

# Experimenting with machine learning algorithms on quantum computers

Nagy Dániel<sup>1</sup>

<sup>1</sup>Institute for Physics, Eötvös Loránd University, H-1117, Pázmány Péter sétány 1/A.  
Budapest, Hungary

Created on December 1, 2019

Last update: May 6, 2021

## **Abstract**

A modern számítástechnika jelentős eredményei közé tartozik a gépi tanulás és mesterséges intelligencia alapvető algoritmusainak kifejlesztése és ezek hasznosságának tesztelése különböző feladatokon. Ugyanakkor az elmúlt években a kvantumszámítás is jelentős fejlődésen ment keresztül, olyannyira, hogy 2019-ben a Google kísérleti csapatának sikerült demonstrálnia a kvantumfölényt. A munka során megvizsgáljuk a két terület átfedéséből származó lehetőségeket: klasszikus adatok kvantumozását illetve a klasszikus gépi tanulás segítségével történő kvantumozás hibajavítást.

# Gépi tanulási algoritmusok vizsgálata kvantumszámítógépeken

Nagy Dániel<sup>1</sup>

<sup>1</sup>Eötvös Loránd Tudományegyetem, Fizika Intézet, H-1117, Pázmány Péter sétány 1/A.  
Budapest, Magyarország

Elkezdve: December 1, 2019

Utolsó frissítés: 2021. május 6.

## **Kivonat**

A modern számítástechnika jelentős eredményei közé tartozik a gépi tanulás és mesterséges intelligencia alapvető algoritmusainak kifejlesztése és ezek hasznosságának tesztelése különböző feladatokon. Ugyanakkor az elmúlt években a kvantumszámítás is jelentős fejlődésen ment keresztül, olyannyira, hogy 2019-ben a Google kísérleti csapatának sikerült demonstrálnia a kvantumfölényt. A munka során megvizsgáljuk a két terület átfedéséből származó lehetőségeket: klasszikus adatok kvantumozását illetve a klasszikus gépi tanulás segítségével történő kvantumozás hibajavítást.

# Contents

<b>1</b>	<b>Introduction</b>	<b>8</b>
<b>2</b>	<b>Quantum Computing with continuous variables</b>	<b>8</b>
<b>3</b>	<b>Machine learning</b>	<b>10</b>
3.1	Supervised Classification and Regression . . . . .	11
3.1.1	Classification . . . . .	11
3.1.2	Regression . . . . .	12
3.2	Deep Neural Networks and backpropagation . . . . .	13
<b>4</b>	<b>Quantum Machine Learning</b>	<b>13</b>
4.1	Parametric quantum circuits . . . . .	14
4.2	Variational Quantum Eigensolver . . . . .	14
4.3	Calculating the gradients of the parameters . . . . .	14
<b>5</b>	<b>Results</b>	<b>15</b>
5.1	Regression . . . . .	15
5.2	Classification . . . . .	15
5.3	CV-VQE for the Bose-Hubbard model . . . . .	17
5.3.1	Solving the model by exact diagonalization . . . . .	17
5.3.2	Variational solutions . . . . .	18
<b>A</b>	<b>Coherent states</b>	<b>23</b>
<b>B</b>	<b>Mathematical preliminaries</b>	<b>24</b>
B.1	Hilbert spaces . . . . .	24
B.2	Linear operators on Hilbert spaces . . . . .	26
B.3	Hermitian Operators, Unitary Operators, Spectral theorem, Hadamard-lemma . . . . .	26

---

B.4 Pure and mixed quantum states . . . . .	28
---	----

# List of Figures

1	.....	15
2	.....	15
3	leiras .....	16
4	.....	16
5	(a) Visual representation of a sparse Hamiltonian matrix $H_{jj'}$ for a Bose-Hubbard model with $\dim(\mathcal{H}) = 35$ . Darker points show the non-zero elements in the matrix (b) Numerical solutions of the ground-state energy for different $U$ and $t$ values. ....	18
6	Illustration of a variational ansatz circuit with its first layer. Initially, each of the four qumodes contain a single photon. The layer is built up from Kerr-gates, CrossKerr gates and Beamsplitters, which are passive optical gates, so the total number of photons is preserved. ....	19
7	.....	19
8	caption .....	20

# 1 Introduction

Quantum computing is probably the most promising emerging technology with many possible applications across all domains of science and business. The idea of quantum computing was first proposed by Richard P. Feynman as a method to simulate quantum mechanics. Since the size of the Hilbert-space grows exponentially with the complexity of the quantum system, and thus the calculations become intractable on any classical computer, Feynmans idea was to simulate quantum physics on devices that behave themselves according to the rules of quantum physics.

Soon after Feynmans proposal, scientists became to establish the theoretical background of quantum information and quantum computing. Some of the quantum algorithms are proven to have an advantage over any known classical algorithm. One of the first such quantum algorithm is the Deutsch-Jozsa algorithm [1], which decides if a binary function  $f$  is balanced or constant. Probably the most notable quantum algorithm was proposed by Peter W. Shor in 1994, which is a polynomial-time quantum algorithm for factoring large integers [2]. This is the first quantum algorithm with a real-life application, because most of the public-key cryptosystems could be broken with an efficient algorithm for integer factoring. Another important quantum algorithm is the Grover's algorithm proposed by Lov K. Grover in 1996 for searching an unordered database [3]. While the best known classical algorithm for searching unordered databases runs in  $O(N)$  time, Grover's algorithm solves this problem with  $O(\sqrt{N})$  oracle queries. Beyond quantum algorithms, many quantum-inspired algorithms were discovered [4, 5, 6].

# 2 Quantum Computing with continuous variables

**[Dani: cite: Xanadu paper, Photonic Supremacy paper, PsiQuantum Fusion-Based QC]**

Continuous variable (CV) systems have infinite degrees of freedom, and therefore the corresponding Hilbert-space is of infinite dimensions. Such systems can be modeled by  $M$  harmonic oscillators, which are usually called *modes*. With  $H_k$  denoting the Hilbert-space of the  $k$ th mode, the Hilbert-space of the full CV system is their direct sum:

$$\mathcal{H} = \bigoplus_{k=1}^M \mathcal{H}_k. \quad (1)$$

The Hamiltonian of mode  $j$  can be described by introducing bosonic creation and annihilation operators  $a_j^\dagger$  and  $a_j$ , obeying the following canonical commutation relations:

$$[\hat{a}_j, \hat{a}_k^\dagger] = \delta_{jk}, [\hat{a}_j^\dagger, \hat{a}_k^\dagger] = [\hat{a}_j, \hat{a}_k] = 0. \quad (2)$$

The effect of the creation and annihilation operators on the Fock space can be described by the following identities:

$$\begin{aligned} \hat{a}_k^\dagger |n_1, \dots, n_k, \dots\rangle &= \sqrt{1 + n_k} |n_1, \dots, n_k + 1, \dots\rangle \\ \hat{a}_k |n_1, \dots, n_k, \dots\rangle &= \sqrt{n_k} |n_1, \dots, n_k - 1, \dots\rangle \end{aligned} \quad (3)$$



We also introduce the number operator  $\hat{n}_j = \hat{a}_j^\dagger \hat{a}_j$ :

$$\hat{a}_k^\dagger \hat{a}_k |n_1, \dots, n_k, \dots\rangle = \hat{n}_k |n_1, \dots, n_k, \dots\rangle = n_k |n_1, \dots, n_k, \dots\rangle \quad (4)$$

Each of the single-mode Hilbert-space  $\mathcal{H}_k$  is an infinite-dimensional Fock-space spanned by the eigenvectors of the number operator  $\hat{n}_k = \hat{a}_k^\dagger \hat{a}_k$  labeled  $\{|n\rangle\}_k$ . It is often useful to introduce the canonical position operators  $X_j$  and momentum operators  $P_j$  which are related to the creation and annihilation operators by the following transformations:

$$\hat{X}_k = \frac{\hat{a}_k + \hat{a}_k^\dagger}{\sqrt{2}} \quad \hat{a}_k = \frac{\hat{X}_k + i\hat{P}_k}{\sqrt{2}} \quad (5)$$

$$\hat{P}_k = \frac{\hat{a}_k - \hat{a}_k^\dagger}{i\sqrt{2}} \quad \hat{a}_k^\dagger = \frac{\hat{X}_k - i\hat{P}_k}{\sqrt{2}} \quad (6)$$

Operators  $\hat{X}_k$  and  $\hat{P}_k$  are called quadrature operators. Following the notation often used in literature, we group these operators into a vector

$$\mathbf{R}^\top = (\hat{X}_1, \hat{P}_1, \hat{X}_2, \hat{P}_2, \dots, \hat{X}_k, \hat{P}_k, \dots, \hat{X}_M, \hat{P}_M) \quad (7)$$

We can calculate the commutators  $[\mathbf{R}_k, \mathbf{R}_l]$  by introducing a symplectic form  $\mathbf{\Omega}$  as

$$\mathbf{\Omega} = \bigoplus_{j=1}^M \boldsymbol{\omega}, \quad \boldsymbol{\omega} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}. \quad (8)$$

Using this notation, we can write

$$[\mathbf{R}_k, \mathbf{R}_l] = i\mathbf{\Omega}_{kl} \quad (9)$$

Apart from the Fock-basis, there is another useful alternative basis, the basis of *coherent states*, which are eigenstates of the annihilation operator  $\hat{a}_k$ :

$$\hat{a}_k |\alpha\rangle_k = \alpha |\alpha\rangle_k, \quad (10)$$

with  $\alpha \in \mathbb{C}$ . From definition this, we can derive that the Fock-basis expansion of the coherent state  $|\alpha\rangle_k$  is

$$|\alpha\rangle_k = e^{-\frac{1}{2}|\alpha|^2} \sum_{n=1}^{\infty} \frac{\alpha^n}{\sqrt{n!}} |n\rangle_k. \quad (11)$$

The derivation of the above formula is explained in detail in appendix A. By introducing the single-mode Weyl-displacement operator

$$\hat{D}_k(\alpha) = e^{\alpha \hat{a}_k^\dagger - \alpha^* \hat{a}_k}, \quad (12)$$

we can create coherent states from the vacuum state:  $|\alpha\rangle_k = \hat{D}_k(\alpha) |0\rangle_k$ . Of course, we can define multimode coherent states

$$|\alpha_1\rangle_1 \otimes |\alpha_2\rangle_2 \otimes \dots \otimes |\alpha_N\rangle_N \equiv |\boldsymbol{\xi}\rangle = \hat{D}(\boldsymbol{\xi}) |0\rangle, \quad (13)$$

where  $\hat{D}(\boldsymbol{\xi})$  is the multi-mode Weyl-displacement operator defined as

$$\hat{D}(\boldsymbol{\xi}) = e^{i\mathbf{R}^\top \boldsymbol{\Omega} \boldsymbol{\xi}}, \quad \boldsymbol{\xi} \in \mathbb{R}^{2N}. \quad (14)$$

For a single mode,

$$\hat{D}_k(\boldsymbol{\xi}) = \hat{D}_k \begin{pmatrix} \xi_1 \\ \xi_2 \end{pmatrix} = e^{i(\xi_2 \hat{X}_k - \xi_1 \hat{P}_k)}, \text{ if we set } \begin{pmatrix} \xi_1 \\ \xi_2 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} \text{Re}(\alpha_k) \\ \text{Im}(\alpha_k) \end{pmatrix} \quad (15)$$

[Dani: Write about phase-space, quasiprobability] [Dani: Write about different quantum gates.] [Dani: Write about homodyne measurements, heterodyne measurements and photon number-resolving measurements]

### 3 Machine learning

Machine learning is simply said, a new paradigm in software development. Traditional software have a predefined behaviour, which does not change over time. In contrast, machine learning models are iteratively "trained" i.e. updated to achieve better performance by the use of data. These algorithms are designed to learn patterns from the data available and update themselves with new experience. Although, there are a lot of machine learning algorithms, they usually fit into one of the following three main classes:

- **Supervised Learning.** Supervised Learning (SL) algorithms need labeled data, and based on the input-output pairs previously seen by the algorithm, it learns to produce correct outputs for previously unseen inputs. A very basic example of supervised learning is the linear regression or support vector machines [Dani: ref?] for classification, but there are a lot of complex models like CNNs for image classification [Dani: ref?], which are trained in a supervised manner.
- **Unsupervised Learning.** Unsupervised Learning (UL) does not require labeled data, instead of that, it is trying to learn from a lot of unlabeled data points. These algorithms include a bunch of clustering methods, like k-means [Dani: ref?], or mixture models [Dani: ref?]. Another example of unsupervised learning is latent variables or distributions for feature extraction or data denoising. A great example of latent distribution learning are Variational Autencoders [7] which have been adopted for a variety of tasks.
- **Reinforcement Learning.** Reinforcement Learning (RL) is very different from the previously mentioned two classes of machine learning method. In RL the goal is not to learn from a given set of data, but rather to solve a specific task in an environment. In RL, an *agent* learns to perform a task by observing the state (or some subspace of the state) of the surrounding environment, and executing actions in order to maximize an objective. RL is inspired by the process of learning from mistakes or rewards. The environment in RL can be any type of physical simulation, or even a computer game. Examples for RL algorithms include Deep Q-networks [8] or AlphaGo [9]

## 3.1 Supervised Classification and Regression

The two most common SL tasks are **classification** and **regression**. Both classification and regression can be performed with traditional algorithms or deep neural networks. In this chapter we present the basic ideas and formalisms of classification and regression which will be useful in the following chapters.

In supervised learning the data points consist of pairs  $(\mathbf{x}_i, \mathbf{y}_i)$ , where  $\mathbf{x}_i$  are called feature vectors and  $\mathbf{y}_i$  are called labels. Both the feature vectors and the labels are real-valued vectors of any dimension. Feature vectors can be pre-processed to achieve better performance. A dataset is usually split into two or three parts. The first part is usually the largest, which is called the training set, and this is used to train the model. The second dataset is called testing or evaluation set and is used to evaluate how the model performs when it is fed with previously unseen data. There might be an additional set called the validation set, which is used to verify the model's performance after training is finished. While the testing set is used to monitor the learning process during training, the validation set is used only after the full training process is finished.

### 3.1.1 Classification

In classification, each point  $\mathbf{x}$  is in one of  $K$  predefined categories. That is, the labels are integers less or equal to  $K$ :  $y_i \in \{1, \dots, K\}$ . The goal of a classifier is to learn the connection between  $\mathbf{x}_i$  and  $y_i$ , in other words, to learn, how to put each vector  $\mathbf{x}_i$  into the right category. The performance of a classifier is usually measured with categorical accuracy i.e. what percentage of the predicted labels are in the right category. Apart from accuracy, it is often useful to introduce the categorical cross-entropy. In general, the crossentropy of two probability distributions  $p$  and  $q$  is defined as

$$H(p, q) = - \int p(x) \log q(x) dx \quad (16)$$

**[Dani: Explain the intuition behind entropy, cross-entropy etc.]** To transform this general formula of cross-entropy to the case of classification, we need to introduce one-hot encoding. One-hot encoding maps integer labels to binary vectors in a way that vector corresponding to the label  $j$  will contain 1 at the  $j$ th position and 0 everywhere else. The length of the resulting vectors is  $K$ . For example the one-hot encoding of labels 1 and 3 in case of three categories would look like

$$\text{OneHot}(1, 3) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad \text{OneHot}(3, 3) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (17)$$

In machine learning, especially in deep learning one-hot encoding of categorical features or labels is proven to be very useful. Now, with one-hot encoded labels, we can easily define the categorical

crossentropy:

$$\text{CrossEntropy}(\mathbf{y}^{\text{pred}}, \mathbf{y}^{\text{true}}) = - \sum_{i=1}^K y_i^{\text{true}} \log y_i^{\text{pred}} \quad (18)$$

The crossentropy essentially measures how the predicted distribution of labels differs from the true distribution, hence it is widely used in machine learning. Applications of classification include image recognition, image segmentation, speech and voice recognition, fraud detection and a lot more.

### 3.1.2 Regression

Regression models are usually used for forecasting and prediction, or simply to uncover the relationships between two or more features. Opposed to classification, in regression, the output of the model is not restricted to a discrete set of predefined classes, instead the set of possible outputs is a bounded or unbounded subset of  $\mathbb{R}^d$ . Therefore, classification is a special case of regression where the set of possible values of the outcomes is a finite set of integers. Apart from this difference, the purpose of regression is the same: to find a connection between the features  $\mathbf{x}$  and labels  $\mathbf{y}$ . The elements of feature vectors  $x_i$  are usually called independent variables or predictors, while the elements of the label vectors  $y_i$  are called dependent variables. Formally the regression model is a function approximator, which is itself a function that can be parametrized by a set of real parameters  $\boldsymbol{\theta} \in \mathbb{R}^m$ . Given a set of observations  $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}$  a function approximator  $f(\cdot; \boldsymbol{\theta})$ , we can write

$$y_i = f(\mathbf{x}_i; \boldsymbol{\theta}) + \epsilon_i, \quad (19)$$

Where  $\epsilon_i$  denotes a random statistical noise, usually assumed to follow Gaussian distribution. The goal is to find an optimal set of parameters  $\boldsymbol{\theta}^*$  which minimizes the mean squared error:

$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta}} \sum_{k=1}^N (y_k - f(\mathbf{x}_k; \boldsymbol{\theta}))^2 \quad (20)$$

The mean squared error is itself a good measure of the performance of the regression model and is often used in machine learning and deep learning. Another scalar value for getting some information about the goodness of fit is the coefficient of determination or often called  $R^2$ -score. The  $R^2$ -score is a real value usually falling between zero and one (the latter meaning perfect fit), although sometimes it can be negative. The  $R^2$ -score measures the proportion of variance in the predictions that can be explained by the data. To calculate the  $R^2$ -score, we first introduce the total sum of squares

$$SS_{\text{tot}} = \sum_i (y_i^{\text{true}} - \langle y^{\text{true}} \rangle)^2, \quad (21)$$

where  $\langle y^{\text{true}} \rangle = \frac{1}{N} \sum_k y_k^{\text{true}}$  is the mean of the dependent variable in the dataset. Then we calculate the sum of squared residuals, or sum of squared errors:

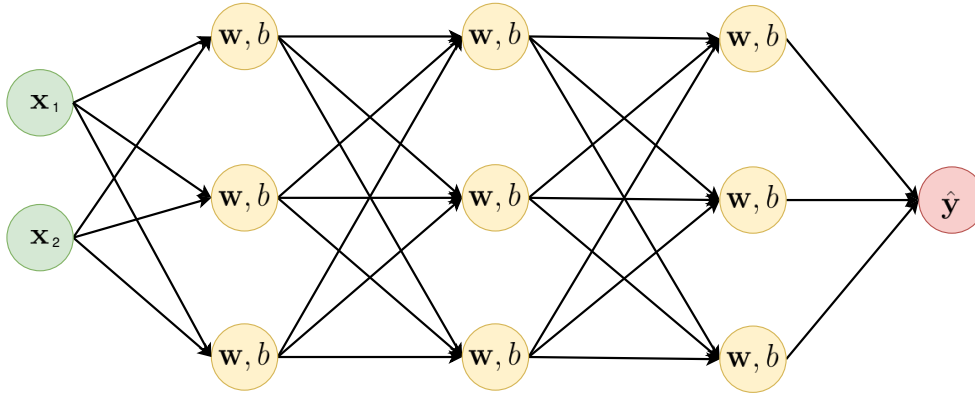
$$SS_{\text{res}} = \sum_i (y_i^{\text{true}} - y_i^{\text{pred}})^2. \quad (22)$$

With these two quantities, we can now define the  $R^2$ -score as follows:

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}. \quad (23)$$

### 3.2 Deep Neural Networks and backpropagation

Probably the most successful machine learning models are deep neural networks, which enjoy a great popularity in the recent years because of their flexibility and expressive power [10]. The training of deep neural networks for various machine learning task became possible after the proposal of backpropagation algorithm in 1986 by Geoffrey Hinton et. al. [11].



## 4 Quantum Machine Learning

QSVM paper = [12] HHL paper = [13] Feature Hilbert Spaces = [14] Cv-Qnns -LLoyd = [15] [Dani: related papers: quantum svm, Kernel methods, HHL algorithm for matrix inversion, quantum-inspired classical algos

motivation: summary from Lloyd paper, regression, classification, hybrid traning, encoding, inference

General structure: encoding, rotations and beamsplitter = interferometer, squeezing, interferometer, displacement, all = affine transformation + at the end non-linear, e.g., cubic phase operation (numeric instability switch to Kerr gates -> write at the performance test)

Loss functions: regression, classification

Regularization: keep trace close to zero, non-Gaussian

]

## 4.1 Parametric quantum circuits

## 4.2 Variational Quantum Eigensolver

Variational Quantum Eigensolvers [16] are hybrid quantum-classical algorithms designed to find the lowest energy eigenvalue of some Hamiltonian  $H$ . VQEs are one of the most studied quantum algorithms because of their expected use cases in quantum chemistry and materials science [17, 18, 19, 20]. The basic idea behind the variational quantum eigensolver is as follows. First we fix the structure of a parametric quantum circuit called the ansatz, defined by a unitary  $U(\boldsymbol{\theta})$ , where  $\boldsymbol{\theta} \in \mathbb{R}^m$  are real parameters. Then we start by some initial state  $|\psi_0\rangle$ , which is usually  $|0\rangle^{\otimes M}$  in the qubit setting. For the continuous variable setting, the initial state can be the vacuum state, a displaced state, a squeezed state, or a photon number eigenstate. After preparing the circuit, we evaluate the expectation value of the Hamiltonian which is eventually the cost, or loss:

$$\mathcal{L}(\boldsymbol{\theta}) = \langle H \rangle = \langle \psi_0 | U^\dagger(\boldsymbol{\theta}) H U(\boldsymbol{\theta}) | \psi_0 \rangle \quad (24)$$

The goal is to minimize the loss  $\mathcal{L}$  by iteratively updating the parameters  $\boldsymbol{\theta}$  with some update rule. This update rule can be any gradient-free [21] or gradient-based method [Dani: idezet]. To use gradient descent optimization in the VQE setting, we need to evaluate the gradient  $\nabla_{\boldsymbol{\theta}} \mathcal{L}$ , which can be challenging, and is described in details in chapter 4.3. Algorithm 1 describes the variational quantum eigensolver with a general gradient-based learning method. Notice that we wrote  $\alpha_j^{(t)}$  for the learning rate, since in most modern optimizers like Adam [22], each parameter has its own learning rate and it may depend on the number of iterations already done.

---

**Algorithm 1** VQE

---

```

1: procedure VQE( $H, K, T, \alpha$ )
2:   for all  $t \in \{1, \dots, T\}$  do
3:     Prepare the ansatz  $U(\boldsymbol{\theta})$ 
4:     Estimate the loss  $\mathcal{L} = \langle \psi_0 | U^\dagger(\boldsymbol{\theta}) H U(\boldsymbol{\theta}) | \psi_0 \rangle$  by averaging  $K$  shots
5:     for all  $\theta_j \in \boldsymbol{\theta}$  do
6:       Calculate the quantum gradient  $\partial_{\theta_j} \mathcal{L}$ 
7:       Update the parameter  $\theta_j \leftarrow \theta_j - \alpha_j^{(t)} \partial_{\theta_j} \mathcal{L}$ 
8:     end for
9:   end for
10: end procedure

```

---

## 4.3 Calculating the gradients of the parameters

In this section, we introduce the theory of calculating quantum gradients in the continuous variable setting. We follow the notation of Schuld et. al. [23]. A variational quantum circuit is defined by a unitary  $U(\boldsymbol{\theta})$ ,  $\boldsymbol{\theta} \in \mathbb{R}^m$ , an initial state  $|\psi_0\rangle$  and a measurement operator  $B$ . We can understand a variational circuit as  $f : \mathbb{R}^m \rightarrow \mathbb{R}$ , [Dani: The Schuld paper writes  $f : \mathbb{R}^m \rightarrow \mathbb{R}^n$ , maybe a typo???] which

maps the circuit parameters to the expectation value of operator  $B$ :

$$f(\boldsymbol{\theta}) = \langle \psi_0 | U^\dagger(\boldsymbol{\theta}) B U(\boldsymbol{\theta}) | \psi_0 \rangle \quad (25)$$

In gradient-based optimizations, we need the gradient  $\nabla_{\boldsymbol{\theta}} f$ , and this eventually means calculating the partial derivatives  $\partial_{\mu} f$ ,  $\mu \in \boldsymbol{\theta}$ . In the continuous variable setting, observables are usually polynomials of the quadrature operators  $X_j$  and  $P_j$ .

## 5 Results

### 5.1 Regression

$$\mathcal{L} = \frac{1}{|\mathcal{B}|} \sum_{(x,y) \in \mathcal{B}} (y - \langle \psi_0 | \hat{U}^\dagger(x; \boldsymbol{\theta}) \hat{X}_2 \hat{U}(x; \boldsymbol{\theta}) | \psi_0 \rangle)^2 + \frac{\lambda}{|\mathcal{B}|} \sum ||W^{\text{active}}||^2 \quad (26)$$

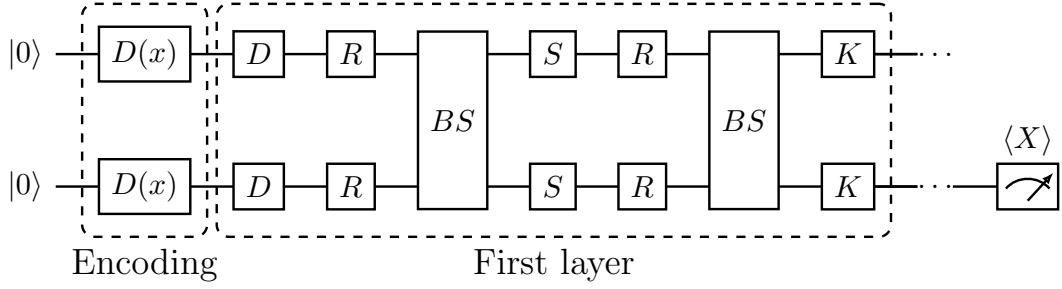


Figure 1.

### 5.2 Classification

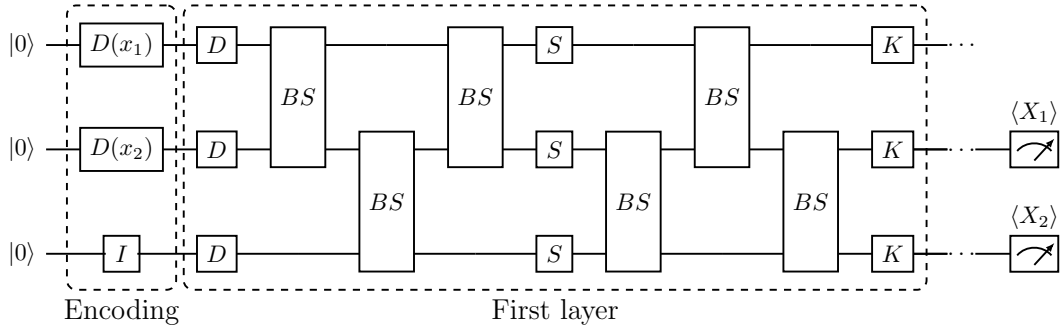


Figure 2.

## 5 Results

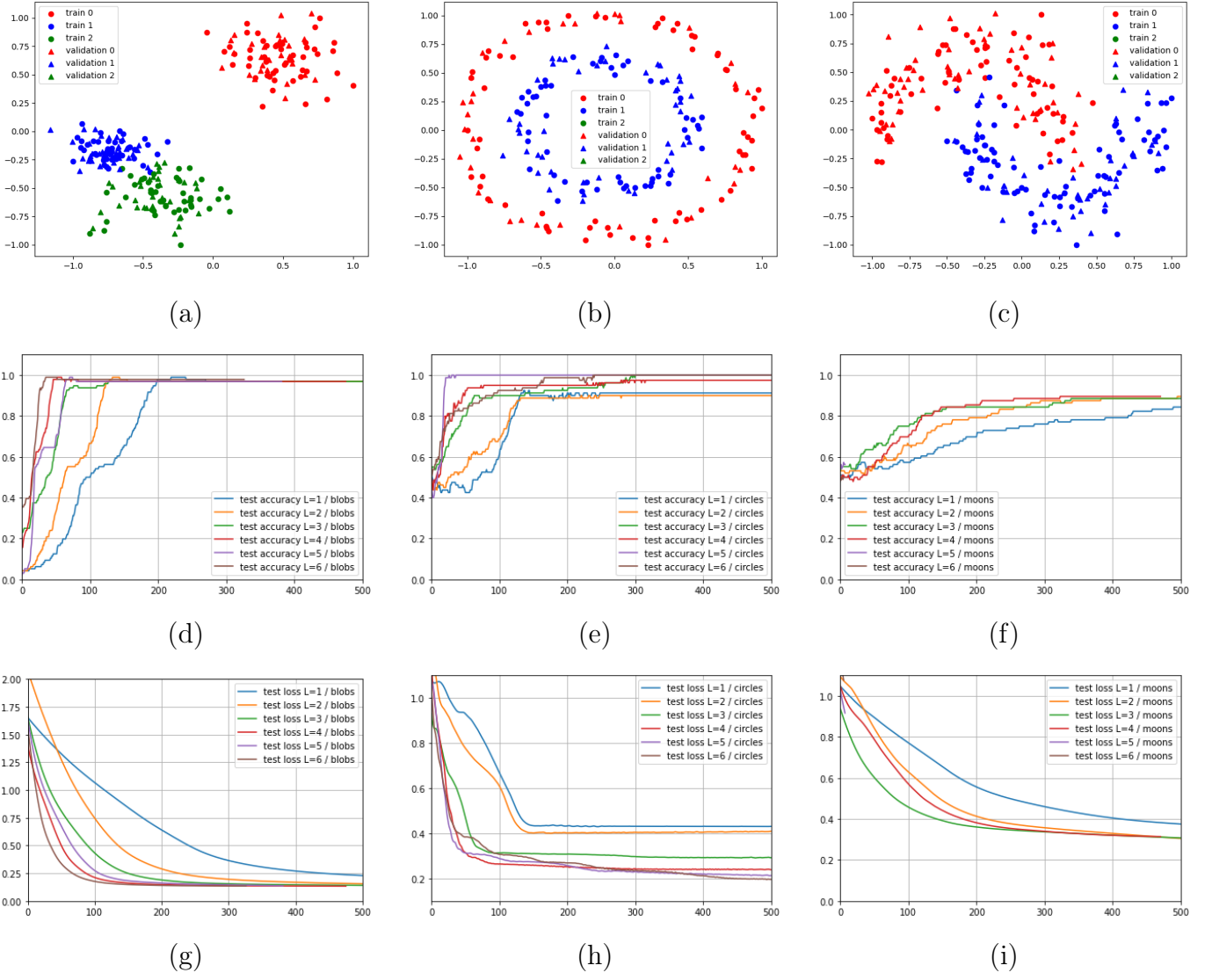


Figure 3. leiras



Figure 4.



### 5.3 CV-VQE for the Bose-Hubbard model

The Bose-Hubbard model introduced first by Gersch and Knollman in 1963 for the description of interacting spinless bosons on a lattice, such as ultracold bosonic atoms on an optical lattice [24, 25]. This model has also gained popularity because it is found to be an adequate model for the description of the Superfluidity-Mott insulator (SF-MI) transition, which has been experimentally demonstrated [26, 27]. We chose this model, because an exact diagonalization of the Bose-Hubbard Hamiltonian with respect to a finite dimensional Fock-basis is possible and thus the model is numerically solvable. This enables the verification of the correctness of the results given by our variational quantum solver for small problem instances.

The Bose Hubbard model which we used in our simulations is defined by the following Hamiltonian expressed with the bosonic creation and annihilation operators:

$$H = -t \sum_{i=1}^{M-1} (a_i^\dagger a_{i+1} + a_{i+1}^\dagger a_i) + \frac{U}{2} \sum_{i=1}^M n_i^2, \quad (27)$$

where  $M$  is the number of bosonic modes in the system. If we restrict the total number of bosons in the system to be  $N$ , the dimension of the restricted Fock-space is

$$D = \frac{(N + M - 1)!}{N!(M - 1)!}, \quad (28)$$

which explosively grows with the system size. Therefore finding the groundstate of the Bose-Hubbard Hamiltonian for larger systems can be intractable even on classical supercomputers, making quantum computers, especially continuous-variable quantum computers a potential candidate for solving these problems in the future. Compared to the original definition of the Bose-Hubbard Hamiltonian, we omitted the term  $-\mu \sum_i n_i$ , because it is not relevant in our particular experiments.

#### 5.3.1 Solving the model by exact diagonalization

**[Dani: Analytic solution by diagonalization]** If we want to calculate the groundstate of a Bose-Hubbard model, then we need to solve the eigenvalue-problem of Hamiltonian 27. Of course a general Fock-space corresponding to a bosonic system is infinite-dimensional, therefore we have to restrict the system so that a maximum of  $N$  particles are allowed in each mode:

$$|0, 0, \dots, 0\rangle, |0, 0, \dots, N\rangle, |0, 0, \dots, 1, N\rangle, \dots |N, N, \dots, N\rangle.$$

Then, we can relabel the Fock-states so that each Fock basis has its own unique label  $j$ :

$$|n_{M-1}, n_{M-2}, \dots, n_1, n_0\rangle = \left| \sum_{k=0}^{M-1} n_k (N+1)^k \right\rangle. \quad (29)$$

This is useful, because with these relabeled states we can calculate the matrix elements of the new Hamiltonian:

$$H_{jj'} = \langle j | H | j' \rangle = \langle n_{M-1}, n_{M-2}, \dots, n_1, n_0 | H | n'_{M-1}, n'_{M-2}, \dots, n'_1, n'_0 \rangle \quad (30)$$

We can further reduce the dimension of the Hilbert space if we prescribe that the total number of photons must be  $N$ :

$$\sum_k n_k = N.$$

In this case for example if the number of modes is  $M = 4$  and the total number of particles is  $N = 4$ , the dimension of the restricted Hilbert-space is only 35.

The resulting matrix  $H_{jj'}$  is usually a sparse matrix (see fig. 5), and can be diagonalized using existing software.

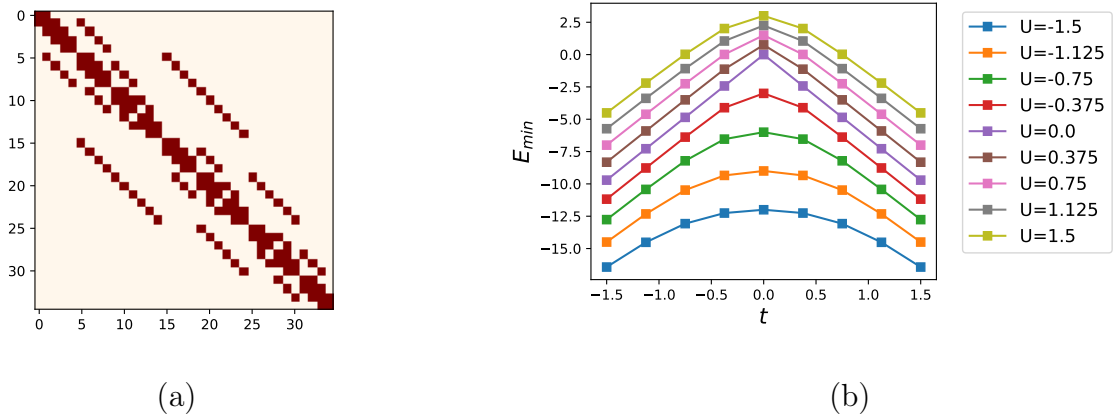


Figure 5. (a) Visual representation of a sparse Hamiltonian matrix  $H_{jj'}$  for a Bose-Hubbard model with  $\dim(\mathcal{H}) = 35$ . Darker points show the non-zero elements in the matrix (b) Numerical solutions of the ground-state energy for different  $U$  and  $t$  values.

### 5.3.2 Variational solutions

VQEs introduced in section 4.2 are shown to be effective for simulating different fermionic quantum systems, like molecules or metals [17, 18, 19, 20]. In this chapter we demonstrate a photonic VQE for finding the ground-state energy of a Bose-Hubbard model. Furthermore, we show that stochastic gradient descent methods can be applied to the VQE to reduce the number of shots required, and thus reduce the overall cost of finding the ground state energy on a quantum hardware.

The variational circuit  $U(\theta)$  can be realized with sequences of quantum gates, in our case photonic quantum gates. Since we want the total number of particles in the system to be preserved during the simulations, we must use photon-number preserving gates only. We composed each quantum layer from Beamsplitters, Kerr-gates and Cross-Kerr-gates (see fig. 7).

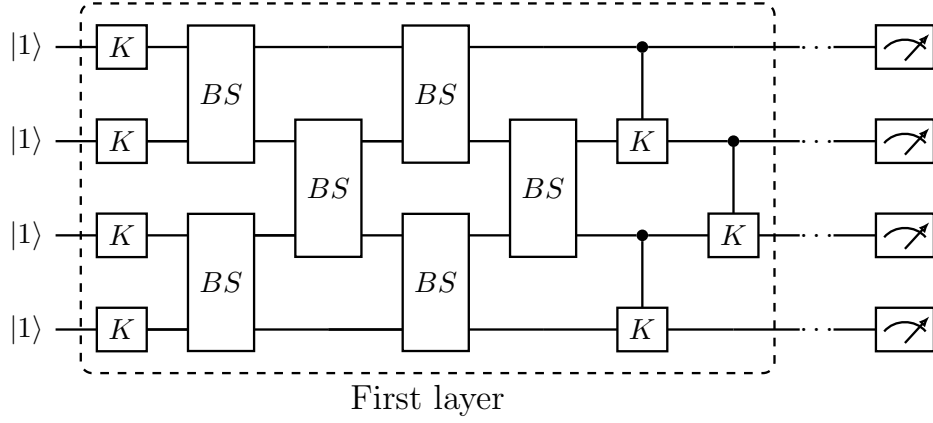


Figure 6. Illustration of a variational ansatz circuit with its first layer. Initially, each of the four qumodes contain a single photon. The layer is built up from Kerr-gates, CrossKerr gates and Beamsplitters, which are passive optical gates, so the total number of photons is preserved.

Throughout the numerical experiments, we set initially one single photon in each of the four qumodes i.e.  $|\psi_0\rangle = |1\rangle^{\otimes M}$ , and applied a number of identical quantum layers followed by quadrature measurements.

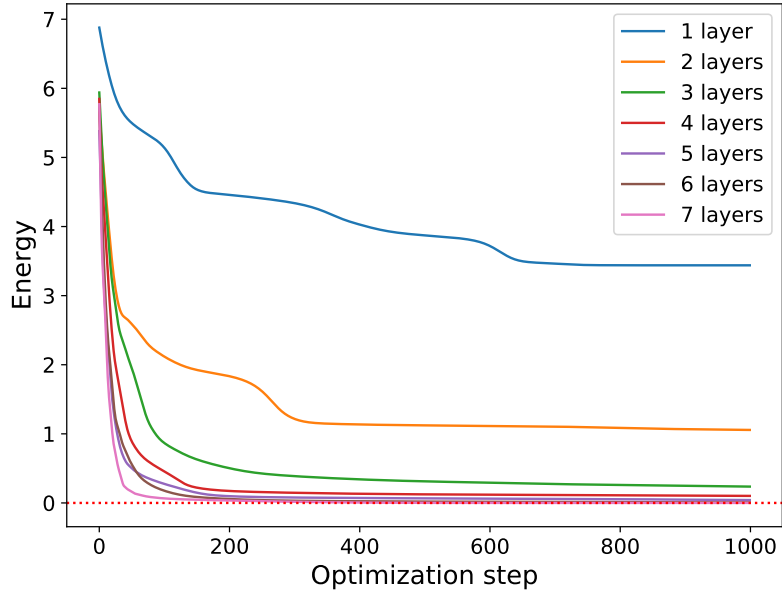
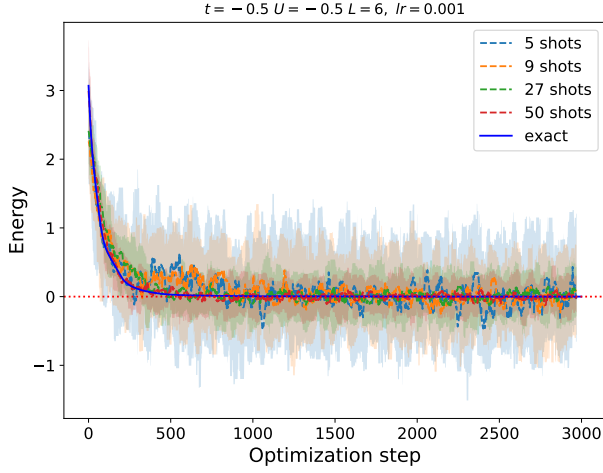
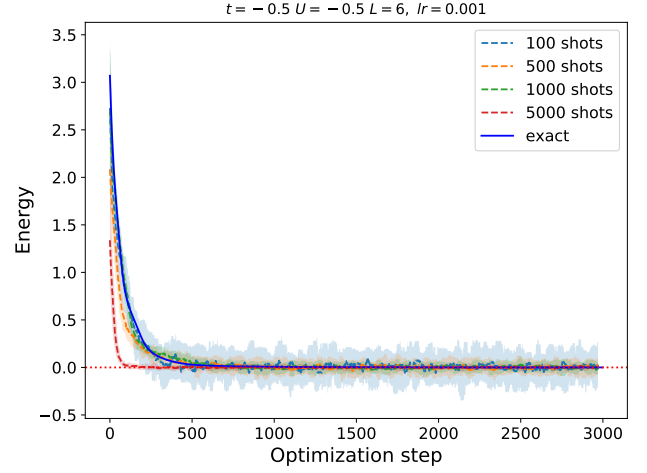


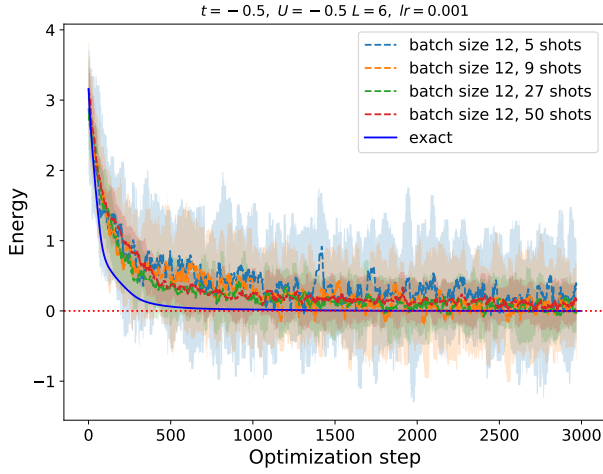
Figure 7.



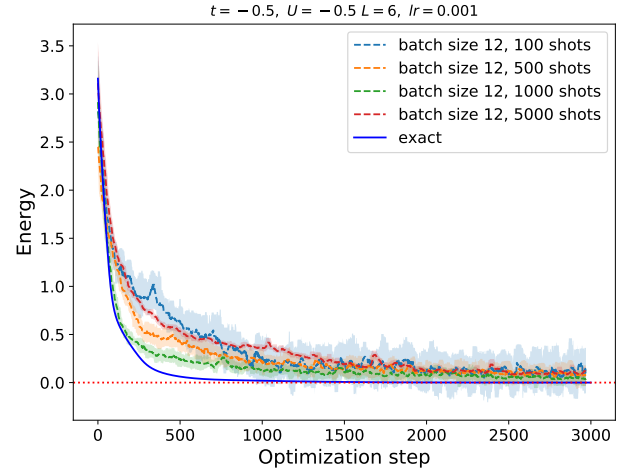
(a) first



(b) second



(c) third



(d) fourth

Figure 8. caption

## References

- [1] David Deutsch and Richard Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1907): 553–558, December 1992. doi: 10.1098/rspa.1992.0167. URL <https://doi.org/10.1098/rspa.1992.0167>.
- [2] P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*. IEEE Comput. Soc. Press. doi: 10.1109/sfcs.1994.365700. URL <https://doi.org/10.1109/sfcs.1994.365700>.
- [3] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC 1996*. ACM Press, 1996. doi: 10.1145/237814.237866. URL <https://doi.org/10.1145/237814.237866>.

- [4] Ewin Tang. A quantum-inspired classical algorithm for recommendation systems. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. ACM, June 2019. doi: 10.1145/3313276.3316310. URL <https://doi.org/10.1145/3313276.3316310>.
- [5] Chen Ding, Tian-Yi Bao, and He-Liang Huang. Quantum-Inspired Support Vector Machine. *arXiv e-prints*, art. arXiv:1906.08902, June 2019.
- [6] Juan Miguel Arrazola, Alain Delgado, Bhaskar Roy Bardhan, and Seth Lloyd. Quantum-inspired algorithms in practice. *arXiv e-prints*, art. arXiv:1905.10415, May 2019.
- [7] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv e-prints*, art. arXiv:1312.6114, December 2013.
- [8] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. *arXiv e-prints*, art. arXiv:1312.5602, December 2013.
- [9] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016. doi: 10.1038/nature16961. URL <https://doi.org/10.1038/nature16961>.
- [10] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015. doi: 10.1038/nature14539. URL <https://doi.org/10.1038/nature14539>.
- [11] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986. doi: 10.1038/323533a0. URL <https://doi.org/10.1038/323533a0>.
- [12] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Phys. Rev. Lett.*, 113:130503, Sep 2014. doi: 10.1103/PhysRevLett.113.130503. URL <https://link.aps.org/doi/10.1103/PhysRevLett.113.130503>.
- [13] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.*, 103:150502, Oct 2009. doi: 10.1103/PhysRevLett.103.150502. URL <https://link.aps.org/doi/10.1103/PhysRevLett.103.150502>.
- [14] Maria Schuld and Nathan Killoran. Quantum machine learning in feature hilbert spaces. *Phys. Rev. Lett.*, 122:040504, Feb 2019. doi: 10.1103/PhysRevLett.122.040504. URL <https://link.aps.org/doi/10.1103/PhysRevLett.122.040504>.
- [15] Nathan Killoran, Thomas R. Bromley, Juan Miguel Arrazola, Maria Schuld, Nicolás Quesada, and Seth Lloyd. Continuous-variable quantum neural networks. *Phys. Rev. Research*, 1:033063, Oct 2019. doi: 10.1103/PhysRevResearch.1.033063. URL <https://link.aps.org/doi/10.1103/PhysRevResearch.1.033063>.

- [16] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O’Brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5(1), July 2014. doi: 10.1038/ncomms5213. URL <https://doi.org/10.1038/ncomms5213>.
- [17] Shijie Wei, Hang Li, and GuiLu Long. A full quantum eigensolver for quantum chemistry simulations. *Research*, 2020:1–11, March 2020. doi: 10.34133/2020/1486935. URL <https://doi.org/10.34133/2020/1486935>.
- [18] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Sergio Boixo, Michael Broughton, Bob B. Buckley, David A. Buell, Brian Burkett, Nicholas Bushnell, Yu Chen, Zijun Chen, Benjamin Chiaro, Roberto Collins, William Courtney, Sean Demura, Andrew Dunsworth, Edward Farhi, Austin Fowler, Brooks Foxen, Craig Gidney, Marissa Giustina, Rob Graff, Steve Habegger, Matthew P. Harrigan, Alan Ho, Sabrina Hong, Trent Huang, William J. Huggins, Lev Ioffe, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Cody Jones, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Seon Kim, Paul V. Klimov, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Pavel Laptev, Mike Lindmark, Erik Lucero, Orion Martin, John M. Martinis, Jarrod R. McClean, Matt McEwen, Anthony Megrant, Xiao Mi, Masoud Mohseni, Wojciech Mruczkiewicz, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Hartmut Neven, Murphy Yuezhen Niu, Thomas E. O’Brien, Eric Ostby, Andre Petukhov, Harald Putterman, Chris Quintana, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Doug Strain, Kevin J. Sung, Marco Szalay, Tyler Y. Takeshita, Amit Vainsencher, Theodore White, Nathan Wiebe, Z. Jamie Yao, Ping Yeh, and Adam Zalcman. Hartree-fock on a superconducting qubit quantum computer. *Science*, 369(6507):1084–1089, 2020. ISSN 0036-8075. doi: 10.1126/science.abb9811. URL <https://science.sciencemag.org/content/369/6507/1084>.
- [19] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M. Chow, and Jay M. Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, 549(7671):242–246, September 2017. doi: 10.1038/nature23879. URL <https://doi.org/10.1038/nature23879>.
- [20] Cornelius Hempel, Christine Maier, Jonathan Romero, Jarrod McClean, Thomas Monz, Heng Shen, Petar Jurcevic, Ben P. Lanyon, Peter Love, Ryan Babbush, Alán Aspuru-Guzik, Rainer Blatt, and Christian F. Roos. Quantum chemistry calculations on a trapped-ion quantum simulator. *Phys. Rev. X*, 8:031022, Jul 2018. doi: 10.1103/PhysRevX.8.031022. URL <https://link.aps.org/doi/10.1103/PhysRevX.8.031022>.
- [21] D. Zhu, N. M. Linke, M. Benedetti, K. A. Landsman, N. H. Nguyen, C. H. Alderete, A. Perdomo-Ortiz, N. Korda, A. Garfoot, C. Brecque, L. Egan, O. Perdomo, and C. Monroe. Training of quantum circuits on a hybrid quantum computer. *Science Advances*, 5(10):eaaw9918, October 2019. doi: 10.1126/sciadv.aaw9918. URL <https://doi.org/10.1126/sciadv.aaw9918>.
- [22] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

- [23] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *Phys. Rev. A*, 99:032331, Mar 2019. doi: 10.1103/PhysRevA.99.032331. URL <https://link.aps.org/doi/10.1103/PhysRevA.99.032331>.
- [24] H. A. Gersch and G. C. Knollman. Quantum cell model for bosons. *Phys. Rev.*, 129:959–967, Jan 1963. doi: 10.1103/PhysRev.129.959. URL <https://link.aps.org/doi/10.1103/PhysRev.129.959>.
- [25] D. Jaksch and P. Zoller. The cold atom Hubbard toolbox. *Annals of Physics*, 315(1):52–79, January 2005. doi: 10.1016/j.aop.2004.09.010.
- [26] D. Jaksch, C. Bruder, J. I. Cirac, C. W. Gardiner, and P. Zoller. Cold bosonic atoms in optical lattices. *Phys. Rev. Lett.*, 81:3108–3111, Oct 1998. doi: 10.1103/PhysRevLett.81.3108. URL <https://link.aps.org/doi/10.1103/PhysRevLett.81.3108>.
- [27] Markus Greiner, Olaf Mandel, Tilman Esslinger, Theodor W. Hänsch, and Immanuel Bloch. Quantum phase transition from a superfluid to a mott insulator in a gas of ultracold atoms. *Nature*, 415(6867):39–44, January 2002. doi: 10.1038/415039a. URL <https://doi.org/10.1038/415039a>.

## A Coherent states

A coherent state is the eigenstate of  $\hat{a}$ :

$$\hat{a} |\alpha\rangle = \alpha |\alpha\rangle$$

$$|\alpha\rangle = \sum_{n=0}^{\infty} |n\rangle \langle n|\alpha\rangle$$

we have

$$\hat{a} |\alpha\rangle = \alpha |\alpha\rangle \implies \langle n|\hat{a}|\alpha\rangle = \alpha \langle n|\alpha\rangle$$

and

$$\begin{aligned} \hat{a}^\dagger |n\rangle &= \sqrt{n+1} |n+1\rangle \implies \langle n|\hat{a} = \sqrt{n+1} \langle n+1| \\ \implies \langle n|\hat{a}|\alpha\rangle &= \sqrt{n+1} \langle n+1|\alpha\rangle = \alpha \langle n|\alpha\rangle \\ \implies \langle n|\alpha\rangle &= \frac{\alpha}{\sqrt{n}} \langle n-1|\alpha\rangle = \dots = \frac{\alpha^n}{\sqrt{n!}} \langle 0|\alpha\rangle \end{aligned}$$

therefore,

$$|\alpha\rangle = \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} \langle 0|\alpha\rangle |n\rangle$$

, The value of  $\langle 0|\alpha\rangle$  can be calculated from the normalization condition  $\langle\alpha|\alpha\rangle \stackrel{!}{=} 1$ . Since

$$\begin{aligned} |\alpha\rangle &= \langle 0|\alpha\rangle \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} |n\rangle \\ \langle\alpha| &= \langle\alpha|0\rangle \sum_{k=0}^{\infty} \frac{(\alpha^*)^k}{\sqrt{k!}} \langle k|, \end{aligned}$$

we have

$$\begin{aligned} 1 \stackrel{!}{=} \langle\alpha|\alpha\rangle &= \left( \langle\alpha|0\rangle \sum_{k=0}^{\infty} \frac{(\alpha^*)^k}{\sqrt{k!}} \langle k| \right) \left( \langle 0|\alpha\rangle \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} |n\rangle \right) \\ &= \langle\alpha|0\rangle \langle 0|\alpha\rangle \sum_{k=0}^{\infty} \sum_{n=0}^{\infty} \frac{(\alpha^*)^k \alpha^n}{\sqrt{k!n!}} \langle k|n\rangle \\ &= \langle\alpha|0\rangle \langle 0|\alpha\rangle \sum_{k=0}^{\infty} \sum_{n=0}^{\infty} \frac{(\alpha^*)^k \alpha^n}{\sqrt{k!n!}} \delta_{kn} \\ &= |\langle 0|\alpha\rangle|^2 \sum_{n=0}^{\infty} \frac{(|\alpha|^2)^n}{n!} = |\langle 0|\alpha\rangle|^2 e^{|\alpha|^2}. \\ &\implies \langle 0|\alpha\rangle = e^{i\varphi} e^{-\frac{|\alpha|^2}{2}} \end{aligned}$$

And so, a coherent state can be written as

$$|\alpha\rangle = e^{i\varphi} e^{-\frac{|\alpha|^2}{2}} \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} |n\rangle$$

## B Mathematical preliminaries

### B.1 Hilbert spaces

#### Definition 1. *Hilbert-space*

Given a field  $T$  (real or complex), a vector space  $\mathcal{H}$  endowed with an inner product, is called a *Hilbert-space*, if it is a complete metric space with respect to the distance function induced by the inner product. The inner product is a map  $\langle \cdot | \cdot \rangle : \mathcal{H} \times \mathcal{H} \rightarrow T$ , for which  $\forall x, y, z \in \mathcal{H}$ :

- $\langle x|x \rangle \geq 0$
- $\langle x|x \rangle = 0 \iff x = \mathbf{0} \in \mathcal{H}$
- $\langle x|y \rangle = \langle y|x \rangle^*$ , where  $*$  denotes complex conjugation.
- $\langle x|\alpha y + \beta z \rangle = \alpha \langle x|y \rangle + \beta \langle x|z \rangle$ , where  $\alpha, \beta \in T$



The norm induced by this inner product is a map  $|| \cdot || : \mathcal{H} \rightarrow T$  defined as

$$||x|| = \sqrt{\langle x|x \rangle},$$

And the metric induced by this norm is defined as

$$d(x, y) = ||x - y|| = \sqrt{\langle x - y|x - y \rangle}.$$

The space  $\mathcal{H}$  is said to be complete if every Cauchy-sequence is convergent with respect to the norm, and the limit is in  $\mathcal{H}$ . That is, each sequence  $x_1, x_2, \dots$ , for which

$$\forall \varepsilon > 0 \exists N(\varepsilon) \text{ so, that } n > m > N(\varepsilon) \implies ||x_n - x_m|| < \varepsilon.$$

**Definition 2. Linear functional**

Let  $\mathcal{H}$  be a Hilbert-space over the field  $T$ . Then, the map  $\varphi : \mathcal{H} \rightarrow T$  is called a linear functional, if

$$\varphi(\alpha x + \beta y) = \alpha \varphi(x) + \beta \varphi(y), \quad \forall \alpha, \beta \in T, x, y \in \mathcal{H}.$$

**Definition 3. Dual space**

Given a Hilbert-space  $\mathcal{H}$ , its dual space,  $\mathcal{H}^*$  is the space of all continuous linear functionals from the space  $H$  into the base field. The norm of an element in  $\mathcal{H}^*$  is

$$||\varphi||_{\mathcal{H}^*} \stackrel{\text{def}}{=} \sup_{||x||=1, x \in \mathcal{H}} |\varphi(x)|.$$

**Theorem 1. Riesz representation theorem**

For every element  $y \in \mathcal{H}$ , there exists a unique element  $\varphi_y \in \mathcal{H}^*$ , defined by

$$\varphi_y(x) = \langle y|x \rangle, \quad \forall x \in \mathcal{H}.$$

The mapping  $y \mapsto \varphi_y$  is an antilinear mapping i.e.  $\alpha y_1 + \beta y_2 \mapsto \alpha^* \varphi_{y_1} + \beta^* \varphi_{y_2}$ , and the Riesz-representation theorem states that this mapping is an antilinear isomorphism. The inner product in  $\mathcal{H}^*$  satisfies

$$\langle \varphi_x | \varphi_y \rangle = \langle x|y \rangle^* = \langle y|x \rangle.$$

Moreover,  $||y||_{\mathcal{H}} = ||\varphi_y||_{\mathcal{H}^*}$ .

**Definition 4. Dirac-notation**

From now on, the elements in  $\mathcal{H}$  will be denoted by  $|x\rangle$  and their corresponding element in  $\mathcal{H}^*$  as  $\langle x|$ .

## B.2 Linear operators on Hilbert spaces

### Definition 5. *Linear operators*

A map  $\hat{A} : \mathcal{H}_1 \rightarrow \mathcal{H}_2$  is a linear operator, if

$$\hat{A}(\alpha |x\rangle + \beta |y\rangle) = \alpha(\hat{A}|x\rangle) + \beta(\hat{A}|y\rangle).$$

**Remark 1.** If not stated otherwise, we will assume that  $\mathcal{H}_1 = \mathcal{H}_2 = \mathcal{H}$ .

**Remark 2.** Operators will be denoted with a hat ( $\hat{\cdot}$ ).

### Definition 6. *Bounded linear operators*

A linear operator  $\hat{A} : \mathcal{H} \rightarrow \mathcal{H}$  is bounded, if

$$\exists m \in \mathbb{R} : |\langle v|\hat{A}|v\rangle| \leq m \langle v|v\rangle, \forall |v\rangle \in \mathcal{H}$$

**Remark 3.** The set of all bounded operators on  $\mathcal{H}$  is denoted  $\mathcal{B}(\mathcal{H})$ .

### Definition 7. *Commutators and anticommutators*

Since operators usually do not commute, its useful to define their commutator and anticommutator:

$$\begin{aligned} [\hat{A}, \hat{B}] &= \hat{A}\hat{B} - \hat{B}\hat{A} \\ \{\hat{A}, \hat{B}\} &= \hat{A}\hat{B} + \hat{B}\hat{A} \end{aligned}$$

### Definition 8. *Operator norm*

The operator norm of an operator  $\hat{A}$  is defined as

$$\|\hat{A}\| \stackrel{\text{def}}{=} \inf\{c \geq 0 : \|\hat{A}|v\rangle\| \leq c\| |v\rangle \|, \forall |v\rangle \in \mathcal{H}\}$$

### Definition 9. *Trace-class operators*

An operator  $\hat{A}$  is called trace-class if it admits a well defined and finite trace  $\text{Tr} [\hat{A}] = \sum_j \langle j|\hat{A}|j\rangle$

### Definition 10. *Positive operators*

An operator  $\hat{A}$  is called positive if  $\langle v|\hat{A}|v\rangle \geq 0, \forall |v\rangle \in \mathcal{H}$ . If  $\hat{A} = \sum_j \lambda_j |j\rangle \langle j|$  then  $\hat{A}$  is positive if  $\lambda_j \geq 0$ .

**Definition 11. *Projections*** An operator  $\Pi : \mathcal{H} \rightarrow \mathcal{H}$  is a projection if  $\Pi^2 = \Pi$ .

## B.3 Hermitian Operators, Unitary Operators, Spectral theorem, Hadamard-lemma

### Definition 12. *Hermitian adjoint*

Consider a **bounded** linear operator  $\hat{A} : \mathcal{H} \rightarrow \mathcal{H}$ . The hermitian adjoint of  $\hat{A}$  is a bounded linear

operator  $\hat{A}^\dagger : \mathcal{H} \rightarrow \mathcal{H}$  which satisfies

$$\langle y | \hat{A} | x \rangle = \left( \langle x | \hat{A}^\dagger | y \rangle \right)^*, \quad \forall |x\rangle, |y\rangle \in \mathcal{H}. \quad (31)$$

**Definition 13. Hermitian operators**

A bounded linear operator  $\hat{H} : \mathcal{H} \rightarrow \mathcal{H}$  is Hermitian if

$$\hat{H} = \hat{H}^\dagger, \text{ i.e. } \hat{H} |x\rangle = \hat{H}^\dagger |x\rangle, \quad \forall |x\rangle \in \mathcal{H}. \quad (32)$$

**Definition 14. Unitary operator**

A bounded linear operator  $\hat{U} : \mathcal{H} \rightarrow \mathcal{H}$  is unitary if

$$\hat{U}\hat{U}^\dagger = \hat{U}^\dagger\hat{U} = 1, \text{ in other words, } \hat{U}^\dagger = \hat{U}^{-1}. \quad (33)$$

**Definition 15. Eigenvalues and eigenvectors**

Consider bounded linear operator  $\hat{A}$ . If exist a vectors  $|k\rangle \in \mathcal{H}$  such that

$$\hat{A} |k\rangle = \lambda_k |k\rangle, \quad (34)$$

then  $|k\rangle$  is called an eigenvector of  $\hat{A}$  and  $\lambda_k$  is the corresponding eigenvalue.

An important property of Hermitian operators is that they can be diagonalized with real eigenvalues. This is formally stated by the spectral theorem:

**Theorem 2. The Spectral theorem**

Let  $\hat{A}$  be a bounded Hermitian operator on some Hilbert-space  $\mathcal{H}$ . Then there exists an orthonormal basis in  $\mathcal{H}$  which consists of the eigenvectors of  $\hat{A}$  and each eigenvalue of  $\hat{A}$  is real.

This means that any bounded Hermitian operator  $\hat{H}$  can be decomposed as

$$\hat{H} = \sum_k \lambda_k \hat{P}_k = \sum_k \lambda_k |k\rangle \langle k| \quad (35)$$

where  $\lambda_k$  and  $|k\rangle$  are the eigenvalues and eigenvectors of  $\hat{H}$ .

**Definition 16. Exponential of operators** If  $X$  is a linear operator, we can define the exponential of  $X$ :

$$e^X = \sum_{n=0}^{\infty} \frac{X^n}{n!}$$

**Important:** The product of exponentials of operators generally isn't equal to the exponential of their sum:

$$e^X e^Y = e^{Z(X,Y)} \neq e^{X+Y},$$

where  $Z(X, Y)$  is given by the Baker-Campbell-Hausdorff formula:

$$Z(X, Y) = X + Y + \frac{1}{2}[X, Y] + \frac{1}{12}[X, [X, Y]] - \frac{1}{12}[Y, [X, Y]] - \frac{1}{24}[Y, [X, [X, Y]]] \\ - \frac{1}{720}([[[[X, Y], Y], Y], Y] + [[[Y, X], X], X], X) + \dots$$

It is however equal if  $[X, Y] = 0$ :

$$\text{if } [X, Y] = 0 \implies e^X e^Y = e^{X+Y}$$

There are 2 important special cases:

**Theorem 3. *The Hadamard-lemma***

$$e^X Y e^{-X} = Y + [X, Y] + \frac{1}{2!}[X, [X, Y]] + \frac{1}{3!}[X, [X, [X, Y]]] + \dots$$

**Theorem 4.** *If  $X$  and  $Y$  commute with their commutator, i.e.  $[X, [X, Y]] = [Y, [X, Y]] = 0$ , then:*

$$e^X e^Y = e^{X+Y+\frac{1}{2}[X, Y]}$$

**Theorem 5.** *If  $[X, Y] = sY$  with  $s \in \mathbb{C}, s \neq 2i\pi n, n \in \mathbb{Z}$  then:*

$$e^X e^Y = \exp\left(X + \frac{s}{1 - e^{-s}} Y\right)$$

## B.4 Pure and mixed quantum states

**Definition 17. *Quantum states***

*A quantum state of a quantum system is a mathematical entity that provides a probability distribution for the outcomes of each possible measurement on the system.*

**Definition 18. *Pure quantum states***

*Pure quantum states are quantum states that can be described by a vector  $|\psi\rangle$  of norm 1.*

If one multiplies a pure quantum state by a complex scalar  $e^{i\alpha}$ , then the new state is physically equivalent to the former, thus  $|\psi\rangle$  and  $e^{i\alpha}|\psi\rangle$  are the same pure state. The transformation  $|\psi\rangle \rightarrow e^{i\alpha}|\psi\rangle$  does not change the outcomes of measurements on the state, however the phase  $\alpha$  is important in quantum algorithms.

**Example.** *For example, the states  $\frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi}|1\rangle)$  and  $\frac{1}{\sqrt{2}}(|0\rangle + e^{i\frac{\pi}{2}}|1\rangle)$  are not the same quantum state, but in both states there is 50-50 percent probability of measuring  $|0\rangle$  and  $|1\rangle$ .*

**Definition 19. *Density Matrix***

*A quantum state  $\hat{\rho}$  is a trace-1, self-adjoint, positive semidefinite operator. The set of quantum states is*

$$\mathcal{S}(\mathcal{H}) = \{\hat{\rho} : \hat{\rho} \geq 0, \hat{\rho} = \hat{\rho}^\dagger, \text{Tr}[\hat{\rho}] = 1\}$$

*A quantum state is pure if and only if  $\hat{\rho}^2 = \hat{\rho}$ . Also, if  $\rho$  is a pure state, then it can be written as  $\hat{\rho} = |\psi\rangle\langle\psi|$ . The operator  $\rho$  is called the density operator or density matrix.*