\theta}_i^+}(x)\hat{P}_{\boldsymbol{\theta}}(y) - \sum_{x,y}k(x,y)\hat{P}_{\boldsymbol{\theta}_i^-}(x)\hat{P}_{\boldsymbol{\theta}}(y) &(291)\\ &- \sum_{x,y}k(x,y)\hat{P}_{\boldsymbol{\theta}_i^+}(x)P(y) + \sum_{x,y}k(x,y)\hat{P}_{\boldsymbol{\theta}_i^-}(x)P(y). &(292)\end{aligned}$$

□

Calculating the gradient of the MMD thus makes use of calculating the expectation values for the same VQC but with shifted parameters. For the data encoding unit of our ansatz, we multiply a past observation $\overleftarrow{x}$ with a trainable parameter θ_i, which we referred to as an ansatz with constant factors. Thus, we essentially perform a mapping $\alpha(\theta_i) = \overleftarrow{x}\cdot\theta_i$, which will then be the rotation angle of a Pauli rotation gate, i.e., $G_i(\alpha(\theta_i))$, for the Pauli gate G_i affected by the parameter θ_i. Therefore, we extend the above formula for the partial derivative of the MMD to also include such mappings.

Lemma 3.24. Let $\{\alpha_i, \alpha_i : \mathbb{R} \to \mathbb{R}\}_{i=1}^n$ be a set of n continuously differentiable maps. Moreover, let $U(\boldsymbol{\theta})$ be a VQC where each θ_i affects only one Pauli rotation gate according to $G_i(\alpha_i(\theta_i))$. The gradient of the MMD is then given by

$$\frac{\partial MMD^2[P, \hat{P}_{\boldsymbol{\theta}}]}{\partial \theta_i} = \alpha_i'(\theta_i)\Big[\underset{\substack{x\sim \hat{P}_{\boldsymbol{\theta}_i^+} \\ y\sim \hat{P}_{\boldsymbol{\theta}}}}{\mathbb{E}}[k(x,y)] - \underset{\substack{x\sim \hat{P}_{\boldsymbol{\theta}_i^-} \\ y\sim \hat{P}_{\boldsymbol{\theta}}}}{\mathbb{E}}[k(x,y)] - \underset{\substack{x\sim \hat{P}_{\boldsymbol{\theta}_i^+} \\ y\sim P}}{\mathbb{E}}[k(x,y)] + \underset{\substack{x\sim \hat{P}_{\boldsymbol{\theta}_i^-} \\ y\sim P}}{\mathbb{E}}[k(x,y)]\Big], \tag{293}$$

with $\hat{P}_{\boldsymbol{\theta}_i^\pm}(x)$ being the probability to measure x from the VQC with parameters $\boldsymbol{\alpha}(\boldsymbol{\theta}_i^\pm) = (\alpha_1(\theta_1), \alpha_2(\theta_2), \ldots, \alpha_i(\theta_i) \pm \frac{\pi}{2}, \ldots, \alpha_n(\theta_n))^T$.

Proof. The proof is analogous to the one of Lemma 3.23, but applies the generalized parameter-shift rule for continuously differentiable maps $\alpha : \mathbb{R} \to \mathbb{R}$ instead (Lemma 3.20), which leads to

$$\frac{\partial \hat{P}_{\boldsymbol{\theta}}(x)}{\partial \theta_i} = \frac{1}{2}\alpha_i'(\theta_i)\Big(\hat{P}_{\boldsymbol{\theta}_i^+}(x) - \hat{P}_{\boldsymbol{\theta}_i^-}(x)\Big). \tag{294}$$

Inserting this identity into Equation (284) finishes the proof. □

Lemma 3.24 shows that the use of some maps $\alpha_i : \mathbb{R} \to \mathbb{R}$ simply lead to a factor $\alpha_i'(\theta_i)$ when calculating the gradient. With the quantum post-processing step, we have also introduced shared weights into our circuit. Thus we have parameters θ_i that affect $j = 1, 2, \ldots, m$ Pauli rotation gates, which we denote as $G_{i,j}(\theta_i)$. The formula for the gradient can be extended to such architectures as follows.

Lemma 3.25. Let $U(\boldsymbol{\theta})$ be a VQC where each θ_i affects at most m Pauli rotation gates according to $G_{i,j}(\theta_i)$ with $j = 1, 2, \ldots, m$. The gradient of the MMD is then given by

$$\frac{\partial MMD^2[P, \hat{P}_{\boldsymbol{\theta}}]}{\partial \theta_i} = \sum_{j=1}^m \Big[\underset{\substack{x\sim \hat{P}_{\boldsymbol{\theta}_{i,j}^+} \\ y\sim \hat{P}_{\boldsymbol{\theta}}}}{\mathbb{E}}[k(x,y)] - \underset{\substack{x\sim \hat{P}_{\boldsymbol{\theta}_{i,j}^-} \\ y\sim \hat{P}_{\boldsymbol{\theta}}}}{\mathbb{E}}[k(x,y)] - \underset{\substack{x\sim \hat{P}_{\boldsymbol{\theta}_{i,j}^+} \\ y\sim P}}{\mathbb{E}}[k(x,y)] + \underset{\substack{x\sim \hat{P}_{\boldsymbol{\theta}_{i,j}^-} \\ y\sim P}}{\mathbb{E}}[k(x,y)]\Big], \tag{295}$$

with $\hat{P}_{\boldsymbol{\theta}_{i,j}^\pm}(x)$ being the probability to measure x from the VQC with parameters $\boldsymbol{\theta}$, but only the j-th occurrence of θ_i is shifted according to $\theta_i \pm \frac{\pi}{2}$.

Proof. Applying Lemma 3.22 to $\hat{P}_{\boldsymbol{\theta}}(x) = \langle 0|U(\boldsymbol{\theta})^\dagger \hat{O} U(\boldsymbol{\theta})|0\rangle$ with $\hat{O} = |x\rangle\langle x|$ leads to

$$\frac{\partial \hat{P}_{\boldsymbol{\theta}}(x)}{\partial \theta_i} = \frac{1}{2}\sum_{j=1}^m \Big(\hat{P}_{\boldsymbol{\theta}_{i,j}^+}(x) - \hat{P}_{\boldsymbol{\theta}_{i,j}^-}(x)\Big). \tag{296}$$

This identity can be inserted into Equation (284) and the finite sums can be swapped, which completes the proof. □

Thus, the use of shared weights leads to a sum over all occurrences for the particular parameter θ_i. The last step now is to combine Lemma 3.24 and Lemma 3.25 to get the final expression for the gradient of the MMD.

Theorem 3.5. Let $U(\boldsymbol{\theta})$ be a VQC where each θ_i affects at most m Pauli rotation gates according to $G_{i,j}(\alpha_{i,j}(\theta_i))$ with $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$, and all maps $\alpha_{i,j} : \mathbb{R} \to \mathbb{R}$ being continuously differentiable. The gradient of the MMD is then given by

$$\frac{\partial MMD^2[P, \hat{P}_{\boldsymbol{\theta}}]}{\partial \theta_i} = \sum_{j=1}^m \alpha_{i,j}'(\theta_i)\Big[\underset{\substack{x\sim \hat{P}_{\boldsymbol{\theta}_{i,j}^+} \\ y\sim \hat{P}_{\boldsymbol{\theta}}}}{\mathbb{E}}[k(x,y)] - \underset{\substack{x\sim \hat{P}_{\boldsymbol{\theta}_{i,j}^-} \\ y\sim \hat{P}_{\boldsymbol{\theta}}}}{\mathbb{E}}[k(x,y)] - \underset{\substack{x\sim \hat{P}_{\boldsymbol{\theta}_{i,j}^+} \\ y\sim P}}{\mathbb{E}}[k(x,y)] + \underset{\substack{x\sim \hat{P}_{\boldsymbol{\theta}_{i,j}^-} \\ y\sim P}}{\mathbb{E}}[k(x,y)]\Big], \tag{297}$$

with $\hat{P}_{\boldsymbol{\theta}_{i,j}^\pm}(x)$ being the probability to measure x from the VQC with parameters $\boldsymbol{\alpha}(\boldsymbol{\theta})$, but only the j-th occurrence of θ_i is shifted accordingly to $\alpha_{i,j}(\theta_i) \pm \frac{\pi}{2}$.

Proof. Applying the generalized parameter-shift rule (Theorem 3.4) leads to

$$\frac{\partial \hat{P}_{\boldsymbol{\theta}}(x)}{\partial \theta_i} = \frac{1}{2} \sum_{j=1}^{m} \alpha'_{i,j}(\theta_i) \Big(\hat{P}_{\boldsymbol{\theta}^+_{i,j}}(x) - \hat{P}_{\boldsymbol{\theta}^-_{i,j}}(x) \Big). \tag{298}$$

Substituting the partial derivative in Equation (284) and swapping the sums then yields the final formula for the gradient of the MMD. □

Therefore, the gradient of the MMD for a VQC with shared weights and constant factors can be calculated as a weighted sum over the probability distributions of the VQC with shifted parameters. Moreover, Theorem 3.5 is a generalization of Lemma 3.23, that handles the case for circuit architectures without shared weight and constant factors. This can be easily seen when choosing $\alpha_{i,j}(\theta_i) = \delta_{i,j} \cdot \theta_i$. Thus, $\alpha_{i,j}(\theta_i)' = \delta_{i,j}$ and the sum in Equation (297) reduces to the formula given in Lemma 3.23.
Our VQC for predictive models is of the form of $U(\overleftarrow{x}, \boldsymbol{\theta}) = V(\boldsymbol{\theta}) T(\boldsymbol{\theta}) D(\overleftarrow{x}, \boldsymbol{\theta})$ where $D(\overleftarrow{x}, \boldsymbol{\theta})$ is the data encoding unit, $T(\boldsymbol{\theta})$ the unitary that acts on memory states and $V(\boldsymbol{\theta})$ the quantum post-processing step. The data encoding unit as well as the post-processing step use Pauli rotation gates with shared parameters and constant factors to encode past observations $\overleftarrow{x}$. Thus, the probability distribution of the VQC is conditioned on the initialized past and so is the MMD. However, we can recover the conditional MMD via replacing the probability distributions $\hat{P}_{\boldsymbol{\theta}^\pm_{i,j}}(\cdot)$, $\hat{P}_{\boldsymbol{\theta}}(\cdot)$ and $P(\cdot)$ in Theorem 3.5 with their conditional counterparts $\hat{P}_{\boldsymbol{\theta}^\pm_{i,j}}(\cdot|\overleftarrow{x})$, $\hat{P}_{\boldsymbol{\theta}}(\cdot|\overleftarrow{x})$ and $P(\cdot|\overleftarrow{x})$, and set the mappings $\alpha_{i,j}$ according to the ansatz. We conclude that we can calculate the gradient of the conditional MMD via calculating a set of expectation values over the probability distribution of the VQC with shifted parameters.
The remaining part of the gradient of the cost function is the second term, which is the partial derivative of the probability distribution for measuring the memory register in the $|1\rangle$ state, i.e., $\partial P_{\boldsymbol{\theta}}(M_q = 1|\overleftarrow{x})/\partial \theta_i$. However, this can be calculated analogously when using the generalized parameter-shift rule, such that

$$\frac{\partial P_{\boldsymbol{\theta}}(M_q = 1|\overleftarrow{x})}{\partial \theta_i} = \frac{1}{2} \sum_{j=1}^{m} \alpha'_{i,j}(\theta_i) \Big(P_{\boldsymbol{\theta}^+_{i,j}}(M_q = 1|\overleftarrow{x}) - P_{\boldsymbol{\theta}^-_{i,j}}(M_q = 1|\overleftarrow{x}) \Big). \tag{299}$$

To end this chapter, we will briefly summarize the quantum learning algorithm in the next section.

3.6 Summary

Within this chapter, we have presented our proposed quantum learning algorithm for quantum approximate predictive models. This algorithm takes as input only a sample $x_{1:L} \in A^L$, drawn from a stationary stochastic process and the desired quantum model memory size $\hat{d}_q$. The model itself is given as a variational quantum circuit and is trained in a hybrid quantum-classical optimization procedure. To perform the optimization, we have proposed a cost function $C(\boldsymbol{\theta})$ which is based on the maximum mean discrepancy – a distance measure for probability distributions. Thus, the optimization aims to minimize the distance between the true underlying probability distribution P of the stochastic process and the model distribution $\hat{P}_{\boldsymbol{\theta}}$, which is given by the measurement of the auxiliary register A_q of the VQC. The introduction of a quantum post-processing step, together with a measurement of the memory register M_q, has led to a regularization term $R(\boldsymbol{\theta})$ for the cost function. This regularization term penalizes models with a rich set of internal states, which captures our observation that optimal models only have a minimal set of memory states. We have introduced a class of stochastic processes – the period-N uniform renewal processes – and constructed a q-simulator for the case of $N = 2$, i.e., a quantum exact predictive model. From here, we have gradually decomposed the circuit such that the result is a parameterized quantum circuit. This circuit was used as an ansatz for

the learning algorithm. To calculate the gradient of the cost function, we have introduced the common parameter-shift rule and extended it to be applicable for ansätze with shared weights and constant factors. This allowed us to derive expressions for the gradient that include the execution of the same VQC but with shifted parameters. Furthermore, to validate the learned models, we have applied the KL divergence and the TV distance to the domain of predictive models.
The algorithm itself consists of four steps, where the first one is the classical pre-processing step that takes the input sample and splits it into a training data set. Next, the VQC is initialized based on samples from the data set together with the trainable parameters, and executed several times. Both registers the memory and the auxiliary register are measured, where the outputs of the latter one are interpreted as the future samples and the outcome of the first one as the regularization term. Both pieces of information together form the cost function, for which the gradient is calculated next. The last step is to apply a classical gradient-based optimizer to update the parameters such that a next training iteration can be performed.
Now that we have presented the methodology of the quantum learning algorithm, we present the results for the numerical experiments in the next chapter.

4 Results

This chapter presents the results of the numerical experiments for the proposed learning algorithm applied to the period-2 uniform renewal process. We start in Section 4.1 with an overview of the setup for the numerical experiments. Next, in Section 4.2, we analyze the inherent stochastic error of the maximum mean discrepancy for different input sample lengths. This yields a measure of the expected best value of the cost function for a given length, and thus provides insight into how long the input sample should be. Afterward, we consider the learning stage of our algorithm in more detail in Section 4.3. Here, we analyze the value of the cost function for two versions of the algorithm, one with and one without the quantum post-processing step. Additionally, we take a look at the mean KL divergence and the full TV distance for one single time step and increasing number of training iterations. Next, in Section 4.4, we validate the trained models by using the same validation metrics as before but applying them to multiple future time steps. Furthermore, we present the learned memory states of the two models that performed best and compare them with each other.

4.1 Setup

All numerical experiments were performed using the quantum computing library Qiskit [40]. For the execution of quantum circuits, we chose a noise-free simulator such that the states of the quantum registers could be accessed directly. Thus, no measurements were needed and the exact values for the probability distributions $\hat{P}_{\boldsymbol{\theta}}$ and $P_{\boldsymbol{\theta}}$ could be read out of the quantum states. Therefore, the values for the gradient of the cost function were calculated exactly, which can be seen as the limit of performing infinitely many measurements of the quantum circuit.
For each run of the learning algorithm, we chose the initial parameters $\boldsymbol{\theta}^{(0)}$ uniformly at random from $(-\pi, +\pi)$, ran six different instances with a fixed number of iterations, and computed the mean for each training step. For the validation, we took the parameters $\boldsymbol{\theta}^*$ of the model for the training step with minimal overall cost, i.e., $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}^{(n)}} C(\boldsymbol{\theta}^{(n)})$.
To update the parameters, we used the gradient-based optimizer ADAM [36]. Recall that the gradient descent optimizer computes an update according to $\boldsymbol{\theta}^{(n+1)} = \boldsymbol{\theta}^{(n)} - \eta \cdot \nabla C(\boldsymbol{\theta}^{(n)})$ with a learning rate of $\eta > 0$. The ADAM optimizer follows the same rule but fine-tunes the learning rate adaptively via estimating the first and second moment of the gradient. We set the initial learning rate to $\eta = 0.02$ and took the heuristically suggested values from Kingma and Ba [36] for the remaining hyper-parameters.
The learning algorithm also includes the hyper-parameter $c \in \mathbb{R}_+$ to balance between the MMD and the regularization term $R_{\overline{x}}(\boldsymbol{\theta})$ in the cost function. We chose $c = 1$ for all runs, but note that this hyper-parameter could be further optimized.
For calculating the MMD, we chose the Gaussian kernels

$$k(x, y) = \frac{1}{n_\gamma} \sum_{i=1}^{n_\gamma} \exp\left(-\frac{|x-y|^2}{2\gamma_i}\right) \tag{300}$$

with bandwidths $\boldsymbol{\gamma} = [0.1, 0.5, 1.0, 5.0]$.
Before we present the results for the learning algorithm, we first discuss the inherent stochastic error of the MMD, which offers some insights into what we can ideally expect for values of the cost function.

4.2 Stochastic Error

Our proposed quantum learning algorithm requires as input a sample $x_{1:L} \in A^L$ drawn from a stochastic process. The sample should have a *reasonable* length such that all the necessary information describing the process can be captured from it. Practically, the length also depends on the ability of the learning algorithm to exploit the structure of the sample. Both quantities

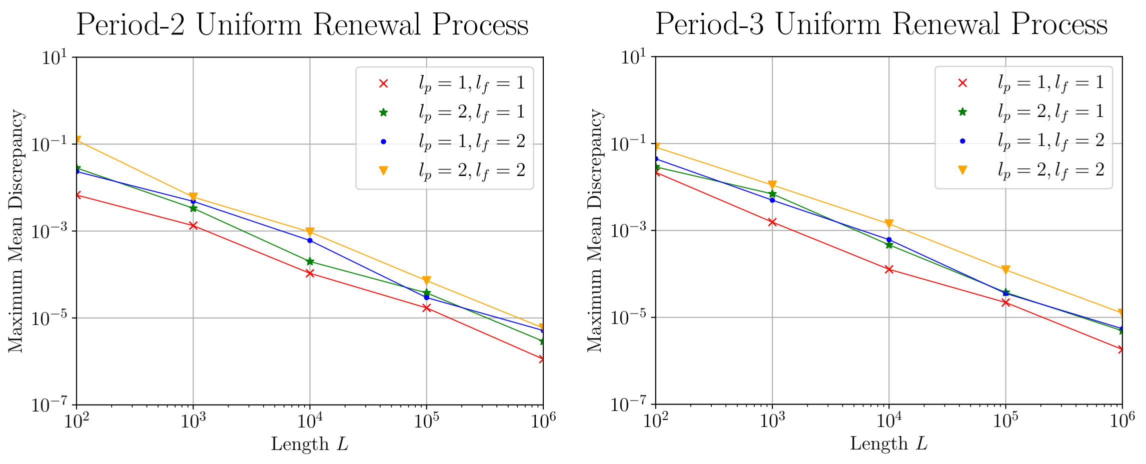

Figure 17: Stochastic error for the MMD of the period-2 (left) and period-3 (right) uniform renewal process. The plots show the MMD for different lengths $L \in \mathbb{N}$ and different choices $l_p, l_f \in \{1,2\}$ used for the classical pre-processing step.

are hard to measure in general and especially based only on a given sample without further knowledge of the underlying stochastic process. Therefore, we turn the question upside-down and ask what we can ideally expect for a given sample of fixed length. We answer this question as follows. Assume we are given a sample $x_{1:L} \in A^L$ of length $L \in \mathbb{N}$. The first step is to partition $x_{1:L}$ into two halves and perform the classical pre-processing step with both sub-samples, yielding two training data sets $\{(\overleftarrow{x}, \overrightarrow{x})\}$ and $\{(\overleftarrow{y}, \overrightarrow{y})\}$ with pasts and futures of lengths l_p and l_f, respectively. Then, we calculate the MMD based on these sets. Since the stochastic process is stationary, we know that both sub-samples are drawn from the same distribution P and the value of the MMD can thus be seen as an inherent stochastic error, i.e., the estimated distance between two identical distributions but based on a finite sample set.

The results are presented in Figure 17 where the MMD refers to the sum of the individual conditional MMD values, i.e. $\sum_{\overleftarrow{x}} w_{\overleftarrow{x}} MMD^2[P,P|\overleftarrow{x}]$. The two plots show the values for both, the period-2 and period-3 uniform renewal process, and for input lengths $L = 10^2$–10^6 as well as for different choices $l_p, l_f \in \{1,2\}$. For each data point, ten runs were performed and the mean values were calculated.

We first note that the stochastic error decreases with increasing sample length L. This is the case for both processes and all configurations of l_p, l_f. Furthermore, the configuration $l_p = l_f = 1$ shows the smallest error while the configuration $l_p = l_f = 2$ performs worst. The curves of the remaining two configurations are in between. For a length of $L = 10^6$, all configurations show a stochastic error of less than 10^{-5}, except the configuration $l_p = l_f = 2$ for the period-3 uniform renewal process, for which the value of the MMD is slightly higher.

The pre-processing step splits the input sample into two training data sets. For $l_p = l_f = 1$, the resulting data sets are the largest of all configurations. By contrast, the data sets for the choice of $l_p = l_f = 2$ are the smallest. For the period-2 uniform renewal process, we would expect that the stochastic error associated with a larger data set is smaller, which is in line with the numerical results.

The period-2 uniform renewal process has Markov order $\kappa = 1$ and thus the use of pasts with $l_p \geq 2$ might be superfluous, i.e., a waste of information. On the other hand, the period-3 uniform renewal process has Markov order $\kappa = 2$, yet the configuration with $l_p = 1$ shows the lowest stochastic error. This is counter-intuitive at first since we know that the distribution of one future time step depends on two past time steps. Thus, we would expect that the configuration with $l_p = 2, l_f = 1$ has the smallest stochastic errors. However, we calculate the MMD as a sum over the conditional MMDs for all possible pasts. Thus, when using data sets with $l_p = 1$, we essentially perform a pairwise comparison of probability distributions of the form of $P(\overrightarrow{x}|x_1)$,

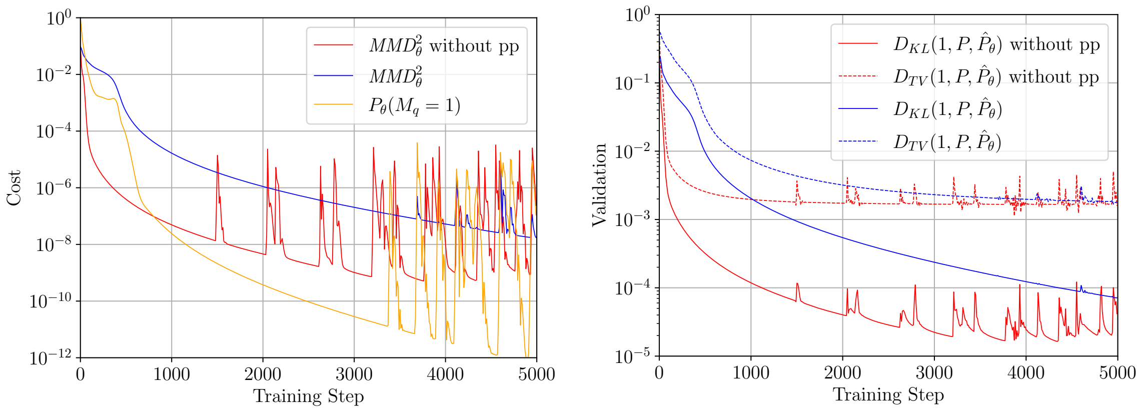

Figure 18: Left: The cost function as a function of the training steps. The MMD_θ^2 without pp refers to the version of the learning algorithm without the quantum post-processing step. For the version with the post-processing step, the components MMD_θ^2 and $P_{\boldsymbol{\theta}}(M_q = 1)$ are shown separately, while the weighted sum of both components would be the overall cost function. Right: The mean KL divergence and full TV distance for one future time step as a function of the training steps.

for a given $x_1 \in A$. Since both underlying distributions are identical, the MMD simply shows the smallest errors for the largest data sets, just as for the period-2 uniform renewal process. Yet, when training predictive models, we are interested in comparing distributions of the form of $P(\overrightarrow{x}|x_{0:\kappa})$ with Markov order κ. Thus, the length l_p should chosen to be at least as large as the Markov order to ensure that the conditional behavior is fully respected and the correct distributions are compared.
For the numerical experiments of the learning algorithm, we chose a configuration of $l_p = l_f = 1$ for the period-2 uniform renewal process. The choice of $l_p = 1$ is sufficient due to the Markov order and future samples of length $l_f = 1$ are sufficient in general as long as valid memory states are present.

4.3 Learning Stage

Within this section, we consider two versions of the quantum learning algorithm, both applied onto the period-2 uniform renewal process. The first one only takes the MMD as cost function, whereas the second one includes the regularization, i.e., the application of the quantum post-processing step. We refer to the first version as the learning algorithm *without pp* or *without regularization*. Figure 18 shows the value of the cost function as well as the mean KL divergence and the full TV distance for one future time step as a function over training steps. We start with the left figure, i.e., the trend of the cost function. Here, the MMD represents the sum over all possible pasts, i.e., $MMD_{\boldsymbol{\theta}}^2 = \sum_{\overleftarrow{x}} w_{\overleftarrow{x}} MMD^2[P, \hat{P}_{\boldsymbol{\theta}}|\overleftarrow{x}]$. For the version with the post-processing step, the regularization term $P_{\boldsymbol{\theta}}(M_q = 1)$ is shown separately.
Our first observation is that all measures start to oscillate after reaching a certain number of training steps. Here, the magnitude of the oscillation for the MMD without regularization is larger compared to the other one. While the MMD without regularization shows its first peak after 1,500 training steps, the MMD with regularization as well as the regularization term itself, starts to oscillate after 3,500 steps. Additionally, the regularization term shows the largest magnitude of oscillation.
Next, we observe that the learning algorithm without regularization decreases the cost function much faster. Already after 2,000 training steps, the value is smaller than 10^{-8}, while this is not the case for the version with regularization, even after 5,000 steps. Here, the value is at most smaller than 10^{-7}. On the other hand, the regularization term is even further decreased and

takes a value below 10^{-10} after 2,500 steps. It shows a minimum at around 4,800 iterations with a value of about 10^{-12}.
A possible explanation for the oscillations is the periodical nature of the rotation gates together with a high learning rate. Thus, if several parameters are updated based on ADAMs rule, the values could exceed the period, leading to an intermediate rise of the cost function. With an increasing number of training steps, the learning rate decreases such that the parameters are updated more carefully. Additionally, this could explain why the cost function between two consecutive peaks is further decreased. It should be noted that the same behavior has also been observed for training the quantum circuit Born machine [18].
The regularization term and the corresponding MMD show a small peak after 500 steps. This could indicate some minimum for the MMD, but a relatively large value for the regularization term. That means that the parameters corresponding to this minimum offer a good approximation of the very first future time step, but no valid set of memory states are present. Recap that we call the output states of the unitary T a valid set of memory states, if they match with the memory states generated by the data encoding unit E. The version without pp lacks this information in the cost function, and therefore advances further into this minimum of the MMD, while the version without pp is able to escape from it, advancing towards learning valid memory states.
It is particularly interesting that the MMD for both versions is decreased even below the stochastic error, which is around 10^{-5} for the used sample length of $L = 10^5$, see Section 4.2. One explanation could read as follows: a predictive model is trained given one input sample, which corresponds to one training data set $\{(\overleftarrow{x}, \overrightarrow{x})\}$. This data set offers an implicit probability distribution $P_D(\overrightarrow{x}|\overleftarrow{x})$, and when we calculate the MMD, we draw values from the data set according to this distribution. However, this distribution is fixed over the entire learning stage. Therefore, optimizing based on the MMD leads to fine-tuning the parameters $\boldsymbol{\theta}$ such that the model's distribution $\hat{P}_{\boldsymbol{\theta}}$ approximates P_D, rather than P, a phenomenon known as overfitting. Generally, we do not expect a model that has been trained beyond the stochastic error to perform better. On the contrary, we expect the validation metrics for these models to worsen since they approximate the data distribution well but not necessarily the true underlying process' distribution.

The plot on the right side of Figure 18 shows the mean KL divergence and the full TV distance for one future time step as a function of the training steps. We can see that both measures principally follow the curves of the MMD, i.e., the values of the metrics decrease with an increasing number of time steps and the values of the version without pp are smaller compared to their counterparts. Additionally, the values of the full TV distance are always larger compared to the mean KL divergence. Furthermore, we can observe peaks that correspond to the same number of training steps as for the MMD, but with a much smaller magnitude. The full TV distance approaches a value slightly above 10^{-3} for both versions, whereas the mean KL divergence is below 10^{-4}. Notably, the version without regularization offers a mean KL divergence which is almost one order of magnitude lower.
For calculating the validation metrics, the a priori unknown probability distribution P is used. Since the curves of the validation metrics decrease, this shows that the MMD can indeed be used as a cost function to train predictive models towards low distortions, which is in line with the observations of Liu et al. for the QCBM [18]. However, even for training steps that correspond to an MMD below the stochastic error, the mean KL divergence and full TV distance are further decreased. This contradicts our explanation of overfitting since we would expect the validation metrics to worsen or at least stop decreasing. We leave this question open for further investigation.
We conclude the discussion of the learning stage by noting that the version without regularization offers a smaller value for the MMD as well as for the mean KL and TV distance for one future time step. However, as we show in the next section, the regularization term is crucial to

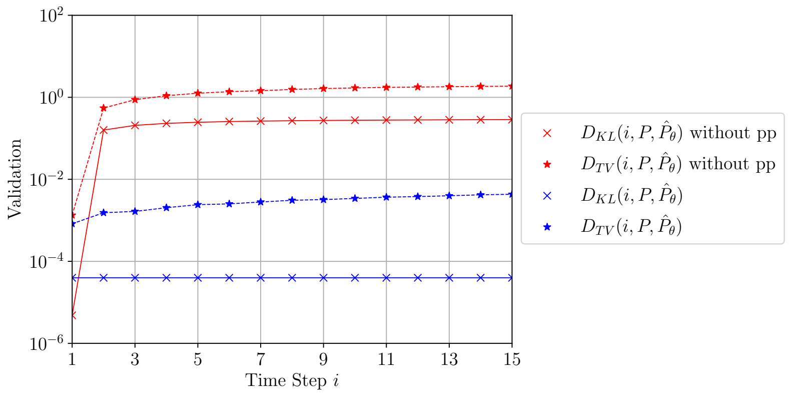

Figure 19: Mean KL divergence D_{KL} and full TV distance D_{TV} of the two models for the period-2 uniform renewal process. Measures without pp refer to the model that was trained without post-processing. The validation metrics compare the true underlying distribution of the process with the models distribution $\hat{P}_{\boldsymbol{\theta}}(X_{1:i}|\overleftarrow{x})$ for time steps $i \in \mathbb{N}$.

train predictive models that can simulate stochastic processes for multiple future time steps.

4.4 Validation

Validating our predictive models is essential for determining whether they can be used for simulating stochastic processes over multiple future time steps. Thus, the goal of this section is to present the validation results over multiple future time steps, where we again consider two models – one that was learned with the regularization term and one without. To this end, we consider two approaches; a quantitative approach in terms of the mean KL divergence and the full TV distance, and a qualitative approach in terms of visualizing the learned internal quantum states. We start with the quantitative results that are presented in Figure 19. Here, the two validation metrics are shown as a function of future time steps $i \in \mathbb{N}$, e.g., $D_{KL}(i, P, \hat{P}_{\boldsymbol{\theta}})$ shows the value of the mean KL divergence for sampling a future trajectory of length i, i.e., the distortion for the probability distribution $\hat{P}_{\boldsymbol{\theta}}(X_{1:i}|\overleftarrow{x})$.
Our first observation is that for the version without regularization, both metrics show a large jump from the first to the second time step and stay almost constant afterward. By contrast, for the model with regularization, the mean KL divergence is constant for all time steps, whereas the full TV distance is just slightly increasing. Here, the value does not exceed 10^{-2} over 15 time steps and the mean KL divergence is always below 10^{-4}. For the version without regularization, the full TV distance is larger than 10^{0} after three steps and the mean KL divergence shows values above 10^{-1} starting with the second step.
The curves of the metrics for the model without regularization indicate that the model is able to sample the very next future time step accurately. This means that if the unitary T acts on a memory state, it outputs quantum states that correspond to a good approximation of the underlying stochastic process. However, already for the second time step, the accuracy collapses entirely. This shows, that no valid set of memory states was found during the training since the unitary T would output correct future samples if it acted on a memory state. By contrast, the mean KL divergence of the model with regularization stays constant, indicating a valid set of memory states. While the full TV distance is slightly increasing, this is in line with our expectation, since it captures the accumulated absolute values of all errors without averaging over pasts or future time steps. However, the increase is very small, such that even large trajectories can be sampled with reasonable accuracy. Moreover, we expect the performance of the mean KL divergence to stay constant due to the same argument.

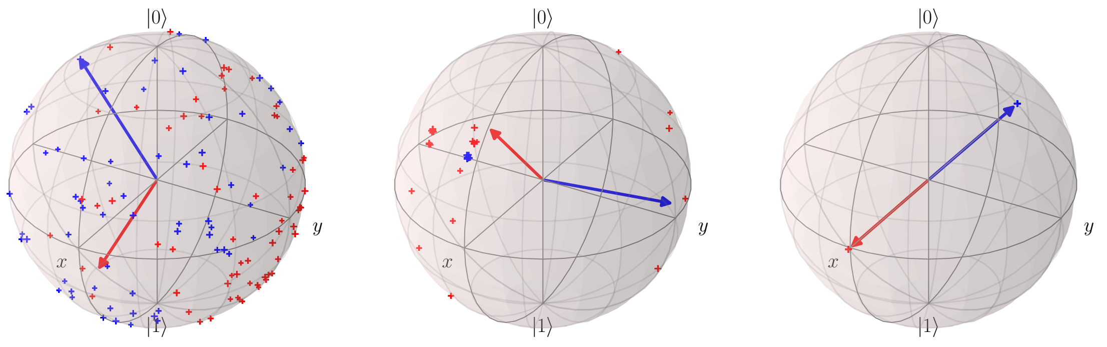

Figure 20: Visualization of the quantum states of the memory register of models for the period-2 uniform renewal process. The states of three models are shown: (left) a model with random parameters $\theta_i \in (-\pi, \pi)$, (center) the model trained without the regularization term, and (right) the model with the regularization term. The vectors point to the states prepared by the data encoding unit $E(\overleftarrow{x}, \boldsymbol{\theta})$. Points on the surface represent quantum states of M_q after the models have output $i \in \{1,2,3,4,5\}$ future time steps. All states are shown at once and the blue points and vectors are associated with the observation of $x_i = 0$ and $\overleftarrow{x} = 0$, respectively. Similarly, the red points and vectors refer to $x_i = 1$ and $\overleftarrow{x} = 1$.

While these quantitative measures indicate that the model with regularization has a valid set of memory states, we also show that this is indeed the case via visualizing the model's internal states. Since the memory register is a single qubit, we can represent these quantum states as a point on the Bloch sphere.
The results are shown in Figure 20, where we present three models: one with random parameters $\theta_i \in (\pi, \pi)$, one with optimized parameters but without the regularization, and one with the regularization. Here, the vectors are the two quantum states prepared by the data encoding unit $E(\overleftarrow{x}, \boldsymbol{\theta})$, i.e., the memory states of the model. The points on the surface are the internal quantum states of the memory register after generating $i = 1, 2, \ldots, 5$ future time steps. We performed $1{,}000$ runs for each past and each $i \in \{1,2,3,4,5\}$, and collected the resulting states. The states for all time steps and all runs are shown at once. Their colors indicate the expected states, i.e., the blue vector is pointing to the quantum state associated with $\overleftarrow{x} = 0$ and the blue points represent the quantum states after the last output was measured to be $x_i = 0$. Similarly, the red vector and points are associated with $x_i = 1$ and $\overleftarrow{x} = 1$.
The left picture of Figure 20 shows the result for the model with random parameters, where all states are randomly distributed along the sphere. By contrast, the trained model without regularization (center) shows much fewer states. Notably, the states associated with an observation of $x_i = 0$ concentrate, while the others are spread over. Additionally, the vectors do not point onto a state that was subsequently observed to appear again. For the model with regularization (right), all states are concentrated at two locations. Furthermore, these two locations coincide with the memory states generated by the data encoding unit.
For the model without regularization, the vectors do not point to an internal state. Thus, these states are not observed to appear again during the simulation of at least 5 future time steps. Therefore, the internal states learned by the unitary T are not a valid set of memory states. However, much fewer states are observed in general, which indicates that certain observations were not encountered during the simulation. This includes in particular trajectories with two consecutive zeros, e.g., “01001”, which are not valid outputs of the period-2 uniform renewal process. Although the states associated with $x_i = 0$ are very concentrated, they are not memory states, since they don't match with the states of the encoding unit E. By contrast, for the model with regularization, the states of the data encoding coincide with the model's internal states.

No other states were observed over 5 time steps and we conclude that this model offers a valid set of memory states.

5 Conclusion and Outlook

Within this work, we have developed a hybrid quantum-classical learning algorithm for approximate predictive models. The algorithm only requires one long sample drawn by a stochastic process as input. Classical resources are used to perform a pre-processing step and to update the training parameters. The quantum part is taken by a quantum circuit that represents a quantum approximate predictive model and generates future samples via quantum measurements. Our learning algorithm is memory efficient since the training is directly performed on a quantum computer and no classical representation of the quantum model is needed. Furthermore, the trained model is given as a quantum circuit and can thus be directly executed on a quantum computer for simulation purposes.
An essential part of the work was the definition of a suitable cost function. Inspired by the field of implicit generative modeling, we used the maximum mean discrepancy as a cost function but extended it to be applicable for predictive models. However, we have shown with numerical experiments that a cost function solely based on the MMD does not lead to a valid set of memory states. Furthermore, these models exhibit a bad simulation performance for future time steps $L \geq 2$, i.e., a KL divergence of more than 10^{-1} and a TV distance of more than 10^{0}. Therefore, we have added a regularization term to the cost function that penalizes models with a large set of memory states, and that successfully led to the training of a valid set of memory states. The regularization term itself is computed based on a quantum post-processing step, which is applied only during the learning stage. Our overall cost function successfully leads to a decrease of the KL divergence and the TV distance during learning the model.
Moreover, the learned models show a constantly good simulation performance over multiple future time steps. Here, the KL divergence remains constant at a value below 10^{-4} and the TV distance only slightly increases, but never exceeds a value of 10^{-2} over 15 future time steps.
As ansatz for the variational quantum circuit, we used a problem-inspired ansatz for supervised machine learning. To make sure that the ansatz contains an optimal solution, we have first constructed a quantum circuit that contains only static gates and that corresponds to a q-simulator, i.e., a quantum exact predictive model. Afterward, we have subsequently decomposed the gates involved into rotation and *CNOT* gates, such that the output is a parameterized quantum circuit. We have filled up this PQC with additional rotation gates such that it follows the same structure as commonly used ansätze for supervised machine learning.
Furthermore, to utilize gradient-based optimizers, we have generalized the common parameter-shift rule for ansätze with shared weights and constant factors. Based on that, we have derived a compact expression for the gradient of the MMD with respect to these ansätze. Beyond the current work, this result can also be used to construct quantum circuit Born machines with more complicated ansätze.

While we have successfully shown that the learning algorithm works for a specific process, namely the period-2 uniform renewal process, it remains to be shown that it also performs well for other stationary stochastic processes. The current version of the algorithm, however, has two caveats. On the one hand, only past samples of length $l_p = 1$ can be encoded, and on the other hand, only processes with a binary alphabet $A = \{0, 1\}$ can be used. This lies in the fact that the proposed data encoding unit, and thus the post-processing step, are specifically designed to work with single bits. Yet, this can be overcome by applying the data encoding unit multiple times sequentially, with each unit offering two independent trainable parameters. Thus, if an observation $x \in \mathbb{N}_0$ needs to be encoded, its binary representation can be used and each bit can be encoded individually. Furthermore, past samples $\overleftarrow{x}$ of length $l_p > 1$ can be encoded the same way. This, however, needs to be respected in the post-processing step as well. On the one hand, the post-processing step then also corresponds to a sequence of inverse data encoding units, but on the other hand, a single observation $x \in \mathbb{N}_0$ is no longer sufficient to apply the correct inverse encoding. In general, the post-processing step can still be performed

by post-selecting the events that correspond to a valid run but again losing some precision for the regularization term. Another way is to perform the same number of future time steps as the encoding length l_p, and let the inverse encoding units be controlled by these additional auxiliary qubits.
Additionally, we want to note that the MMD in the cost function can, in principle, be replaced by any other distance measure for probability distributions. The only requirement is that training based on this cost function leads to a reasonable accuracy for approximating the underlying probability distribution of the stochastic process. One possibility is to use the recently proposed Sinkhorn divergence [37], for which it is shown that in some situations, it can better minimize the TV distance compared to the MMD [20].

While previous works on quantum predictive models were more focused on their theoretical advantage over classical ones [7, 12, 32, 33], this work shows a practical approach to how these models can be learned on quantum computers and how they can be used for simulation purposes. Moreover, the proposed quantum learning algorithm opens a way for how a quantum advantage can be rigorously shown. This can be done by calculating a lower bound on the distortions of any classical model via the methods developed by Yang et al. [11] and comparing them to models learned by our algorithm. However, for this purpose, the period-2 uniform renewal process is insufficient, since it has a classical topological complexity of $d_c < 1$ and a single qubit is therefore enough to exactly simulate the process. Thus, the next natural step is to learn a model for the period-3 uniform renewal process, which has a topological complexity of $d_c > 1$, i.e., any classical predictive model for this process with only one bit as memory exhibits distortions. A quantum approximate model learned by our algorithm thus might perform better and offers a provably better accuracy compared to any classical model. Furthermore, this advantage could be shown on near-term quantum devices already.

References

[1] Cosma Rohilla Shalizi and James P. Crutchfield. “Computational Mechanics: Pattern and Prediction, Structure and Simplicity”. In: *Journal of Statistical Physics* 104.3 (Aug. 2001), pp. 817–879. DOI: 10.1023/A:1010388907793.

[2] Sarah E. Marzen and James P. Crutchfield. “Informational and Causal Architecture of Discrete-Time Renewal Processes”. In: *Entropy* 17.7 (2015), pp. 4891–4917. DOI: 10.3390/e17074891.

[3] Robert Haslinger, Kristina Lisa Klinkner, and Cosma Rohilla Shalizi. “The computational structure of spike trains”. In: *Neural computation* 22.1 (Jan. 2010), pp. 121–157. DOI: 10.1162/neco.2009.12-07-678.

[4] Joongwoo Brian Park et al. “Complexity analysis of the stock market”. In: *Physica A: Statistical Mechanics and its Applications* 379.1 (2007), pp. 179–187. DOI: 10.1016/j.physa.2006.12.042.

[5] Jae-Suk Yang et al. “Increasing market efficiency in the stock markets”. In: *The European Physical Journal B* 61.2 (Jan. 2008), pp. 241–246. DOI: 10.1140/epjb/e2008-00050-0.

[6] Alexander B Boyd, Dibyendu Mandal, and James P Crutchfield. “Identifying functional thermodynamics in autonomous Maxwellian ratchets”. In: *New Journal of Physics* 18.2 (Feb. 2016), p. 023049. DOI: 10.1088/1367-2630/18/2/023049.

[7] Jayne Thompson et al. “Causal Asymmetry in a Quantum World”. In: *Phys. Rev. X* 8 (3 June 2018), p. 031013. DOI: 10.1103/PhysRevX.8.031013.

[8] James P. Crutchfield and Karl Young. “Inferring statistical complexity”. In: *Phys. Rev. Lett.* 63 (2 June 1989), pp. 105–108. DOI: 10.1103/PhysRevLett.63.105.

[9] James P. Crutchfield. “The calculi of emergence: computation, dynamics and induction”. In: *Physica D: Nonlinear Phenomena* 75.1 (1994), pp. 11–54. DOI: https://doi.org/10.1016/0167-2789(94)90273-9.

[10] Cosma Rohilla Shalizi and Kristina Lisa Shalizi. “Blind Construction of Optimal Nonlinear Recursive Predictors for Discrete Sequences”. In: *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence.* UAI ’04. Banff, Canada: AUAI Press, 2004, pp. 504–511. ISBN: 0974903906.

[11] Chengran Yang et al. *Provable superior accuracy in machine learned quantum models.* 2021. arXiv preprint: 2105.14434v2.

[12] Felix C. Binder, Jayne Thompson, and Mile Gu. “Practical Unitary Simulator for Non-Markovian Complex Processes”. In: *Physical Review Letters* 120.24 (June 2018). DOI: 10.1103/physrevlett.120.240502.

[13] Alberto Peruzzo et al. “A variational eigenvalue solver on a photonic quantum processor”. In: *Nature Communications* 5.1 (July 2014), p. 4213. ISSN: 2041-1723. DOI: 10.1038/ncomms5213.

[14] P. J. J. O’Malley et al. “Scalable Quantum Simulation of Molecular Energies”. In: *Phys. Rev. X* 6 (3 July 2016), p. 031007. DOI: 10.1103/PhysRevX.6.031007.

[15] Abhinav Kandala et al. “Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets”. In: *Nature* 549.7671 (Sept. 2017), pp. 242–246. ISSN: 1476-4687. DOI: 10.1038/nature23879.

[16] J. S. Otterbach et al. *Unsupervised Machine Learning on a Hybrid Quantum Computer.* 2017. arXiv preprint: 1712.05771.

[17] Ryan Sweke, Jean-Pierre Seifertand Dominik Hangleiter, and Jens Eisert. “On the Quantum versus Classical Learnability of Discrete Distributions”. In: *Quantum* 5 (Mar. 2021), p. 417. DOI: 10.22331/q-2021-03-23-417.

[18] Jin-Guo Liu and Lei Wang. “Differentiable learning of quantum circuit Born machines”. In: *Phys. Rev. A* 98 (6 Dec. 2018), p. 062324. DOI: 10.1103/PhysRevA.98.062324.

[19] Arthur Gretton et al. “A Kernel Two-Sample Test”. In: *Journal of Machine Learning Research* 13.25 (2012), pp. 723–773. URL: http://jmlr.org/papers/v13/gretton12a.html.

[20] Brian Coyle et al. “The Born supremacy: quantum advantage and training of an Ising Born machine”. In: *npj Quantum Information* 6.1 (July 2020), p. 60. DOI: 10.1038/s41534-020-00288-9.

[21] Maria Schuld et al. “Evaluating analytic gradients on quantum hardware”. In: *Phys. Rev. A* 99 (3 Mar. 2019), p. 032331. DOI: 10.1103/PhysRevA.99.032331.

[22] Bert Fristedt and Lawrence Gray. *A Modern Approach to Probability Theory*. Cambridge, MA: Birkhäuser Boston, 1997. ISBN: 978-1-4899-2837-5. DOI: 10.1007/978-1-4899-2837-5.

[23] Sheldon M. Ross. *Introduction to Probability Models*. 11th. San Diego, CA: Academic Press, 2014. ISBN: 978-0-12-407948-9. DOI: 10.1016/C2012-0-03564-8.

[24] Olav Kallenberg. *Foundations of modern probability*. Second. Probability and its Applications (New York). Springer-Verlag, New York, 2002. ISBN: 0-387-95313-2. DOI: 10.1007/978-1-4757-4015-8.

[25] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. ISBN: 978-1107002173. DOI: 10.1017/CBO9780511976667.

[26] John Preskill. “Quantum Computing in the NISQ era and beyond”. In: *Quantum* 2 (Aug. 2018), p. 79. DOI: 10.22331/q-2018-08-06-79.

[27] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. *A Quantum Approximate Optimization Algorithm*. 2014. arXiv preprint: 1411.4028.

[28] Marcello Benedetti et al. “Parameterized quantum circuits as machine learning models”. In: *Quantum Science and Technology* 4.4 (Nov. 2019), p. 043001. DOI: 10.1088/2058-9565/ab4eb5.

[29] M. Cerezo et al. “Variational quantum algorithms”. In: *Nature Reviews Physics* 3.9 (Sept. 2021), pp. 625–644. DOI: 10.1038/s42254-021-00348-9.

[30] Michael J. Bremner, Richard Jozsa, and Dan J. Shepherd. “Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 467.2126 (Aug. 2010), pp. 459–472. DOI: 10.1098/rspa.2010.0301.

[31] Aram W. Harrow and John C. Napp. “Low-Depth Gradient Measurements Can Improve Convergence in Variational Hybrid Quantum-Classical Algorithms”. In: *Phys. Rev. Lett.* 126 (14 Apr. 2021), p. 140502. DOI: 10.1103/PhysRevLett.126.140502.

[32] Farzad Ghafari et al. “Dimensional Quantum Memory Advantage in the Simulation of Stochastic Processes”. In: *Phys. Rev. X* 9 (4 Oct. 2019), p. 041013. DOI: 10.1103/PhysRevX.9.041013.

[33] Mile Gu et al. “Quantum mechanics can reduce the complexity of classical models”. In: *Nature Communications* 3.1 (Mar. 2012), p. 762. DOI: 10.1038/ncomms1761.

[34] Mohamed Shakir and Lakshminarayanan Balaji. “Learning in Implicit Generative Models”. In: (Oct. 2016). arXiv preprint: 1610.03483.

[35] Song Cheng, Jing Chen, and Lei Wang. “Information Perspective to Probabilistic Modeling: Boltzmann Machines versus Born Machines”. In: *Entropy* 20.8 (2018). DOI: 10.3390/e20080583.

[36] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization.* 2017. arXiv preprint: 1412.6980.

[37] Aude Genevay, Gabriel Peyre, and Marco Cuturi. “Learning Generative Models with Sinkhorn Divergences”. In: *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics.* Ed. by Amos Storkey and Fernando Perez-Cruz. Vol. 84. Proceedings of Machine Learning Research. PMLR, Apr. 2018, pp. 1608–1617.

[38] Holger Wendland. *Scattered Data Approximation.* Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2004. DOI: 10.1017/CBO9780511617539.

[39] Ingo Steinwart. “On the Influence of the Kernel on the Consistency of Support Vector Machines”. In: *Journal of Machine Learning Research* 2 (2001), pp. 67–93.

[40] MD SAJID ANIS et al. *Qiskit: An Open-source Framework for Quantum Computing.* 2021. DOI: 10.5281/zenodo.2573505.