Automated design of photonic quantum circuits

Yuan Yao

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2023IPPAT005

Thèse de doctorat

Automated design of photonic quantum circuits

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom Paris

n° 626 : École Doctorale de l'Institut Polytechnique de Paris (ED IP Paris)
Spécialité de doctorat : Réseaux, informations et communications

Thèse présentée et soutenue à Palaiseau, le 23/02/2023, par

YUAN YAO

Composition du Jury :

Valentina Parigi Maître de conférence, LKB (Multimode Quantum Optics team)	Rapporteur
John Jeffers Professor, University of Strathclyde (Department of Physics)	Rapporteur
Frédéric Grosshans Chercheur CNRS, Sorbonne Université (LIP6)	Examinateur
Giuseppe Di Molfetta Professeur, Aix-Marseille University (LIS)	Président / Examinateur
Isabelle ZAQUINE Professeur, Télécom Paris (LTCI)	Directeur de thèse
Filippo MIATTO Chercheur, Xanadu (Architecture group) Maître de conférence invité, Télécom Paris (LTCI)	Co-directeur de thèse
Olivier Fercoq Professeur, Télécom Paris (LTCI)	Invité
Nicolás Quesada Assistant Professor, Polytechnique de Montréal (Department of Engineering Physics)	Invité

Abstract

Quantum computing is based on quantum physics phenomena, such as superposition and entanglement, and it promises to revolutionize the world of computing. Photonics is a prominent platform for realizing fault-tolerant quantum computing. It has various qualities: working at room temperature, large-scale manufacturability using existing foundries for silicon chips, and compatibility with optical communication to interconnect different quantum computers.

Our main goal is to automate the design of photonic quantum circuits and of their interconnects. Before a real photonic quantum computer can be manufactured, it is essential to numerically simulate and optimize the corresponding circuits, which in practice are built out of Gaussian components such as squeezers, beamsplitters, phase shifters, and homodyne detectors. To achieve universality, we also need non-Gaussian effects, which can be supplied by photon-number-resolving detectors. We design circuits from this toolbox and optimize them for various applications using various gradient descent algorithms, some of which we adapted to our purpose.

The main contributions are:

1. In photonics, Fock space and phase space representations are both useful formalisms to describe quantum states and transformations. We introduce a unified Fock space representation of all Gaussian objects in terms of a single linear recurrence relation that can recursively generate their Fock space amplitudes.

2. We find the composition rule of Gaussian unitaries in Fock space, which allows us to obtain the correct global phase when composing Gaussian unitaries (normally absent from the phase space description), and therefore to extend our model to states that can be written as linear combinations of Gaussians.

3. We first propose two methods to calculate the gradient of a Gaussian object based on our recursive representation, which enables us to further adapt different gradient-based optimizations to the problem of circuit optimization. The differentiability of our recursive representation allows for a straightforward computation of the gradients. We implement a Euclidean optimizer (i.e., which doesn't take the geometry of parameter space into account) in order to optimize each parametrized component of a circuit. Then we study two ways to account for geometry: first, we apply Riemannian optimization, by combining all the Gaussian transformations into a global transformation and following a geodesic on the manifold of symplectic matrices to find the optimized transformation, at which point we can decompose it back into fundamental optical components. Second, we generalize a complex version of the natural gradient for optical quantum circuits to accelerate the convergence of the training process.

4. We also give some optimal task-based strategies for using our recurrence relations. New algorithms are proposed to calculate, for instance, the amplitudes of a mixed state and the transformation matrix of interferometers. In addition, we derive a Gaussian evolution algorithm, which allows us to "fuse" the computation of the amplitudes of a Gaussian transformation and its action on any state.

5. We achieve the automated design of photonic quantum circuits by implementing the tools to simulate and optimize the circuits built from our recurrence relation. We give state preparation as the first example; we find circuits that can produce high-fidelity states in a reasonable time, such as cat states

with mean photon number 4, fidelity 99.38%, and success probability 7.3%. We can also optimize a 216-mode interferometer to make a Gaussian Boson Sampling experiment harder to spoof.

6. We made this work available in various open-source libraries: TheWalrus, StrawberryFields, Poenta, and MrMustard.

Introduction en français

L'informatique quantique est basée sur des phénomènes de physique quantique, tels que superposition et intrication, et il promet de révolutionner le monde de l'informatique. La photonique est une plateforme de premier plan pour réaliser l'informatique quantique tolérante aux erreurs. Il possède diverses qualités : fonctionnement à température ambiante, fabricabilité à grande échelle à l'aide des fonderies existantes pour les puces de silicium et compatibilité avec la communication optique pour interconnecter différents ordinateurs quantiques.

Notre objectif principal est d'automatiser la conception de circuits quantiques photoniques et de leurs interconnexions. Avant de pouvoir fabriquer un vrai ordinateur quantique photonique, il est essentiel de simuler numériquement et d'optimiser les circuits correspondants, qui sont en pratique construits à partir de composants gaussiens tels que des squeezers, des séparateurs de faisceaux, des déphaseurs et des détecteurs homodynes. Pour atteindre l'universalité, nous avons également besoin d'effets non gaussiens, qui peuvent être fournis par des détecteurs résolvant le nombre de photons. Nous concevons des circuits à partir de cette boîte à outils et les optimisons pour diverses applications en utilisant divers algorithmes de descente de gradient, dont certains que nous avons adaptés à notre objectif.

Les principaux apports sont :

1. En photonique, les représentations de l'espace de Fock et de l'espace des phases sont toutes deux des formalismes utiles pour décrire les états et les transformations quantiques. Nous introduisons une représentation unifiée dans l'espace de Fock de tous les objets gaussiens en termes d'une seule relation de récurrence linéaire qui peut générer de manière récursive leurs amplitudes dans l'espace de Fock.

2. On retrouve la règle de composition des unitaires gaussiens dans l'espace de Fock, qui permet d'obtenir la bonne phase globale lors de la composition des unitaires gaussiens (normalement absente de la description de l'espace des phases), et donc d'étendre notre modèle à des états qui s'écrivent comme combinaisons linéaires de gaussiennes.

3. Nous proposons d'abord deux méthodes pour calculer le gradient d'un objet gaussien à partir de notre représentation récursive, ce qui nous permet d'adapter davantage différentes optimisations basées sur le gradient au problème d'optimisation de circuit. La dérivabilité de notre représentation récursive permet un calcul simple des gradients. Nous implémentons un optimiseur euclidien (c'est-à-dire qui ne prend pas en compte la géométrie de l'espace des paramètres) afin d'optimiser chaque composant paramétré d'un circuit. Ensuite, nous étudions deux façons de rendre compte de la géométrie : premièrement, nous appliquons l'optimisation riemannienne, en combinant toutes les transformations gaussiennes en une transformation globale et en suivant une géodésique sur la variété des matrices symplectiques pour trouver la transformation optimisée, à quel point nous pouvons la décomposer dans les composants optiques fondamentaux. Deuxièmement, nous généralisons une version complexe du gradient naturel pour les circuits quantiques optiques afin d'accélérer la convergence du processus d'apprentissage.

4. Nous donnons également quelques stratégies optimales basées sur les tâches pour utiliser nos relations de récurrence. De nouveaux algorithmes sont proposés pour calculer, par exemple, les amplitudes d'un état mixte et la matrice de transformation des interféromètres. De plus, nous dérivons un algorithme d'évolution gaussienne, qui nous permet de "fusionner" le calcul des amplitudes d'une transformation gaussienne et son action sur n'importe quel état.

5. Nous réalisons la conception automatisée de circuits quantiques photoniques en implémentant les outils pour simuler et optimiser les circuits construits à partir de notre relation de récurrence. Nous donnons la préparation de l'état comme premier exemple ; nous trouvons des circuits capables de produire des états haute fidélité dans un temps raisonnable, tels que des états cat avec un nombre moyen de photons de 4, une fidélité de 99,38% et une probabilité de réussite de 7,3%. Nous pouvons également optimiser un interféromètre à 216 modes pour rendre une expérience d'échantillonnage de boson gaussien plus difficile à usurper.

6. Nous avons rendu ce travail disponible dans diverses bibliothèques open source : TheWalrus, StrawberryFields, Poenta et MrMustard.

Contents

List of Figures

List of Tables

List of notations

We will adopt the following notation conventions.

We use boldface for vectors $\boldsymbol{r}$ and matrices $\boldsymbol{S}$ but denote their components as r_i and S_{ij} respectively.

We especially use 0_M for the $M \times M$ null matrix, $\mathbf{0}$ for a zero vector, and 0 for a scalar zero. $\mathbb{1}_M$ denotes for the $M \times M$ identity matrix and I denotes for the identity operator.

In addition, if the operators in Fock representation are treated in the matrix representation, we use boldface for its tensor.

Table 1: List of mathematical notations

Object	notation
Hilbert space	$\mathscr{H}$
Hamiltonian	$\mathcal{H}$
reduced Plank constant	$\hbar$
Transportation	$.^T$
Hermitian conjugation for matrices	$.^\dagger$
Hermitian conjugation for expression	H.c.
Tensor product	$\otimes$
Direct plus	$\boldsymbol{A_1} \oplus \boldsymbol{A_2} = \begin{bmatrix} \boldsymbol{A_1} & 0 \\ 0 & \boldsymbol{A_2} \end{bmatrix}$

List of Abbreviation

Automatic Differentiation	AD
Adaptive Momentum estimation	ADAM
Canonical Commutation Relation	CCR
Complex Natural Gradient	CNG
Classical data Quantum algorithm type	CQ type
Continuous variable	CV
Gaussian Boson Sampling	GBS
Jacobian-Vector Product	JVP
Fubini Study metric	FS metric
MrMustard	MM
Natural Gradient	NG
Parameterized Quantum Circuit	PQC
Photon-Number-Resolving detector	PNR detector
Quantum Machine Learning	QML
Quantum data Quantum algorithm type	QQ type
Quantum Repeater	QR
Strawberryfields	SF
TensorFlow	TF
Vector-Jacobian Product	VJP
Variational Quantum Eigensolver	VQE

Chapter 1

Introduction

1.1 Background

Quantum mechanics is a fundamental theory in physics that provides a description of the physical properties of nature on an atomic scale [1]. Richard Feynman's talk in 1982 [2] proposed the idea of building a universal quantum simulator to describe quantum phenomena. Although it turned out to be the field of quantum simulation afterward, it didn't spark much interest at that time.

The interest in **Quantum computing** arose since the development of Shor's algorithm [3] in 1994, which could find out prime factors of large numbers more efficiently and is believed to be hard for classical computers and gives a shock to the security of the well-established cryptosystem RSA [4].

Quantum and classical computers are used to solve problems, but they fundamentally differ in manipulating data. Quantum computers involve quantum physics phenomena, such as superposition and entanglement. The term **quantum advantage** is introduced with the goal of demonstrating that a quantum computer can solve a problem that a classical computer can not solve in a reasonable time. It consists of two tasks:

- finding a problem that can be solved by a quantum computer and outperform the best-known algorithm for a classical one;
- building a quantum computer in practice.

Google first claimed to have achieved quantum supremacy [5] in 2019 with a quantum computer called Sycamore, which is based on superconducting materials and has 53-qubit. They chose the task of sampling the output of a pseudo-random quantum circuit to compare with the state-of-the-art computers.

In 2020, a team from the University of Science and Technology of China reached this milestone with their light-based quantum computer JiuZhang [6], which is able to finish a task in 200 seconds, while the classical computer would take more than half a billion years. The Gaussian boson sampling task was performed on it.

In late 2021, IBM's "Eagle" processor claimed to have achieved quantum supremacy in their research Blog, which broke the 100-qubit barrier and reached a 127-qubit quantum processor.

Xanadu won this race in 2022 [7] with their quantum computer Borealis, which carries the Gaussian boson sampling task with 216-mode as well [8, 9] in 36 μs, whereas it requires more than 9,000 years for the best available algorithms.

Though many demonstrations of quantum advantages exist, the way to a real usable quantum computer is still difficult and long. The first problem for all possible quantum computer candidates is decoherence which influences how many gates one can apply on the circuit successively. These factors can come from the external environment or the interactions between blocks inside a quantum computer. Fortunately, quantum error correction can help fight against this problem: the information can be protected by encoding it into more qubits and can be corrected after a noisy process. Except for looking for the resolution in the way of quantum error correction, researchers are now more or less working with noisy, intermediate-scale quantum

(NISQ) devices. Threshold fault-tolerant quantum computing [10, 11] is another answer to the errors in the quantum computer. Even though we don't have fault-tolerant quantum computing today, it is on its way.

Quantum information can be realized in different techniques: photon [12], superconductor [13], ultra-cold/Rydberg atom [14], electron spin [15] and ion traps [16]. This thesis focuses on **quantum optical computing**. Among all possible implementations for quantum computing, photonics is a prominent platform for several reasons. Photons are cheap, and building blocks are easy to make and add. Everything with a photon works at room temperature except for the detectors. Because of its characteristics, there are lots of options for encoding the information (polarization, time, frequency, photon number, quadrature, or particular states). Light can transmit information on the way of propagation. Moreover, optical fibers are everywhere and can be used directly for long-distance communication (compatibility). However, it has some challenges. Photons do not interact with each other therefore it is difficult to make the operators like squeezing effective. And the loss is also a big issue during transmission. So quantum error correction is also of importance in the scheme of quantum optical computing.

The biggest challenge now is to build a fault-tolerant quantum computer in which every component is lossy and noisy. And photonics quantum computer is a prominent platform for realizing fault-tolerant quantum computing.

Before a real photonic quantum computer can be manufactured, it is essential to numerically simulate and optimize the corresponding circuits, where "circuits" refer to the circuit model to represent the evolution of quantum states inside a quantum computer. Photonic quantum circuits, in practice, are built out of Gaussian components such as squeezers, beam-splitters, phase shifters, and homodyne detectors. There needs to be more, as to achieve universality, we also need non-Gaussian effects, which are supplied by Photon Number Resolving (PNR) detectors. In the numerical simulation, we design the circuits from the two components above and then optimize them for various applications. Machine learning techniques are used in our optimizations.

Machine learning techniques have become powerful tools in different areas of data processing [17]. The idea is to combine machine learning techniques with quantum systems appears, which is known as **quantum machine learning**. There are no definitions clearly now of what is quantum machine learning:

- from the point of view of data type and algorithm type, one can combine either classical data or quantum data with the quantum algorithm or classical algorithm. For example, analyzing classical data on a quantum computer is a CQ type (i.e., the quantum-enhanced machine learning [18]). Another example can be a QQ type, which is to classify the quantum data with a quantum algorithm (i.e., the quantum Grover search algorithm applied in the unstructured search for data processing [19]).

- one can also make the quantum computer as the quantum subroutines to run as part of classical machine learning techniques, such as the famous variational quantum eigensolver (VQE) algorithm to search the ground state of the electronic Hamiltonian of molecules [20]; formalizing problems of interest as variational optimization problems, such as the quantum approximate optimization algorithm (QAOA) [21] to find approximate solutions for MaxCut. This kind of hybrid quantum-classical algorithm would be considered quantum machine learning sometimes.

- optimizing parameterized quantum circuits via Machine Learning techniques [22] is also a famous direction. This approach is under the QQ type because both the learning device and the system are quantum. Parameterized quantum circuits could be referred to as the quantum neural network if one arranges the circuit in a layered structure and have the linear and activation part.

- other important quantum machine learning-related ideas: quantum reinforcement learning (combining reinforcement learning with quantum properties [23], quantum annealing (solving large-scale combinatorial optimization problems [24]), and hidden quantum Markov models (a quantum generalization of classical Markov chains [25]).

This thesis uses the third idea above to optimize photonic parameterized quantum circuits via machine learning techniques. Optimization is an aspect of Machine Learning, which is the process of training the

parameters in the model iteratively in order to reach a maximum or minimum function evaluation. We use one of the most important optimization algorithms: gradient descent.

In summary, this thesis focuses on the classical simulation and optimization of photonic parametrized quantum circuits via machine learning techniques, with the objective of automatic circuit design.

1.2 Context on detailed questions

In this section, we elaborate on the specific questions addressed in this thesis, which are all geared toward solving the following challenging task: *how can we design photonic quantum circuits automatically?* We need to separate this task into several steps to answer this question.

The photonic quantum circuit is composed of fundamental optical components (such as phase shifters and beam-splitters), non-linear optical components (such as Kerr gate and cubic gate), and detectors (such as homodyne detectors and PNR detectors). "How can we work with Gaussian and non-Gaussian objects (such as PNR detectors) in the same circuit" becomes our first question. To describe these fundamental optical components in mathematical forms and then simulate them on a classical computer, one can use phase space and Fock space representations, such as the coherent state that can be described both in phase space and Fock space representation. Another example is the PNR detectors, which can be easily described in Fock space representation but not in phase space. In this thesis, we develop the theory of *Gaussian quantum mechanics* to generate the Fock representation of Gaussian objects and work with non-linear optical components and detectors together in Fock space.

Our Gaussian quantum mechanics theory gives a mathematical tool to describe the evolution inside the photonic quantum circuits: the *recurrence relation*. Based on this, we build the quantum circuit with fundamental optical components, non-linear ones, and detectors and simulate the evolution of quantum states through it. The circuit component is parametrized, and the output changes with different parameter values. We can then train this parametrized circuit by calculating the gradient with respect to each parameter and updating it. So, getting the gradient with respect to the parameters is the first step in training the circuit. "How can we use the gradient to train a photonic quantum circuit" becomes the following question.

During the simulation, the Fock representation of a multi-mode Gaussian tensor occupies enormous computational resources, such as a 3-mode Gaussian operator with a cutoff of 50 uses more than one billion storing units, not to mention calculating the multiplication it with an input state. With a faster learning speed, the simulation ability can increase a lot and it allows us to design on larger circuits. We then introduce the idea "How can we improve the convergence rate".

This work can be used to build a playground for quantum optical communities. Unlike doing experiments in the laboratory directly, one can simply build the circuit, get all the optimized parameters that one wants and start the experiment. We implemented all the methods in this thesis in the freely available open-source library (which is a software library with an open-source license that is free to reuse, modify, and/or publish without permission) for sharing our work.

Now we are going to elaborate on all details of each question:

- How can we work with Gaussian and non-Gaussian objects in the same circuit at the same time?
- How can we use gradient descent to train a photonic quantum circuit?
- How can we improve the convergence rate of gradient descent?
- What is the best way of making this work publicly available?

1.2.1 Gaussian quantum mechanics

Gaussian quantum mechanics is a subset of quantum mechanics that finds applications in several fields of quantum physics, such as quantum optics [26], quantum key distribution [27], optomechanical systems [28], quantum chemistry [20], condensed matter systems [29]. In the context of quantum optics, many of the available states (e.g. coherent, squeezed, thermal), transformations (e.g., beam-splitter, squeezer,

displacement, attenuator, amplifier), and measurements (e.g., homodyne, heterodyne) are Gaussian, which are all characterized by a Gaussian phase space representation.

Gaussian objects are easy to manipulate inside the phase space representation. But it is essential to access a universal set of states and transformations, that is, to include non-Gaussian effects as well, which can be done by the photon number resolver. So the calculation is needed to transform from the Gaussian phase space representation to the Fock space representation, where we can describe photon number measurements. Hence, studying the Fock representation of Gaussian objects plays an important role in optical quantum simulation and optical quantum information processing.

The Fock representation of Gaussian objects has been studied in different communities: in chemical physics, one studies vibronic transitions using the Hermite polynomials as a computational tool [30, 31, 32, 33], and the matrix elements of Gaussian and non-Gaussian transformations have been evaluated in [34] by using the multimode Bogoliubov transformation.

Our recurrence relation work in chapter 6 paves the way to work with Gaussian quantum mechanics in Fock space. [35] introduced a method to compute the Fock space amplitudes of Gaussian unitary transformations using a generating function. Part of the thesis work is to extend that method to cover Gaussian pure states, mixed states, and all Gaussian channels as well.

1.2.2 Gradient descent to train the optical quantum circuit

The most used algorithm to train the optimization circuits is gradient descent, which was first proposed in 1847 by Augustin-Louis Cauchy [36]. The gradient of a function is the calculation in a numeric way to point out where to adjust the parameters. Stepping repeatedly by applying the gradient will lead to the minimum of that function. It has the disadvantage of converging into the local minimum depending on the initial points. Also, the choice of the learning rate is challenging if one doesn't know well the shape of the parameter space; it could be too low or too high.

Usually, the gradient is to compute the partial derivatives of each parameter. So that it requires the cost function to be differentiable; however, it is not an easy task in quantum optical circuits.

If we want to simulate the quantum optical circuit, we can consider the state as the vector, and the operators are matrices. The evolution of the state in the circuit is matrix-vector multiplication. The difficult part is to get the matrices for the operators. Each optical component is expressed as a matrix exponential of a linear combination of infinite-dimensional non-commuting operators:

$$e^{\frac{1}{2}\hat{\boldsymbol{z}}^T M \hat{\boldsymbol{z}} + \hat{\boldsymbol{z}}^T \hat{\boldsymbol{z}}}, \tag{1.1}$$

where $\boldsymbol{z}$ is defined in Eq. (3.2).

One cannot keep the infinite-dimensional matrices when one tries to simulate this kind of infinite-dimensional operator in matrix form. Therefore, truncation them with some cutoffs is a good choice, but one cannot simply truncate the matrix at the exponent and compute the matrix exponential [35]:

$$\text{trunc}(\exp(A)) \neq \exp(\text{trunc}(A)). \tag{1.2}$$

The previous work in [35] proposed generating the matrix elements of general Gaussian transformations recursively up to the desired cutoff dimension. Our recurrence relation work generalizes this work and extends the recurrence relation to all Gaussian objects.

Because the equation is recursive and each element can be obtained from the linear combination of other elements, we can differentiate it directly in order to get the gradient. We then show the differentiability of our recurrence relation in section 7.2.3 and give the exact equation to calculate the gradients with respect to each parameter.

1.2.3 Improvements in the speed of the optimization

Once the gradient for each element in the circuit is obtained, one can optimize the circuit by applying the gradient at each step. We want to make the optimization faster so that we are able to train bigger circuits.

We first develop the Euclidean method to optimize each parameter in specific circuits. Then we propose to treat all Gaussian components into a global transformation whose symplectic matrix is endowed in the Riemannian manifold. This is where the Riemannian optimization method comes from. After optimizing the global transformation, we decompose it back to each component. Both the Euclidean and Riemannian optimization methods are explained in section 7.3.

On the other side, our parameter space is complicated if we have many parameters in our circuit; it is where we start to consider the natural gradient algorithm. This algorithm involves the underlying geometry of the parameter space in the calculation of the gradient. The quantum version has been proposed in [37], and we provide the complex natural gradient in section 7.4. Moreover, a systematic geometric framework is presented to study the closed quantum systems from the point of view of the geometry of variational quantum circuits [38].

Except for the improvements in the circuit, we can also find various algorithms to gain calculation speed with various tasks. These strategies are introduced in chapter 8.

Complex natural gradient

The natural gradient (NG) has long been established as an approach to learning tasks that outperforms vanilla gradient descent at the expense of needing a matrix inversion of an $n \times n$ matrix at each step, where n is the number of parameters being updated during that step [39, 40, 41]. Such a matrix represents the metric tensor of the parameter space as seen by the cost function, and we can use it to find the steepest descent direction while taking into account the local geometry [40, 41, 42]. The NG approach has also been proposed as a component of variational quantum algorithms [43].

In the context of parametrized quantum systems, the metric tensor for pure states is the Fubini-Study metric tensor [37], which can then be adapted for density matrices [44]. The quantum NG has proven successful in boosting hybrid quantum-classical algorithms such as VQE for qubit systems [37, 44, 45] outperforming other optimization methods [42, 45]. In fact, the quantum NG yields stable results for various system sizes, and it achieves convergence in fewer epochs than Euclidean gradient approaches [37, 44, 45]. In the VQE scenario, the metric tensor may be evaluated with the gradient directly on quantum hardware [44, 46, 47, 48] and then inverted classically.

The quantum NG also looks promising in the noisy intermediate-scale quantum [49] computing paradigm, as it helps with noisy measurement situations [44]. Furthermore, some block-diagonal approximations have been introduced for layered quantum circuits [37], which can reduce the computational load without excessively compromising the advantages. Note, however, that the metric tensor used for quantum states can be noninvertible or ill-conditioned [42, 50, 45]. This corresponds to cases where one or more parameters become redundant, which means that a single quantum state corresponds to multiple parameter values. To avoid this, one can introduce regularization [37] and use the Penrose pseudo-inverse rather than the inverse of the metric [51].

Our work complex natural gradient (published in [52]) extends NG from the real-valued parameters only to the complex-valued parameters so that the inversion matrix has a shape $n \times n$ instead of $2n \times 2n$ and one can skip the step to separate the real and imaginary part and combine them. Moreover, our complex version's metric tensor is holomorphic, allowing us to use it in the AD model directly. More details can be found in section 7.4.

Riemannian optimization

Differential-geometrical learning algorithms are well used in the machine learning community [53]. Usually, the geometry of the parameter space is Euclidean, while some others are Riemannian. Riemannian manifold in differential geometry is a real and smooth manifold equipped with a Riemannian metric on the tangent space at each point.

If one obtains the Euclidean gradient $\frac{\partial \boldsymbol{M}_i}{\partial \theta}$ of matrix $\boldsymbol{M}_i$ and applies it:

$$\boldsymbol{M}_{i+1} \leftarrow \boldsymbol{M}_i - \eta \frac{\partial \boldsymbol{M}_i}{\partial \theta}. \tag{1.3}$$

The matrix $\boldsymbol{M}_{i+1}$ would step out of its manifold.

The intuitive approach of Riemannian manifold optimization can be geodesic calculation (geodesic on the Riemannian manifold is the shortest curve connection on the manifold between two points [54]), globally convergent steepest descent [55], Newton [56], quasi-Newton [57] and the conjugate gradient methods [58].

In our work, we will optimize the real symplectic matrices by calculating the geodesic, which forms a Lie group endowed with a Riemannian metric. The gradient flow needs to be considered because the standard Euclidean gradient no longer works with the Riemannian manifold. The Riemannian gradient is defined following the tangency condition and compatibility condition, which guarantees the gradient vector is tangent to the manifold and the inner product is invariant.

This way, we can get the Riemannian gradient of the symplectic matrix and calculate the geodesic on it to update the matrix [54, 59]. This is called the Riemannian optimization in section 7.3.2.

1.2.4 Best way to make our work useful

Several libraries are developed along with the thesis. For example, with the Poenta [60] library, one can construct a layered-structure photonic quantum circuit and train it with a given cost function. MrMustard [61] represents our bridge work between phase space and Fock space. I have also partly contributed to some functionalities of TheWalrus [62], and StrawberryFields [63]. Different applications using our libraries are shown in chapter 9, and we also give their code snippets in Appendix D.

There are also many other software libraries to simulate quantum optical circuits, such as QuTiP [64, 65] and Perceval [66, 67]. However, they have different focuses and abilities in various quantum information processing aspects.

- **Strawberryfields** (SF) [68] by Xanadu focuses on constructing and simulation continuous-variable photonic quantum computing. It has several backends for the quantum circuit simulation: Gaussian, Fock, and Bosonic numeric ones. It can, especially, contact with Xanadu's quantum hardware. A gallery of applications on near-term hardware is provided with examples. Some important functions in SF are based on another library TheWalrus [62], which calculates hafnians, Hermite polynomials, and Gaussian boson sampling.

- **MrMustard** (MM) [61] by Xanadu represents our work to bridge between phase space and Fock space with rich functionality in both representations. It can work on both the TensorFlow and PyTorch environments.

- **QuTip** [69] by Paul Nation, and Robert Johansson is a quantum Toolbox in python, focusing on calculating and simulating quantum circuits. Except for normal quantum objects like state vectors, density matrices, and quantum operators, it can also work with super-operators (for master equations). Solving the time-evolution is also its task according to the Schrödinger equation, von-Neuman equation, master equations, Floquet formalism, Monte-Carlo quantum trajectors, and experimental implementations of the stochastic Schrodinger/master equations, as they claim on their website.

- **Perceval** [70] by Quandela focuses on quantum optical simulation based on linear optics components, defining single-photon sources, manipulating Fock states, running simulations, reproducing published experimental papers, and experimenting with a new generation of quantum algorithms.

1.3 Thesis structure

My thesis flows with the construction of photonic parameterized quantum circuits and ends with some applications. It is shown in Fig. 1.1.

The **recurrence relation** is at the core of this thesis, which gives the Fock representation tensors for all Gaussian objects. With its differentiability, we are able to compute the gradient of the parameterized quantum circuits together with the non-Gaussian effects. The two usual types of circuit structures are shown in both figures: the Gaussian and non-Gaussian gates and the Gaussian gates with the measurements. Once

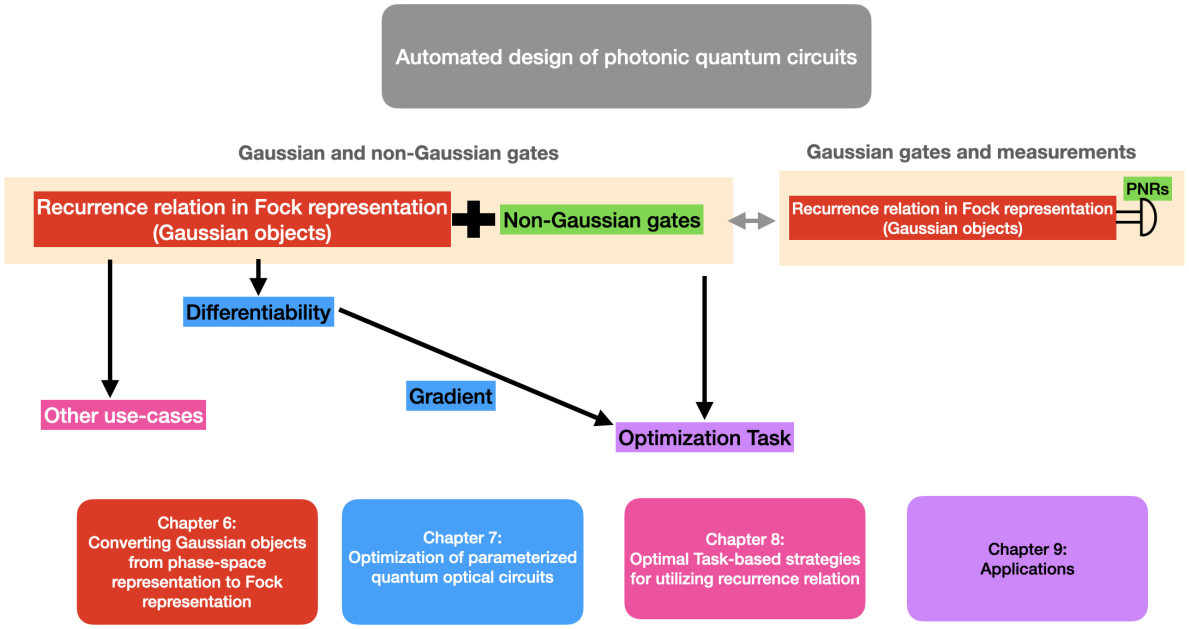


Figure 1.1: The structure of the thesis. We generally have two optimization structures: Gaussian gates and non-Gaussian gates, Gaussian gates and measurements (photon-number-resolving detectors). The discussion about building the recurrence relation of Gaussian objects in Fock representation is shown in chapter 6 (red rectangles). Because of the differentiability of the recurrence relation, we can calculate the gradient with respect to each parameter in the circuit. The optimization of parameterized quantum circuits is shown in chapter 7 (blue rectangles). Chapter 8 (pink rectangles) gives more optical task-based strategies for utilizing recurrence relations. Finally, we build the trainable circuits based on our recurrence relation, and we show that we can realize different optimization tasks in chapter 9 (purple rectangles).

the circuit is built, we can optimize the circuit with different tasks. Otherwise, some strategies for utilizing our recurrence relation are discussed for some given use cases. Last but not least, since our work is the classical simulation of photonic quantum circuits, we still offer insight into turning the simulated circuits into real ones in practice.

Converting Gaussian objects from phase-space representation to Fock representation: Chapter 6 is to find out the Fock representation of Gaussian objects (including Gaussian pure/mixed state, Gaussian unitary, and Gaussian channel) from its phase-space representation, characterized by its covariance matrix and mean vector.

1. The first idea is to show how to form the recurrence relation from the covariance matrix and mean vector for the Gaussian object to generate its corresponding Fock representation. We then show the equation for Gaussian pure/mixed state, Gaussian unitary, and Gaussian channel separately by using the previous method, whether Choi-Jamiolkowski isomorphism or Bargmann representation.

2. The second idea is to show how to get the global phase when composing two Gaussian unitaries to extend our recurrence method to objects beyond Gaussian.

In this chapter, we derive the bridge equation for Gaussian objects from phase-space representation to Fock representation, which is a recurrence relation. Then the explicit formula of the global phase between two Gaussian objects is proposed, which allows extending to bosonic qubits.

Optimization of parameterized quantum optical circuits: This chapter shows the differentiability of the recurrence relation and two ways to optimize Gaussian unitaries: Euclidean optimization and Riemannian optimization.

1. The first idea is to show how to calculate the gradient for each component of the circuit using automatic differentiation techniques. And we can get the explicit expression of the differentiation for any Gaussian object according to our recurrence relation. We propose the chain rule and the generating function methods.

2. The second idea is to show the two optimizations of Gaussian operators: Euclidean and Riemannian optimizations. Euclidean optimization is hardware-friendly because we decompose the Gaussian operator into fundamental optical components and optimize them directly. Riemannian optimization is theory-friendly because it optimizes the whole Gaussian operator using geodesic-based optimization, and we can decompose the operator afterward into optical components.

3. The last one is the complex natural gradient algorithm to reach the steepest step during optimization by considering the underlying geometry of the parameter space for optical components.

This chapter shows how to calculate the gradient from the recurrence relation, proposes two optimization methods in optical circuits, and generalizes the complex version of the natural gradient algorithm.

Optimal task-based strategies for utilizing recurrence relation: This chapter gives some optimal task-based strategies for using the recurrence relation we proposed in chapter 6 more cleverly.

Four algorithms are proposed:

1. The first idea is the global and local cutoff algorithm. By defining new cutoffs, we change the construction way of the final Gaussian tensor from the recurrence relation;

2. The second idea is the diagonal algorithm to obtain the Fock amplitudes of a mixed state.

3. The third idea is the interferometer algorithm to obtain the tensor of the interferometer in two different cases.

4. The last one is the Gaussian evolution algorithm to get the transformed state directly rather than matrix-vector multiplication.

In this chapter, we present four algorithms to accelerate the computation of tensors with the recurrence relation.

Applications: This chapter shows that the photonic PQCs can do many optimization tasks depending on various cost functions.

1. The first idea is to show the preparation task with different structures and gradient-based algorithms to generate single-photon, NOON state, and GKP state is rapid and with high fidelity.
2. The second idea is to show that we can do some other optimization tasks based on one component in the circuit.
3. The third idea is to find a realistic way for the optimized circuits. We propose the lossy model to make the numerical simulations more reasonable.

This chapter gives some applications of optimization tasks with photonic PQCs and finally discusses how to realize them in practice.

Part I

Fundamental concepts

Chapter 2

Brief introduction to quantum physics and quantum optics

This chapter will first briefly review the main ideas of quantum mechanics [71]. Then, we review the second quantization for a harmonic oscillator [72] and the Fock space representation in quantum optics.

2.1 Formalism of quantum mechanics

We consider the situation of an isolated quantum system: the state, the evolution, and the measurement. We use the *Dirac notation*, or *bra-ket notation*, to describe them. This notation is based on vector spaces and linear operations.

2.1.1 Linear algebra

The basic objects of linear algebra are *vector spaces*. The vector space V of interest is $\mathbb{C}^n$, which is spanned by n complex numbers $(z_1, \ldots, z_n)$. The elements of the vector space are vectors. Normally we write a vector $\boldsymbol{z}$ as the column vector:

$$\boldsymbol{z} = \begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix}. \tag{2.1}$$

The addition operation and the scalar multiplication are defined by

$$\boldsymbol{z} + \boldsymbol{z}' = \begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix} + \begin{bmatrix} z_1' \\ \vdots \\ z_n' \end{bmatrix} = \begin{bmatrix} z_1 + z_1' \\ \vdots \\ z_n + z_n' \end{bmatrix}, \tag{2.2}$$

$$c\boldsymbol{z} = c \begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix} = \begin{bmatrix} cz_1 \\ \vdots \\ cz_n \end{bmatrix}, \tag{2.3}$$

where $c \in \mathbb{C}$.

The vector under Dirac notation is written with the ket $|\psi\rangle \in V$. And the vector dual to a ket is called a *bra* $\langle\psi|$, which is mathematically the *hermitian* (transpose conjugate) of its ket vector as the row vector.

The *inner product* between two vectors $|\psi\rangle \in V$ and $|\varphi\rangle \in W$ is:

$$\langle\psi|\varphi\rangle, \tag{2.4}$$

which can be considered as the row vector multiplying the column vector. If $\langle\psi|\varphi\rangle = 0$, it means that these two vectors are *orthogonal* to each other. The physical meaning of orthogonal is that one can distinguish one state from another one with certainty.

The *tensor product* of two vectors $|\psi\rangle \in V$ and $|\varphi\rangle \in W$ is:

$$|\psi\rangle \otimes |\varphi\rangle, \tag{2.5}$$

which also describes the composition of two vector spaces $V \otimes W$.

A spanning set for a vector space V is a set of vectors $\{|v_1\rangle, \ldots, |v_n\rangle\}$ such that any vector $|v\rangle$ in the vector space can be written as linear combination $|v\rangle = \sum_i a_i|v_i\rangle$. For instance, a *qubit* lies in the vector space $\mathbb{C}^2$, and a spanning set of this vector space is

$$|v_1\rangle = \begin{bmatrix}1\\0\end{bmatrix}; \quad |v_2\rangle = \begin{bmatrix}0\\1\end{bmatrix}. \tag{2.6}$$

Any vector $|v\rangle$ in this vector space

$$|v\rangle = \begin{bmatrix}a_1\\a_2\end{bmatrix} \tag{2.7}$$

can be written as a linear combination $|v\rangle = a_1|v_1\rangle + a_2|v_2\rangle$.

A set of vectors is *linearly dependent* if there is a nontrivial linear combination of the vectors that equals the zero vector. If it is not linearly dependent, it is *linearly independent*. A set of linearly independent vectors is a *basis* for V. The minimum number of elements in the basis is the *dimension* of V.

A *linear operator* A between two vector spaces $V \in \mathbb{C}^n$ and $W \in \mathbb{C}^m$ is defined to be any function $A : V \to W$:

$$A\left(|v\rangle\right) = \sum_i a_i A\left(|v_1\rangle\right). \tag{2.8}$$

The most convenient way to understand linear operators on a finite-dimensional vector space is in terms of their *matrix representations*: the operator A sending vectors from V to W can be viewed as the matrix multiplication of $m \times n$ matrix $\boldsymbol{A}$ by a vector in V.

Some important linear operators are the *identity operator* $I|v\rangle = |v\rangle$ and the *zero operator* $0|v\rangle = 0$.

The *trace* of a square matrix $\boldsymbol{A}$, denoted $\mathrm{Tr}(\boldsymbol{A})$, is defined to be the sum of elements on the main diagonal of $\boldsymbol{A}$.

An operator A is *positive semi-definite* if $\forall|\psi\rangle \neq 0, \langle A\psi|\psi\rangle = \langle\psi|A|\psi\rangle \geq 0$.

A *hermitian operator* A is mathematically a self-adjoint operator $A = A^\dagger$. In its matrix representation, $.^\dagger$ means conjugate transpose of the matrix.

Suppose that vector spaces V, W, X and two linear operators $A : V \to W$ and $B : W \to X$, the *composition* of B with A is written as $BA|v\rangle$ and the application order is first A then B.

An *eigenvector* of a linear operator A on a vector space is a non-zero vector $|v\rangle$ such that

$$A|v\rangle = v|v\rangle, \tag{2.9}$$

where $v \in \mathbb{C}$ known as the *eigenvalue* of A.

2.1.2 Quantum state

The most fundamental element in quantum mechanics is the quantum state, which can be defined as a vector in a Hilbert space:

$$|\psi\rangle \in \mathbb{C}^n, \tag{2.10}$$

and $\langle\psi|\psi\rangle = 1$.

Another way to describe the quantum state is by using the *density operator*. A density operator ρ is a positive semi-definite, hermitian operator, which is defined in matrix representation as

$$\rho = \sum_j p_j |\psi_j\rangle\langle\psi_j|, \tag{2.11}$$

where $\rho \geq 0$ and $\mathrm{Tr}(\rho) = 1$.

A density operator represents a *pure state* if and only if it has purity one $\mathrm{Tr}(\rho^2) = 1$. Otherwise, it is a *mixed state*.

For a mixed state, the *quantum state fidelity* is defined between two density matrices ρ and σ [73]:

$$F(\rho, \sigma) = \left(\mathrm{tr}\sqrt{\sqrt{\rho}\sigma\sqrt{\rho}}\right)^2. \tag{2.12}$$

It expresses the probability that one state will pass a test to identify as the other []. It can also be considered as a measure of similarity between two states, indicating how close they are. For two pure states, given two unit vectors $|\psi\rangle$ and $|\phi\rangle$, the quantum state fidelity becomes:

$$F(\psi, \phi) = |\langle\phi|\psi\rangle|^2. \tag{2.13}$$

We will use this definition in our numerical simulations and call it fidelity for short.

2.1.3 Quantum evolution

The deterministic evolution of a quantum state of a quantum system is described by a *unitary transformation*:

$$|\psi'\rangle = U|\psi\rangle, \tag{2.14}$$

where the state $|\psi\rangle$ at time t_1 is related to the state $|\psi'\rangle$ at time t_2 by a unitary operator U which depends only on the time t_1 and t_2.

For example, the Pauli matrices

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \tag{2.15}$$

$$\sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \tag{2.16}$$

$$\sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \tag{2.17}$$

are unitary operators on a single qubit that perform a π rotation around an axis.

If we do not give two specific times t_1 and t_2 and want to describe the evolution of a quantum system in *continuous time*, we should use the *Schrödinger equation*:

$$i\hbar\frac{\partial}{\partial t}|\psi\rangle = \mathcal{H}|\psi\rangle, \tag{2.18}$$

where $\hbar$ is the reduced Planck constant, $\mathcal{H}$ is a fixed Hermitian operator known as the *Hamiltonian* of the system. If we know the Hamiltonian of a closed system, its dynamics are governed completely by this equation.

Because the Hamiltonian is a Hermitian operator (all eigenvalues are real), it can be expressed as a spectral decomposition

$$H = \sum_E E|E\rangle\langle E|, \tag{2.19}$$

where energy eigenvalues E correspond to normalized eigenvectors $|E\rangle$.

2.1.4 Observable

The *observable* is a linear Hermitian operator associated with a physical property on a Hilbert space, such as position and momentum. The corresponding physical property is measurable.

2.1.5 Measurement

A collection of measurement operators describes the *measurement*. Quantum measurement is based on the *Born rule*, which describes the probability of a given result in a measurement. If a quantum state $|\psi\rangle$ is measured by a probability operator M_m, the probability that the result m occurs is

$$p(m) = \langle\psi|\boldsymbol{M}_m|\psi\rangle, \tag{2.20}$$

where $\boldsymbol{M}$ is semi-definite, and the completeness guarantee that all possible outcomes' probability sums up to 1 ($\sum_m \boldsymbol{M}_m = \mathbb{1}$).

The *projective measurement* is the collection of measurement operators $\boldsymbol{P}_m$ equivalent to an observable, and $\boldsymbol{P}_m^2 = \boldsymbol{P}_m$. We call them *projectors* with the corresponding eigenvalue m.

2.1.6 Heisenberg uncertainty principle

The commutator of two operators A and B is defined as

$$[A, B] = AB - BA. \tag{2.21}$$

If this commutator is zero and both of the operators are Hermitian, then the observables A and B are said to be compatible, which means that one can measure them in any order without influencing the result. The commutator of an operator A and the operator product BC can be easily obtained:

$$[A, BC] = B[A, C] + [A, B]C. \tag{2.22}$$

One important commutator is the position operator q and the momentum operator p:

$$[q, p] = i\hbar. \tag{2.23}$$

They do not commute with each other, which means that we cannot measure them at the same time with arbitrary accuracy. Or in other words, the measurement of one will interfere with another one. This is what we call the uncertainty principle.

A more general version of the uncertainty principle is:

$$(\Delta A)^2(\Delta B)^2 \geq |[A, B]|^2/4, \tag{2.24}$$

where $(\Delta A)^2$ and $(\Delta B)^2$ denote the variances of the measured values defined as

$$(\Delta O)^2 = \langle O^2\rangle - \langle O\rangle^2, \tag{2.25}$$

and the $\langle . \rangle$ denotes the expectation of the operator.

2.2 Quantization of a harmonic oscillator

In quantum optics, the harmonic oscillator is an important model to characterize a mode. This section will start from the Hamiltonian and introduce the creation and annihilation operators.

The concept of modes [74] consists of two aspects: the modes are solutions for the propagation of light; the number of photons in the different modes describe the transport of energy or information. In this thesis, a *mode* refers to a single electromagnetic field characterized by its frequency, polarization, phase and etc. Its classical field energy, Hamiltonian $\mathcal{H}$, is equivalent to a harmonic oscillator of unit mass, so we use a harmonic oscillator to characterize it.

2.2.1 Hamiltonian

The Hamiltonian operator of a one-dimensional harmonic oscillator can be written as:

$$\mathcal{H} = \frac{1}{2}(p^2 + \omega^2 q^2), \tag{2.26}$$

where ω is the frequency of the mode, and the q and p are canonical position and momentum operators, which are Hermitian and correspond to the observables.

2.2.2 Canonical commutation relation

The Canonical Commutation Relation (CCR) describes a pair of self-adjoint position q and momentum operators p that satisfy

$$[q, p] = i\hbar. \tag{2.27}$$

2.2.3 Ladder operators

Paul Dirac proposed the *ladder operator* method in order to obtain the energy eigenvalues without solving the differential equation. The two ladder operators a and its adjoint $a^\dagger$ are defined from the quadrature operators q and p:

$$a = \sqrt{\frac{1}{2\hbar\omega}}\,(\omega q + ip)\,, \tag{2.28}$$

$$a^\dagger = \sqrt{\frac{1}{2\hbar\omega}}\,(\omega q - ip)\,. \tag{2.29}$$

Note that a and $a^\dagger$ do not commute:

$$[a, a^\dagger] = 1. \tag{2.30}$$

These two operators, a and its adjoint $a^\dagger$, are not Hermitian, so they are not observables.
As a result, the Hamiltonian operator is written as

$$\mathcal{H} = \hbar\omega\left(a^\dagger a + \frac{1}{2}\right). \tag{2.31}$$

We can also define the *number operator* as:

$$N = a^\dagger a, \tag{2.32}$$

and its name comes after the property $N|n\rangle = n|n\rangle$. So that the Hamiltonian operator becomes to

$$\mathcal{H} = \hbar\omega\left(N + \frac{1}{2}\right). \tag{2.33}$$

We denote energy eigenstates as $\{|n\rangle, n \in \mathbb{N}\}$, also known as the *Fock states*, with the energy eigenvalue E_n. Then we have

$$\mathcal{H}|n\rangle = \hbar\omega\left(N + \frac{1}{2}\right)|n\rangle = E_n|n\rangle. \tag{2.34}$$

When applying both ladder operators on the eigenstates, we have:

$$a^\dagger|n\rangle = \sqrt{n+1}|n+1\rangle, \tag{2.35}$$

$$a|n\rangle = \sqrt{n}|n-1\rangle. \tag{2.36}$$

$a^\dagger$ adds the photon number of the state, while a removes one. We refer to them as *creation* and *annihilation operators*.

If we continue to apply the annihilation operators on the state $|n\rangle$, we will reach the smallest eigenvalue 0:

$$n = \langle n|N|n\rangle = \langle n|a^\dagger a|n\rangle = (a^\dagger|n\rangle)^\dagger a|n\rangle \geq 0. \tag{2.37}$$

So that we have a ground state $|0\rangle$ with the lowest energy $\frac{\hbar\omega}{2}$ and such a state is referred to as the *vacuum state*.

In the second quantization picture, the Fock state $|n\rangle$ represents the presence of n energy quanta for a bosonic quantum field. In this thesis, we work with photons, and the Fock state $|n\rangle$ therefore represents the presence of n photons.

2.3 Fock space representation

The Fock state $|n\rangle$, or *number state* is the energy eigenstates of the harmonic oscillator. The *Fock basis* is a set of Fock states $\{|0\rangle, |1\rangle, \dots\}$. The vector space spanned from the Fock basis is a *Fock space*.

So, any pure quantum state $|\psi\rangle$ can be expressed in the Fock space V as

$$|\psi\rangle = \sum_{n=0}^{\infty} c_n|n\rangle, \tag{2.38}$$

where $c_n \in \mathbb{C}$ and $|c_n|^2$ denotes the probability of having n photons if it is a normalized vector. The Fock representation of this state is the column vector:

$$|\psi\rangle = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \end{bmatrix}. \tag{2.39}$$

The *Fock space representation* can also be used to describe the mixed states, transformations, and measurements.

The Fock representation of a mixed state with a density operator ρ is the corresponding matrix form $V \otimes V$

$$\begin{bmatrix} \rho_{00} & \rho_{01} & \cdots \\ \rho_{10} & \cdots & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix}, \tag{2.40}$$

where $\rho_{ij} \in \mathbb{C}$. The same as the pure state case, if it is a normalized matrix, the $|\rho_{ii}|^2$ denotes the probability of having the projector $|i\rangle\langle i|$.

The vector space of the multi-mode Fock representation is the tensor product of the single-mode ones, and we use the *tensor* to express the elements in its Fock representation, shown in Tab. 2.1.

The tensor has two parameters to identify:

- the dimension (the number of vector spaces).
- the cutoff (the number of basis elements for each vector space).

A scalar is a 0-dimensional tensor. A tensor with one dimension can be thought of as a vector, a tensor with two dimensions is a matrix, etc.

tensor	dimension	cutoff(s)
pure state	1	C
mixed state	2	[C,C]
single-mode operator	2	[C,C]
M-mode operator	2M	[C,C,...]
M-mode channel	4M	[C,C,C,C,...]

Table 2.1: Fock representations for each element..

2.4 Fock-Bargmann representation

Irving Segal and Valentine Bargmann introduced a new way to define the Fock space. The Fock space is defined with the Fock basis in discrete, while the *Segal-Bargmann space*, also known as *Fock-Bargmann space*, is the space of holomorphic functions F in n complex variables satisfying a square-integrability condition with respect to a Gaussian measure:

$$\mathcal{F}_n = \left\{ F : F \quad \text{is entire on} \quad \mathbb{C}^n \quad \text{and} \quad ||F||_{\mathcal{F}}^2 = \int |F(z)|^2 e^{-\pi|z|^2} \mathrm{d}z < \infty \right\}. \tag{2.41}$$

Fock-Bargmann representation is another interesting realization of the infinite-dimensional representation to analyze Hilbert space. The states and operators are presented as holomorphic functions in this representation. This holomorphic characteristic allows us to do more analytic analysis than only in the Fock space.

We will first give some basic definitions from the book by Folland [75]. And we list all the elements in Tab. 2.2 compared with the Fock photon-number space to understand the Fock-Bargmann representation. Then, finally, some more details about this representation will be discussed.

2.4.1 Bargmann transform

The Bargmann transformation is defined as:

$$Bf(z) = 2^{n/4} \int f(x) e^{2\pi x z - \pi x^2 - \frac{\pi}{2} z^2} \mathrm{d}x, \quad \text{for} \quad z \in \mathbb{C}^n. \tag{2.42}$$

B is a unitary operator from $L^2(\mathbb{R})$ to $\mathcal{F}_n$. Bf is an entire analytic function on $\mathbb{C}^n$ because of the integral over all $x \in \mathbb{C}^n$.

2.4.2 Basis

The orthonormal basis on $\mathcal{F}_n$ can be defined

$$\zeta_\alpha(z) = \sqrt{\frac{\pi^{|\alpha|}}{\alpha!}} z^\alpha, \tag{2.43}$$

where α are positive integers.

2.4.3 Reproducing kernel

If the map $F \to F(z)$ is a bounded linear functional on $\mathcal{F}_n$ for each z, there is a E_z such that:

$$F(z) = \langle F, E_z \rangle_{\mathcal{F}}, \tag{2.44}$$

and we know that $F(z)$ can also be written as:

$$F(z) = \int e^{\pi z w^*} F(w) e^{-\pi|w|^2} \mathrm{d}w. \tag{2.45}$$

We call the term $E_z(w^*) = e^{\pi z w^*}$ as the reproducing kernel. Note that the scalar product on $\mathcal{F}_n$ is defined by $\langle \cdot, \cdot \rangle_{\mathcal{F}}$, which is not the inner product as we usually understand.

The existence of reproducing kernel helps to write every bounded operator on $\mathcal{F}_n$ as an integral operator.

2.4.4 Ladder operators

Given a function $F \in \mathcal{F}_n$, the ladder operators A_j and A_j^* can be expressed as

$$A_j F = \frac{1}{\sqrt{\pi}} \frac{\partial F}{\partial z_j}, \tag{2.46}$$

$$A_j^* F = \sqrt{\pi} z_j F. \tag{2.47}$$

A_j decrements the power of the variable z by deriving it, and A_j^* increments it by multiplying.

These two operators satisfy the CCR:

$$[A_j, A_k] = [A_j^*, A_k^*] = 0, \quad [A_j, A_k^*] = \delta_{jk} I. \tag{2.48}$$

If we act them on the basis, we have:

$$A_j \zeta_\alpha = \sqrt{\alpha_j} \zeta_{\alpha - 1_i}, \tag{2.49}$$

$$A_j^* \zeta_\alpha = \sqrt{\alpha_j + 1} \zeta_{\alpha + 1_j}. \tag{2.50}$$

A_j decrements the power of the basis on the variable z, and A_j^* increments it.

2.4.5 Operator's kernel

If T is a bounded operator on $\mathcal{F}_n$, the kernel for T can be written as $K_T(z, w^*) = T E_w(z)$ on $\mathbb{C}^{2n}$, and the application of this operator can be written as:

$$TF(z) = \int K_T(z, w^*) |F(w)| e^{-\pi |w|^2} dw \quad \text{for all} \quad F \in \mathcal{F}_n \quad \text{and} \quad z \in \mathbb{C}^n. \tag{2.51}$$

2.4.6 Comparison in general

In Table 2.2, we compare the elements in the Fock-Bargmann representation and Fock photon-number space representation.

As we explained before, they are connected by the Bargmann transformation:

$$L^2(\mathbb{R}^n) \to L^2(\mathbb{C}^n, e^{-\pi |z|^2 dz}), \tag{2.52}$$

where dz is the Lebesgue measure on $\mathbb{C}^n$.

For instance, to describe a single photon state, we would have an analytic function z^1 in the Fock-Bargmann space, while $|1\rangle$ in the Fock photon-number space. They both have the annihilation and creation operators and the commutation relations between them hold.

2.4.7 Some special examples for Fock-Bargmann representation

The ket state:

$$|g\rangle \to F(|g\rangle, z) = e^{\frac{1}{2}|z|^2} \langle z^* | g \rangle = \sum_{n=0}^{\infty} g_n \frac{z^n}{\sqrt{n!}}. \tag{2.53}$$

The coherent state:

$$|\alpha\rangle \to \exp(-\frac{1}{2}|\alpha|^2 + \alpha z). \tag{2.54}$$

	Fock-Bargmann	Fock photon-number
basis	$\zeta_\alpha(z) = \sqrt{\frac{\pi^{\lvert\alpha\rvert}}{\alpha!}} z^\alpha$, $\lvert\alpha\rvert \geq 0$	$\lvert n\rangle$
CCR	$[A_j, A_k] = [A_j^*, A_k^*] = 0$, $[A_j, A_k^*] = \delta_{jk} I$	$[a_i, a_k] = [a_i^*, a_j^*] = 0$, $[a, a^\dagger] = 1$
	$A_j \zeta_\alpha = \sqrt{\alpha_j}\zeta_{\alpha - 1_i}$, $A_j^* \zeta_\alpha = \sqrt{\alpha_j + 1}\zeta_{\alpha + 1_j}$	$a\lvert n\rangle = \sqrt{n}\lvert n-1\rangle$, $a^\dagger\lvert n\rangle = \sqrt{n+1}\lvert n+1\rangle$
Arbitrary state	$F(z) = \sum_\alpha a_\alpha \sqrt{\frac{\pi^{\lvert\alpha\rvert}}{\alpha!}} z^\alpha = \langle F, E_z\rangle_{\mathcal{F}}$	$\lvert\psi\rangle = \sum_n a_n \lvert n\rangle$
Arbitrary operator	$K_T(z, w) = TE_w(z)$	$U_{m,n}$

Table 2.2: Dictionary for Bargmann-Fock space and Fock photon-number space.

Chapter 3

Quantum information with continuous variables

Unlike the Fock representation using the discrete photon number basis, the phase space representation offers a compact and efficient formalism to deal with quantum states as well. In phase space representation, the states are defined with the quadrature operators q and p, which are continuous variables (CVs).

3.1 Phase space representation

3.1.1 CCR in a compact way

Given an M-mode quantum system, the annihilation and creation operators a_j, $a_j^\dagger$; $j \in \{1, 2, \ldots, M\}$ satisfy the CCR:

$$[a_i, a_j^\dagger] = \delta_{ij}, \quad [a_i, a_j] = [a_i^\dagger, a_j^\dagger] = 0. \tag{3.1}$$

We can express these relations in a compact way by defining a vector of annihilation and creation operators $\boldsymbol{z} = (a_1, \ldots, a_M, a_1^\dagger, \ldots, a_M^\dagger)$, so that we can write

$$[z_i, z_j^\dagger] = Z_{ij}, \tag{3.2}$$

with

$$\boldsymbol{Z} = \begin{pmatrix} \mathbb{1}_M & \mathbf{0}_M \\ \mathbf{0}_M & -\mathbb{1}_M \end{pmatrix}. \tag{3.3}$$

An alternative way to describe the system is to use the hermitian position q and momentum p operators, which can be obtained from Eq.(2.29) and Eq.(2.28) by setting the frequency $\omega = 1$:

$$q_j = \sqrt{\tfrac{\hbar}{2}}(a_j^\dagger + a_j), \quad p_j = i\sqrt{\tfrac{\hbar}{2}}(a_j^\dagger - a_j). \tag{3.4}$$

We can group these operators into a quadrature vector $\boldsymbol{r} = (q_1, \ldots, q_M, p_1, \ldots, p_M)$ so that $\boldsymbol{r}$ is related to $\boldsymbol{z}$ by the unitary matrix $\boldsymbol{W}$:

$$\boldsymbol{r} = \sqrt{\hbar}\boldsymbol{W}\boldsymbol{z}, \tag{3.5}$$

where

$$\boldsymbol{W} = \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbb{1}_M & \mathbb{1}_M \\ -i\mathbb{1}_M & i\mathbb{1}_M \end{pmatrix}, \tag{3.6}$$

where $i = \sqrt{-1}$ is the imaginary unit.

Combining Eq. (3.5) and Eq. (3.2), we have:

$$[r_j, r_k] = \hbar(\boldsymbol{W}^\dagger \boldsymbol{Z} \boldsymbol{W})_{jk} = i\hbar\Omega_{jk}, \tag{3.7}$$

where $\boldsymbol{\Omega}$ is the skew-symmetric matrix:

$$\boldsymbol{\Omega} = \begin{pmatrix} \mathbf{0}_M & \mathbb{1}_M \\ -\mathbb{1}_M & \mathbf{0}_M \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \otimes \mathbb{1}_M, \tag{3.8}$$

which is central to the description of the symplectic group (see more details in Appendix B).

3.1.2 Phase space

Phase space is a real space of dimension $2M$ with coordinates of position variable $\boldsymbol{q} = (q_1, \ldots, q_M)$ and momentum variable $\boldsymbol{p} = (p_1, \ldots, p_M)$.

In phase space, the quantum state is described by a quasiprobability distribution, and the operator multiplication is replaced by a star product.

The phase space description offers a compact and efficient formalism to deal with Gaussian objects. Before going into a more detailed introduction to Gaussian objects, we will give some definitions in phase space.

3.1.3 Coherent state

The coherent state is an interesting class of quantum states with the dynamics most closely resembling the oscillatory behavior of a classical harmonic oscillator as the state of minimum uncertainty. It can be easily and clearly described in phase space formalism. A coherent state is also easy to generate experimentally, in contrast with the Fock state or quadrature eigenstate.

Mathematically, a coherent state $|\alpha\rangle$ is defined as the eigenstate of the annihilation operator a with the corresponding eigenvalue α:

$$a|\alpha\rangle = \alpha|\alpha\rangle, \tag{3.9}$$

where $\alpha \in \mathbb{C}$.

In Fig. 3.1, a coherent state in phase space is an uncertainty circle located at the point $\alpha = \frac{1}{\sqrt{2\hbar}}(q + ip)$.

We need to emphasize that the notation $|\alpha\rangle$ here does not refer to a Fock state. The exception is the vacuum state which is both a Fock state and a coherent state. The expression $|\alpha\rangle$ with $\alpha = 1$ represents a coherent superposition of Fock number states $|n\rangle$ with a mean of unity photon number (the distribution is Poissonnian and relative phase 0), while the state $|n\rangle$ with $n = 1$ is the single-photon Fock state, shown in Fig. 3.2.

The coherent state can be obtained by displacing a state from the original point to a location α in phase space. We can write this with the displacement operator $\mathcal{D}(\alpha)$:

$$|\alpha\rangle = e^{\alpha a^\dagger - \alpha^* a}|0\rangle = \mathcal{D}(\alpha)|0\rangle, \tag{3.10}$$

which is also called the Weyl operator. The displacement operator is unitary and has the following properties:

$$\mathcal{D}(\alpha) = \mathcal{D}^\dagger(-\alpha) = \mathcal{D}^{-1}(-\alpha). \tag{3.11}$$

We can then write the coherent state in the Fock basis as:

$$|\alpha\rangle = e^{-|\alpha|^2/2} \sum_{n=0}^{\infty} \frac{\alpha^n a^{\dagger n}}{\sqrt{n!}}|0\rangle = e^{-|\alpha|^2/2} \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}}|n\rangle. \tag{3.12}$$

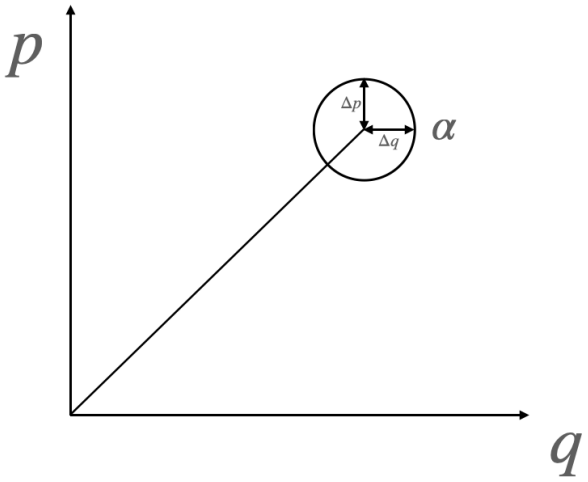


Figure 3.1: Coherent state in phase space.

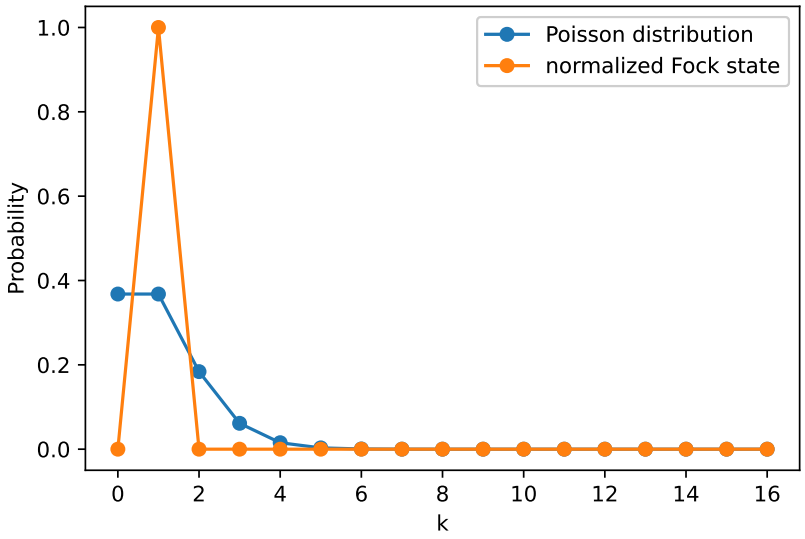


Figure 3.2: Poisson distribution with mean 1 and the single-photon Fock state.

If we apply the displacement operator to the annihilation and creation operator, we have the following:

$$\mathcal{D}^\dagger(\alpha) a \mathcal{D}(\alpha) = a + \alpha, \tag{3.13}$$

$$\mathcal{D}^\dagger(\alpha) a^\dagger \mathcal{D}(\alpha) = a^\dagger + \alpha^*. \tag{3.14}$$

Also, we can deduce the action of the displacement operator on the quadrature operators:

$$\mathcal{D}^\dagger(\alpha) q \mathcal{D}(\alpha) = q + \sqrt{2\hbar}\mathrm{Re}(\alpha), \tag{3.15}$$

$$\mathcal{D}^\dagger(\alpha) p \mathcal{D}(\alpha) = p + \sqrt{2\hbar}\mathrm{Im}(\alpha). \tag{3.16}$$

where $\mathrm{Re}(\alpha)$ and $\mathrm{Im}(\alpha)$ refers respectively to the real and imaginary part of α. We interpret the displacement operator as displacing the state $\mathrm{Re}(\alpha)$ along the quadrature q, and $\mathrm{Im}(\alpha)$ along the quadrature p in phase space.

Coherent states are often referred to the quasi-classical states since they reach the minimum of the Heisenberg uncertainty:

$$\Delta^2 q = \langle q^2 \rangle - \langle q \rangle^2 = \frac{1}{2}, \tag{3.17}$$

$$\Delta^2 p = \langle q^2 \rangle - \langle p \rangle^2 = \frac{1}{2}. \tag{3.18}$$

Two coherent states are not orthogonal to each other:

$$\langle \alpha | \beta \rangle = e^{-\frac{1}{2}(|\alpha|^2 + |\beta|^2)} \sum_{n=0}^{\infty} \frac{(\alpha^* \beta)^n}{n!} \neq 0. \tag{3.19}$$

So that we can not distinguish two coherent states perfectly.

Coherent states also satisfy the completeness relation:

$$\frac{1}{\pi} \int_{\mathbb{C}} d^2\alpha |\alpha\rangle\langle\alpha| = 1, \tag{3.20}$$

where $d^2\alpha = d\mathrm{Re}(\alpha) d\mathrm{Im}(\alpha)$.

If we have a M-mode coherent state:

$$\frac{1}{\pi^M} \int_{\mathbb{C}} d^{2M}\alpha |\boldsymbol{\alpha}\rangle\langle\boldsymbol{\alpha}| = \mathbf{1}, \tag{3.21}$$

where $d^{2M}\boldsymbol{\alpha} = d\mathrm{Re}(\boldsymbol{\alpha}) d\mathrm{Im}(\boldsymbol{\alpha})$.

3.1.4 Characteristic function

The characteristic function of a state with density matrix ρ is defined as:

$$\chi(\alpha; \rho) = \mathrm{Tr}(\mathcal{D}(\alpha)\rho). \tag{3.22}$$

3.1.5 Different phase space distributions

There are several different phase space distributions to describe the quantum state.

Wigner distribution

The Wigner distribution $W(q, p)$ of a quantum state with density matrix ρ is defined as:

$$W(q, p) = \int \frac{1}{\pi\hbar} \langle q - y | \rho | q + y \rangle e^{-2ipy/\hbar} dy, \tag{3.23}$$

where (q, p) is the conjugate variable pair in phase space.

The Wigner distribution or Wigner function is an important tool for studying quantum mechanics in phase space. Normally, the Wigner function is a real-valued function, and it does take on negative values when it has quantum properties that do not correspond to classical properties.

Husimi Q distribution

Another quasiprobability distribution is the Husimi Q in phase space of a quantum state. It can be calculated by:

$$Q(\alpha) = \frac{1}{\pi}\langle\alpha|\rho|\alpha\rangle, \tag{3.24}$$

which can be considered a trace of the density matrix ρ over the coherent state $\{|\alpha\rangle\}$.

Glauber–Sudarshan P distribution

Glauber-Sudarshan P distribution is defined implicitly as:

$$\rho = \int P(\alpha)|\alpha\rangle\langle\alpha| d^2\alpha, \quad d^2\alpha = d\mathrm{Re}(\alpha) d\mathrm{Im}(\alpha). \tag{3.25}$$

3.1.6 Comparison between different representations

The representation of a pure quantum state $|\psi\rangle$ can be interpreted as projecting the state into different bases. For example, the wavefunction on position and momentum spaces:

$$\langle q|\psi\rangle, \quad \langle p|\psi\rangle. \tag{3.26}$$

The representation of a mixed quantum state ρ can be interpreted as tracing the density operator with an operator-valued function in phase space:

$$\mathrm{Tr}\left[\rho\Delta(\alpha)\right], \tag{3.27}$$

where α is the coherent state.

In [76], Husimi Q, Wigner, and P functions are in the same family of phase space functions parametrized by a number s, corresponding to the values $+1, 0$, and -1, respectively. For example, the $\Delta(\alpha)$ of Husimi Q function is

$$\Delta(\alpha) = |\alpha\rangle\langle\alpha|. \tag{3.28}$$

$\Delta(\alpha)$ is known as Stratonovich-Weyl (SW) kernel $\Delta^{(s)}(\alpha)$ with respect to s in [77].

With the same idea, we summarize the representations of pure state and mixed state in Tab. 3.1.

	$\langle z\|\psi\rangle$	$\mathrm{Tr}[\rho\Delta(\alpha)]$
wavefunction	$\psi(q) = \langle q\|\psi\rangle$ $\widetilde{\psi}(p) = \langle p\|\psi\rangle$	
Husimi Q representation	$\langle\alpha\|\psi\rangle$	$Q(\alpha) = \mathrm{Tr}[\rho\Delta^{(+1)}(\alpha)]$
Wigner representation		$W(\alpha) = \mathrm{Tr}[\rho\Delta^{(0)}(\alpha)]$
P representation		$P(\alpha) = \mathrm{Tr}[\rho\Delta^{(-1)}(\alpha)]$
Fock representation	$\psi_n = \langle n\|\psi\rangle$	$\rho_{mn} = \mathrm{Tr}[\rho\|m\rangle\langle n\|]$
Fock-Bargmann representation	$e^{\frac{1}{2}\|\alpha\|^2}\langle\alpha\|\psi\rangle$	$F(\alpha) = e^{\|\alpha\|^2}\mathrm{Tr}[\rho\Delta^{(+1)}(\alpha)]$

Table 3.1: Comparison of different phase space representations.

3.2 Gaussian quantum mechanics

Gaussian quantum mechanics is a subset of quantum mechanics that is related to objects whose representations in phase space are Gaussians. Gaussian objects are Gaussian states, Gaussian unitaries and Gaussian channels. They are in the center of CV quantum systems because they are easy to manipulate in practice.

3.2.1 Gaussian states

Gaussian states are any states whose characteristic functions and quasi-probability distributions are Gaussian functions in phase space [78]. Some well-known examples are coherent states, squeezed states, thermal states, and the vacuum state (which is the only state which is at the same time Gaussian and a number eigenstate).

For a Gaussian state, we write the characteristic function in terms of its mean vector $\bar{\boldsymbol{r}}$ and covariance matrix $\boldsymbol{V}$ as [78]

$$\chi(\boldsymbol{s};\rho) = \exp\left[-\tfrac{1}{2}\boldsymbol{s}^T\boldsymbol{\Omega}^T\boldsymbol{V}\boldsymbol{\Omega}\boldsymbol{s} - i\bar{\boldsymbol{r}}^T\boldsymbol{\Omega}\boldsymbol{s}\right], \tag{3.29}$$

where

$$\bar{r}_i = \langle r_i\rangle, \tag{3.30}$$

$$V_{ij} = \frac{1}{2}\langle r_i r_j + r_j r_i\rangle - \bar{r}_i\bar{r}_j. \tag{3.31}$$

Note that the covariance matrix $\boldsymbol{V}$ is a real, symmetric, positive definite matrix with the basis of quadrature vector $\boldsymbol{r}$ defined in Eq. (3.5).

If we use the basis $\boldsymbol{z}$ of annihilation and creation operators $\boldsymbol{z}$ defined in Eq. (3.2), we find the mean vector $\bar{\boldsymbol{\mu}}$ and the covariance matrix $\boldsymbol{\sigma}$:

$$\bar{\mu}_i = \langle z_i\rangle = \frac{1}{\sqrt{\hbar}}\left(\boldsymbol{W}^\dagger\bar{\boldsymbol{r}}\right)_i, \tag{3.32}$$

$$\sigma_{ij} = \frac{1}{2}\langle z_i z_j^\dagger + z_j z_i^\dagger\rangle - \bar{\mu}_i\bar{\mu}_j^\dagger = \frac{1}{\hbar}(\boldsymbol{W}^\dagger\boldsymbol{V}\boldsymbol{W})_{ij}. \tag{3.33}$$

Compared with the real covariance matrix $\boldsymbol{V}$, we denote the $\boldsymbol{\sigma}$ as the complex covariance matrix.

For example the vacuum state $|0\rangle$, which satisfies $a_j|0\rangle = 0$, has zero mean vector and covariance matrix $\boldsymbol{V} = \frac{\hbar}{2}\mathbb{1}$.

In this thesis, we will write the phase space description of a Gaussian state as the pair $(\boldsymbol{V}, \bar{\boldsymbol{r}})$ or $(\boldsymbol{\sigma}, \bar{\boldsymbol{\mu}})$ depending on which basis we use.

state	mean vector $\bar{\boldsymbol{r}}$	covariance matrix $\boldsymbol{V}$
vacuum state $\|0\rangle$	0	$\frac{\hbar}{2}\mathbb{1}$
coherent state $\|\alpha\rangle$	$\sqrt{2\hbar}[\mathrm{Re}(\boldsymbol{\alpha}), \mathrm{Im}(\boldsymbol{\alpha})]$	$\frac{\hbar}{2}\mathbb{1}$
squeezed state $\|r\rangle$ (ζ is real)	0	$\frac{\hbar}{2}\begin{bmatrix} e^{-r} & 0 \\ 0 & e^{r}\end{bmatrix}$

Table 3.2: Some examples of Gaussian states

3.2.2 Gaussian operators

Gaussian unitaries are those that map Gaussian states to Gaussian states [78], thus in the Schrödinger picture, an input Gaussian state ρ is mapped to an output Gaussian state

$$\rho \mapsto \rho' = U_G\rho U_G^\dagger \tag{3.34}$$

Gaussian unitaries have polynomials of at most degree 2 in the quadrature (or equivalently in the creation and annihilation operators) as generators.

In the Heisenberg picture, a Gaussian unitary (parameterized by a $2M\times 2M$ $\boldsymbol{S}$ matrix and a real vector $\boldsymbol{d}$ with size $2M$) transforms the quadrature operators as follows

$$\boldsymbol{r} \mapsto \boldsymbol{r}' = U_G^\dagger\boldsymbol{r}U_G = \boldsymbol{S}\boldsymbol{r} + \boldsymbol{d}. \tag{3.35}$$

Since $\boldsymbol{r'}$ is obtained from $\boldsymbol{r}$ by unitary conjugation, it must satisfy the canonical commutation relations Eq. (3.7). This implies that the matrix $\boldsymbol{S}$ satisfies

$$\boldsymbol{S\Omega S}^T = \boldsymbol{\Omega}, \tag{3.36}$$

that is, $\boldsymbol{S}$ must be an element of the (real) symplectic group, $\boldsymbol{S} \in \mathrm{Sp}(2M, \mathbb{R})$.

In general, Gaussian unitaries transform the mean vector $\bar{r}$ and the covariance matrix V of a Gaussian state as:

$$(\boldsymbol{V}, \bar{\boldsymbol{r}}) \mapsto (\boldsymbol{V}', \bar{\boldsymbol{r}}') = (\boldsymbol{SVS}^T, \boldsymbol{S}\bar{\boldsymbol{r}} + \boldsymbol{d}). \tag{3.37}$$

A M-mode Gaussian unitary generated by a second-degree polynomial in the quadratures can be decomposed into a M-mode displacement $\mathcal{D}_{\boldsymbol{d}}$ and a M-mode unitary U generated by a strictly quadratic unitary that is responsible for the symplectic matrix $\boldsymbol{S}$ appearing in Eq. (3.35) and thus we can write [79]

$$U_G = \mathcal{D}_{\boldsymbol{d}} U(\boldsymbol{S}), \tag{3.38}$$

where $\mathcal{D}_{\boldsymbol{d}}$ is the displacement operator, parametrized by a real vector $\boldsymbol{d}$ of size $2M$. We can also express the M-mode displacement operator as the tensor product of the single-mode displacement operator, with a complex vector $\boldsymbol{\gamma}$ of size M:

$$\mathcal{D}(\boldsymbol{\gamma}) = \exp\left[\sum_{i=1}^{M} (\gamma_i a_i^\dagger - \gamma_i^* a_i)\right]. \tag{3.39}$$

The relation between them can be derived from Eq. (3.4):

$$\boldsymbol{d} = \sqrt{2\hbar}[\mathrm{Re}(\boldsymbol{\gamma}), \mathrm{Im}(\boldsymbol{\gamma})]. \tag{3.40}$$

Single-mode Gaussian unitaries

We will give the definitions of single-mode Gaussian unitaries, noting that their multi-mode version is just the tensor product extension of their single-mode version.

We have already defined the single-mode *displacement operator* $\mathcal{D}(\alpha)$ in the Eq. (3.10):

$$\mathcal{D}(\alpha) = e^{\alpha a^\dagger - \alpha^* a}. \tag{3.41}$$

It has

$$d = \sqrt{2\hbar}[\mathrm{Re}(\alpha), \mathrm{Im}(\alpha)], \tag{3.42}$$

and $\boldsymbol{S}_{\mathrm{disp}} = \mathbb{1}$ for the phase space representation.

The single-mode *rotation operator*

$$\mathcal{R}(\phi) = \exp\left[i\phi a^\dagger a\right], \tag{3.43}$$

and it has $\boldsymbol{d}_{\mathrm{rot}} = [0, 0]$ and

$$\boldsymbol{S}_{\mathrm{rot}} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}. \tag{3.44}$$

The single-mode *squeezing operator*

$$\mathcal{S}(\zeta) = \exp\left[\frac{1}{2}\zeta^* a^2 - \mathrm{H.c.}\right] \tag{3.45}$$

where $\zeta = re^{i\delta}$, and it has $\boldsymbol{d}_{\mathrm{sq}} = [0, 0]$ and

$$\boldsymbol{S}_{\mathrm{sq}} = \begin{bmatrix} \cosh(r) & -e^{-i\delta}\sinh(r) \\ -e^{i\delta}\sinh(r) & \cosh(r) \end{bmatrix}. \tag{3.46}$$

Other Gaussian unitaries

The **two-mode squeezing operator**

$$\mathcal{S}^{(2)}(\zeta) = \exp(r a_1^\dagger a_2^\dagger - H.c.), \tag{3.47}$$

where $r \in \mathbb{R}$. It has $\boldsymbol{d}_S^{(2)} = [0,0,0,0]$ and its symplectic matrix is

$$\boldsymbol{S}_{\mathcal{S}^{(2)}} = \begin{bmatrix} \cosh r & \sinh r & 0 & 0 \\ \sinh r & \cosh r & 0 & 0 \\ 0 & 0 & \cosh r & -\sinh r \\ 0 & 0 & -\sinh r & \cosh r \end{bmatrix}. \tag{3.48}$$

A M-mode *interferometer* is defined as [80]

$$\mathcal{W}(\boldsymbol{J}) = \exp\left[i \sum_{k,l=1}^{M} J_{k,l} a_k^\dagger a_l \right] \tag{3.49}$$

has $\boldsymbol{d}_{\text{intf}} = 0$, and

$$\boldsymbol{S}_{\text{intf}} = \begin{bmatrix} \text{Re}(\boldsymbol{U}) & -\text{Im}(\boldsymbol{U}) \\ \text{Im}(\boldsymbol{U}) & \text{Re}(\boldsymbol{U}) \end{bmatrix}. \tag{3.50}$$

where $\boldsymbol{U} = \exp[i\boldsymbol{J}]$ is a unitary matrix (since $\boldsymbol{J} = \boldsymbol{J}^\dagger$).

A particular instance of an interferometer is the *beamsplitter* $\mathcal{B}(\theta,\phi)$, parametrized in terms of transmission angle θ and a phase ϕ (the energy transmission is given by $\cos^2\theta$). In this case, we have

$$\boldsymbol{J} = i \begin{bmatrix} 0 & \theta e^{-i\phi} \\ -\theta e^{i\phi} & 0 \end{bmatrix}, \ \boldsymbol{U} = \begin{bmatrix} \cos\theta & -e^{-i\phi}\sin\theta \\ e^{i\phi}\sin\theta & \cos\theta \end{bmatrix}. \tag{3.51}$$

Note that our definition of interferometer immediately implies that $\mathcal{W}(\boldsymbol{J})|0\rangle = |0\rangle$ without any ambiguity in the global phase of the state on the right hand side.

Note that $\boldsymbol{S}_{\text{intf}} \in \text{Sp}(2n,\mathbb{R}) \cup \text{O}(2n) \cong U(n)$ where $\text{Sp}(2n,\mathbb{R})$ is the symplectic group, $\text{O}(2n)$ is the orthogonal group and $U(n)$ is the unitary group (More information is shown in Appendix B).

The most general Gaussian unitaries

The most general M-mode Gaussian unitary can be expressed as the combination of the fundamental Gaussian unitaries $\mathcal{D}_M(\alpha), \mathcal{S}_M(\zeta), \mathcal{R}_M(\theta)$ in an arbitrary order:

$$\mathcal{R}_M(\theta)\mathcal{D}_M(\alpha)\mathcal{S}_M(\zeta), \quad \mathcal{S}_M(\zeta)\mathcal{D}_M(\alpha)\mathcal{R}_M(\theta), \quad \ldots \tag{3.52}$$

The book [27] gives the commutation rules (11.176) to change the order of the fundamental unitaries.

Ma and Rhodes, in paper [80], proved that for a general M-mode quadratic Hamiltonian, a unitary operator $e^{-iH/2}$ can be written as:

$$U = e^{i\gamma_M}\mathcal{R}_M(\theta)\mathcal{D}_M(\alpha)\mathcal{S}_M(\zeta), \tag{3.53}$$

where $e^{i\gamma_M}$ is a phase factor, of which we give the explicit derivation in section 6.3.

3.2.3 Gaussian channels

A map Φ between density operators represents a deterministic physical dynamical process [78], which needs to be

- Linear: $\Phi(\alpha\rho + \beta\sigma) = \alpha\Phi(\rho) + \beta\Phi(\sigma)\forall\alpha, \beta \in \mathbb{C}$ and for all linear operators ρ and σ;
- Trace-preserving: $\mathrm{Tr}\left[\Phi(\rho)\right] = \mathrm{Tr}\left[\rho\right]$;
- Completely positive map (CP-map): $\Phi \otimes \mathcal{I}(|\psi\rangle\langle\psi|) \geq 0, \forall|\psi\rangle \in \mathcal{H}_A \otimes \mathcal{H}_B$, where Φ acts on linear operators on $\mathcal{H}_A$ and $\mathcal{I}$ is the identity super-operator (an operator acting between linear operators on a Hilbert space) on $\mathcal{H}_B$.

Such deterministic physical quantum maps are known as completely positive maps or "quantum channels".

A deterministic Gaussian quantum channel maps Gaussian states to Gaussian states. Such a channel is characterized by two matrices $\boldsymbol{X}$, $\boldsymbol{Y}$ and a vector $\boldsymbol{d}$. The action of the channel on a Gaussian state $(\boldsymbol{V}, \bar{\boldsymbol{r}})$ is

$$(\boldsymbol{V}, \bar{\boldsymbol{r}}) \mapsto (\boldsymbol{X}\boldsymbol{V}\boldsymbol{X}^T + \boldsymbol{Y}, \boldsymbol{X}\bar{\boldsymbol{r}} + \boldsymbol{d}). \tag{3.54}$$

where the matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$ need to satisfy

$$\boldsymbol{Y} + i\frac{\hbar}{2}\boldsymbol{\Omega} \geq i\frac{\hbar}{2}\boldsymbol{X}\boldsymbol{\Omega}\boldsymbol{X}^T. \tag{3.55}$$

More generally, the action of a Gaussian channel on the characteristic function of an arbitrary state amounts to

$$\chi(\boldsymbol{s}) \mapsto \chi'(\boldsymbol{s}) = \chi(\boldsymbol{X}\boldsymbol{s}) \exp\left(-\tfrac{1}{2}\boldsymbol{s}^T\boldsymbol{Y}\boldsymbol{s}\right). \tag{3.56}$$

Note that unitary channels such as Eq. (3.37) are special cases of a Gaussian channel where $\boldsymbol{Y} = 0$ and $\boldsymbol{X}$ is symplectic. More generally, when $\boldsymbol{X}$ is not symplectic and thus the channel is not unitary, the matrix $\boldsymbol{Y}$ represents added noise in the state.

Single-mode attenuator channel

An attenuator channel is a deterministic Gaussian CP-map

$$\boldsymbol{X} = \cos\theta\,\mathbb{1}_2 \quad \text{and} \quad \boldsymbol{Y} = (\sin\theta)^2 n_{th}\,\mathbb{1}_2, \tag{3.57}$$

where $\theta \in [0, 2\pi[$ and $n_{th} \leq 1$. If $n_{th} = 1$, it refers to a pure loss channel.

Single-mode amplifier channel

An amplifier channel is a deterministic Gaussian CP-map

$$\boldsymbol{X} = \cosh r\,\mathbb{1}_2 \quad \text{and} \quad \boldsymbol{Y} = (\sinh r)^2 n_{th}\,\mathbb{1}_2, \tag{3.58}$$

where $r \in [0, \infty[$ and $n_{th} \leq 1$. If $n_{th} = 1$, it refers to a quantum-limited amplifier.

Multi-mode lossy channel

An example of a multi-mode Gaussian channel is the lossy interferometer parametrized in terms of a transmission matrix $\boldsymbol{T}$ with a singular value upper-bounded by 1. For this channel, we find

$$\boldsymbol{X} = \begin{bmatrix} \mathrm{Re}(\boldsymbol{T}) & -\mathrm{Im}(\boldsymbol{T}) \\ \mathrm{Im}(\boldsymbol{T}) & \mathrm{Re}(\boldsymbol{T}) \end{bmatrix}, \tag{3.59}$$

$$\boldsymbol{Y} = \frac{\hbar}{2}\left(\mathbb{1}_{2M} - \boldsymbol{X}\boldsymbol{X}^T\right), \tag{3.60}$$

$$\boldsymbol{d} = 0. \tag{3.61}$$

Note that in the case where $\boldsymbol{T}$ is unitary, then $\boldsymbol{X}$ is symplectic and orthogonal, and thus $\boldsymbol{Y} = 0_{2M}$ recovering the results from the previous subsection about the multi-mode interferometer.

3.3 Non-Gaussian objects

Gaussian objects are a well-defined and restricted ensemble, while non-Gaussian objects are part of a substantial and diverse ensemble.

3.3.1 Non-Gaussian states

The paper [81] gives a full tutorial about the non-Gaussian states; it claims that the stellar rank and Wigner negativity are both quantifiers of non-Gaussianity.

We will optimize our quantum circuit to prepare different quantum states in our final task in section 9.2.2. There are non-Gaussian states such as cat state, Gottesman-Kitaev-Preskill (GKP) state, etc.

3.3.2 Non-Gaussian operators

Non-Gaussian operators are also vital in quantum optical circuits. Commonly we would choose the cubic phase gate and the Kerr gate.

The cubic phase gate is defined as:

$$V(\gamma) = \exp(i\frac{\gamma}{3}q^3), \tag{3.62}$$

where γ is proportional to the third-order nonlinearity of the medium, q is defined in Eqs.(2.28) and (2.29).

The Kerr gate is defined as:

$$K(\kappa) = \exp(i\kappa N^2), \tag{3.63}$$

where κ is Kerr constant of the Kerr medium, N is defined in Eq. (2.32).

We will discuss the implementation of the Kerr gate in section 9.4, which is regarded as the most promising non-Gaussian gate to be realized.

Chapter 4

Automatic differentiation and gradient descent algorithm for optimization

This chapter gives the definition of cost function and gradient. Then we introduce Automatic Differentiation (AD) and gradient descent algorithm.

One of the most used optimization algorithms is the gradient descent algorithm, which was first proposed in 1847 by Augustin-Louis Cauchy [36]. Gradient descent is finding the local minimum of a differentiable function closest to the starting point. It gives a way to adjust the parameters of a cost function during each step of the optimization and to minimize the cost function step by step.

4.1 Cost function

The cost function L is a scalar-valued differentiable function. The cost function quantifies the distance between the targeted result and the actual result obtained with the current parameters, and then, by learning from this distance, we can adjust the parameters.

4.2 Gradient

Given a scalar-valued differentiable function $L : \mathbb{C}^n \to \mathbb{R}$ or $L : \mathbb{R}^n \to \mathbb{R}$, the gradient of L with respect to the variables at point $\boldsymbol{p} = (x_1, \ldots, x_n)$ is defined as

$$\nabla_{\boldsymbol{p}} L = \begin{bmatrix} \frac{\partial L}{\partial x_1} \\ \cdots \\ \frac{\partial L}{\partial x_n} \end{bmatrix}, \tag{4.1}$$

where $\frac{\partial L}{\partial x_i}$ denotes the partial derivative of L with respect to x_i.

In the landscape of the function L, one can visualize that the gradient vector can be interpreted as the direction of the fastest increase of L. If the gradient is non-zero, the direction of the gradient is the direction where the function increases most quickly. The gradient thus plays a fundamental role in optimization theory, where it is used to minimize a cost function by gradient descent i.e., taking steps in the direction opposite to the gradient.

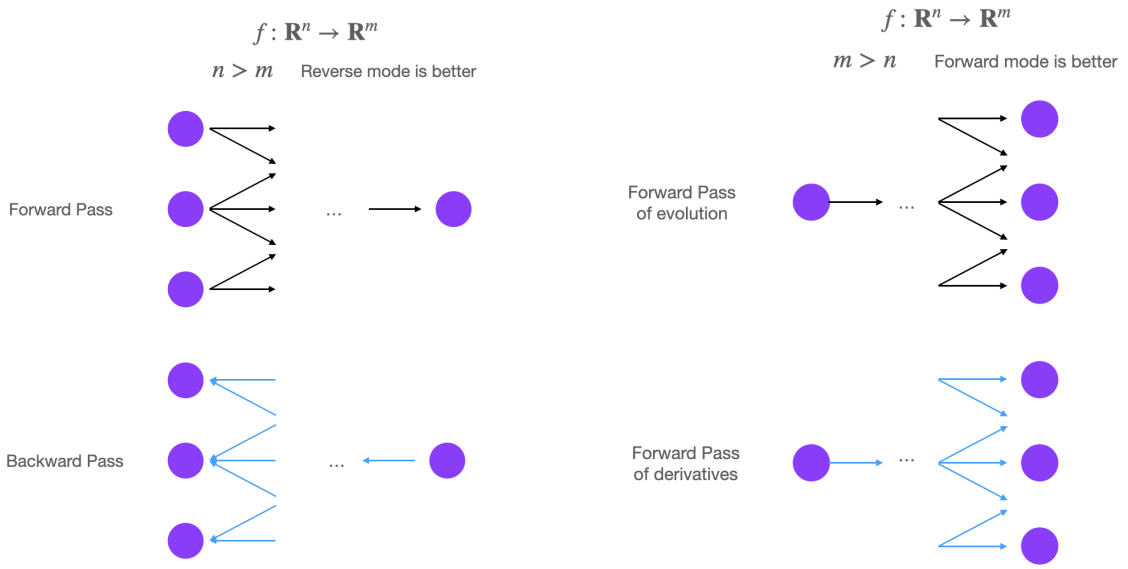
4.3 Introduction to the Automatic differentiation

Automatic differentiation (AD), also called algorithmic differentiation, computational differentiation, or auto-differentiation, is to give the computer a precise series of functions in the program and let it evaluate the derivative automatically. This is the technique we use in our work to calculate the gradients of each parameter of photonic quantum circuits.

AD has two distinct modes: forward mode and reverse mode. We show how these two modes work in Fig. 4.1. They are both computing the chain rule. The forward mode goes from the input to the output, while the reverse mode computes the gradients in the opposite direction.

To have more efficient computations, we are interested in pushing gradient computations from a few units towards many. The choice of mode depends on input and output dimensions n and m:

- $n >> m$: If the input vector is much larger than the output, the reverse mode is computationally cheaper (e.g. in a neural network classifier).
- $n << m$: If the output vector is much larger than the input, the forward mode is computationally cheaper (e.g. for a generative neural network).



(a) Reverse mode of AD.

(b) Forward mode of AD.

Figure 4.1: The comparison of reverse and forward mode of AD.

4.3.1 An example of the reverse mode

In this thesis, our optimization task corresponds to multiple inputs and a single cost function at the end, so it is better to use the reverse mode. Let's use an example to explain how the reverse mode works.

Supposing that we have a sequence of operations:

$$\begin{aligned} w_1 &= \mathcal{G}_1(\theta_1)x_1, \\ w_2 &= \mathcal{G}_2(\theta_2)x_2, \\ w_3 &= \mathcal{G}_3(\theta_3)(w_1 \otimes w_2), \\ y &= L(w_3). \end{aligned}$$

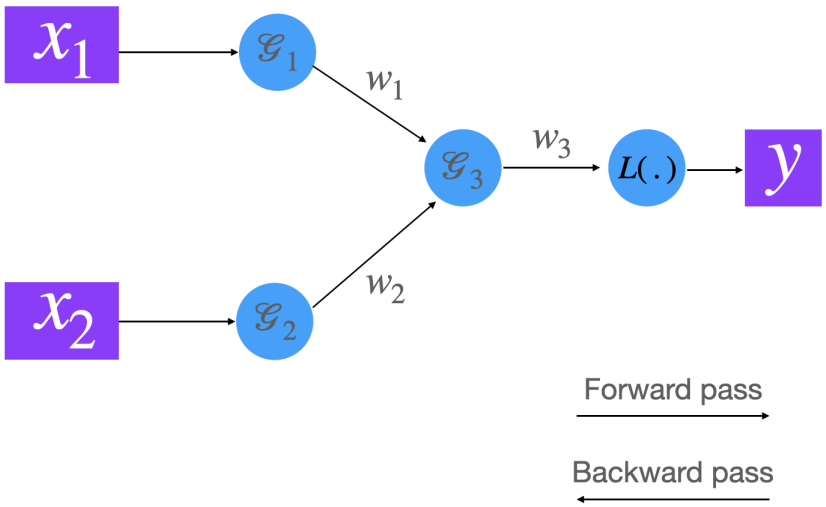


Figure 4.2: Adjoint graph of a sequence of operations to describe the reverse mode of AD.

We could also show this sequence of operations in an adjoint graph 4.2.

If we follow the arrows from left to right, this refers to the *forward pass*, which describes the input state x_1 and x_2 pass all operators and get the value of the cost function L at the end, which corresponds the application of gates in the parametrized quantum circuit.

In the other direction, the *backward pass* consists of the calculation of the derivatives, which shows the accumulation of the derivatives of each parameter in the circuit. Once we have the $\frac{\partial L}{\partial w_3}$ after evaluating the cost function L, the derivatives can be backpropagated following the backward pass in the graph:

$$\frac{\partial L}{\partial \theta_3} = \frac{\partial L}{\partial w_3}\frac{\partial w_3}{\partial \mathcal{G}_3}\frac{\partial \mathcal{G}_3}{\partial \theta_3}, \tag{4.2}$$

$$\frac{\partial L}{\partial \theta_2} = \frac{\partial L}{\partial w_3}\frac{\partial w_3}{\partial w_2}\frac{\partial w_2}{\partial \mathcal{G}_2}\frac{\partial \mathcal{G}_2}{\partial \theta_2}, \tag{4.3}$$

$$\frac{\partial L}{\partial \theta_1} = \frac{\partial L}{\partial w_3}\frac{\partial w_3}{\partial w_2}\frac{\partial w_2}{\partial \mathcal{G}_1}\frac{\partial \mathcal{G}_1}{\partial \theta_1}. \tag{4.4}$$

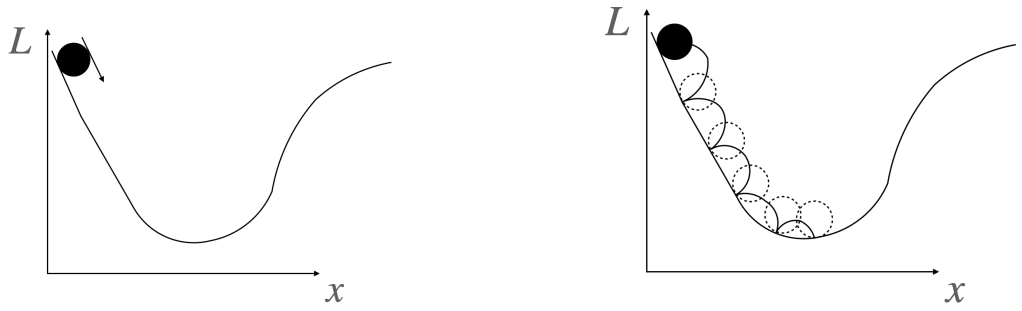
TensorFlow is one library that uses the reverse mode to realize AD. We will explain it in detail in section 7.2.2.

4.4 Gradient Descent algorithm

Gradient descent algorithm can be visualized as a ball rolling down a hill (as shown in Fig. 4.3a). The aim of the ball is to reach the lowest point.

The value of the cost function L changes with different values of parameters. The parameters update is repeated, and the value of L keeps getting smaller and smaller until it reaches the minimum, as the steps shown in Fig. 4.3b.

However, only one single parameter is rare, and usually the parameter space contains several parameters and exhibits a complicated geometry. This is one important reason to take geometry into consideration during our optimization tasks, which we managed in the generalization of the complex natural gradient algorithm in section 7.4 and the work of geodesic-based Riemannian manifold optimization in section 7.3.2.



(a) Ball down the hill

(b) Ball downhill with steps

Figure 4.3: Ball down from the hill as an example of the gradient descent algorithm.

4.5 Update rule

4.5.1 Real parameter update

For a real parameter θ, the gradient descent update uses the partial derivative of a real cost function L:

$$\theta \leftarrow \theta - \eta \frac{\partial L}{\partial \theta}, \tag{4.5}$$

where $\frac{\partial L}{\partial \theta}$ is the gradient that we can compute following the chain rule explained in the section 4.6, and η is the learning rate, which is an important hyper-parameter in the circuit optimization.

4.5.2 Complex parameter update

For a complex parameter ξ, the gradient descent update step should use the partial derivative of a real cost function L with respect to the *conjugate* of the parameter [82, 83]:

$$\xi \leftarrow \xi - \eta \frac{\partial L}{\partial \xi^*}. \tag{4.6}$$

Proof. The differential df of a complex-valued function $f(z)$ can be expressed as [84]

$$df = \frac{\partial f(z)}{\partial z} dz + \frac{\partial f(z)}{\partial z^*} dz^*. \tag{4.7}$$

When it refers to a real-valued function $L(\xi)$, the differential dL can be expressed as:

$$dL = 2\mathrm{Re}\left(\frac{\partial L(\xi)}{\partial \xi^*} d\xi^* \right), \tag{4.8}$$

dL is maximized for real-valued part $\frac{\partial L(\xi)}{\partial \xi^*} d\xi^*$ if the norm of $d\xi$ is fixed.

Hence, the gradient descent update step uses the partial derivative of a real-valued cost function L with respect to the conjugate of the parameter $\frac{\partial L}{\partial \xi^*}$ [84, 85]:

$$z \leftarrow z + 2 \frac{\partial L(\xi)}{\partial \xi^*} ds, \tag{4.9}$$

where ds is the real-valued differential and the steepest ascent point to the direction of $\frac{\partial L(\xi)}{\partial \xi^*}$.

□

This update rule goes back to the regular rule for a real parameter $\xi = \xi^*$.

To compute the gradient for the update, we need to treat complex variables and their conjugate as *independent* variables, which allows us to compute gradients of non-holomorphic functions [82].

4.5.3 Learning rate

The learning rate η is a hyperparameter that controls that controls the size of the step. Choosing the learning rate is challenging because if one chooses it too small, it may result in a long training process that could get stuck at some points, whereas a value too large may result in an unstable training process or jumping over the minimum it should arrive at. For these reasons, a proper learning rate is a trade-off between convergence speed and overshooting.

The learning rate schedule is introduced to adjust the learning rate with the progress of learning. Some learning rate schedules correspond to a pre-defined schedule, such as time-based decay, step decay, and exponential decay. (The function of the changes in the learning rate is defined.) Some other adaptive learning rate methods are proposed as well [86], such as Adaptive gradient algorithm (Adagrad) [87], Adadelta [88], Root Mean Squared Propagation (RMSprop) (unpublished, course notes[1]) and ADAptive Moment estimation (ADAM) [89].

4.6 Chain rule

The chain rule is a formula to calculate the derivative of the composition of any number of differentiable functions in terms of their derivatives and the functions. It can be extended to get the derivative of the composition of several differentiable functions.

An example here is to explain the chain rule of the two functions f and g. Suppose that there is a series of functions:

$$y = f(g(x)), \tag{4.10}$$

and one can also define the output of g as w:

$$w = g(x). \tag{4.11}$$

To get the $\frac{dy}{dx}$, the chain rule gives the intermediate calculations

$$\frac{dy}{dx} = \frac{dy}{dw}\frac{dw}{dx}. \tag{4.12}$$

The chain rule combines the derivatives with respect to the successive variables, starting from x, using the function $g(\cdot)$ and the function $f(\cdot)$ successively; each derivative corresponds to a single function's input and output (for instance for the function $f(\cdot)$), the derivative $\frac{dy}{dw}$ uses the input w and output y.

4.7 Jacobian-vector product and vector-jacobian product

The *Jacobian* is the matrix of partial derivatives of a vector-valued function of several variables. Given a function $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$ where first-order partial derivatives exist on $\mathbb{R}^n$, for each input $\boldsymbol{x}$, it gives an output $\boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^m$. The Jacobian matrix of $\boldsymbol{f}$ is defined as:

$$\boldsymbol{J} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial \boldsymbol{f}}{\partial x_1} \cdots \frac{\partial \boldsymbol{f}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}, \tag{4.13}$$

where each row vector in the final matrix is defined as the transpose of the gradient of the i component $\nabla^T f_i = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} \cdots \frac{\partial f_1}{\partial x_n} \end{bmatrix}$.

This Jacobian matrix is the core of our optimization task, as the gradient update depends on the product of the Jacobian matrix and a vector $\boldsymbol{v} \in \mathbf{R}^n$.

There are two forms of this product corresponding to the forward and reverse modes of AD:

[1] http://www.cs.toronto.edu/ hinton/coursera/lecture6/lec6.pdf

- the vector-Jacobian product (VJP) $\nabla_{\boldsymbol{x}} f = \boldsymbol{v}^T \boldsymbol{J}$ is used in the reverse mode. $\boldsymbol{v}$ is a cotangent vector.
- the Jacobian-vector product (JVP) $\nabla_{\boldsymbol{x}} f = \boldsymbol{J}\boldsymbol{v}$ is used in the forward mode. $\boldsymbol{v}$ is a tangent vector.

This $\boldsymbol{v}$ will be introduced as the upstream in the reverse mode of AD in section 7.2.2.

Chapter 5

Basics of parametrized quantum circuits

This thesis uses the **quantum circuit model** to describe the computing scheme, specifically, the photonic quantum circuit model.

In this circuit model, each wire represents an optical mode and each gate corresponds to an optical component, which could reach a universal and fault-tolerant computing [90]. All components in this kind of circuit are parameterized, which is called **parameterized quantum circuit (PQC)** (or variational quantum circuit).

The most famous result of PQC is the variational quantum eigensolver algorithm (VQE) [91, 92], which is a hybrid algorithm combining a quantum circuit with an optimizer in the classical computer to calculate the ground state of the molecular. It is processed as follows: measuring results from the quantum processor and transmitting, examining the cost function and calculating gradients of the parameters, updating the parameters in the quantum processor and starting the processor, and measuring again until the cost function converges. This work paves the way to consider the quantum processor as a learning model and to combine the idea of the cost function and optimization in machine learning.

This parameterized way to make quantum circuits and to serve with different quantum problems starts a new research direction: such as the programmable quantum processor [93, 94, 95], the programmable quantum computer [5, 6], or the programmable quantum simulators [96]. Even though the realization of quantum circuits is different for them (as I mentioned before, using photons, ultra-atoms, superconductors, etc.) and their names are different as well, they have the same core inside, which is the programmability, where the device is parameterized.

5.1 Continuous variable model with parametrized quantum circuits

Unlike the quantum computing models using "qubits", the CV models use the observables, referring to the model in the platform for optical quantum computing. The results of the CV models can be extended to other models of quantum computing [97].

The general optimization scheme is provided in this section, which consists of the parametrized quantum circuits, the cost function, and the gradient descent algorithm to update the parameters. Our model contains Gaussian and non-Gaussian effects [98]. Since Gaussian unitaries provide Gaussian effects, we are going to talk about two optimization schemes to realize the non-Gaussian effects:

1. optimization scheme with Gaussian and non-Gaussian gates;

2. optimization scheme with Gaussian gates and measurements.

5.1.1 General optimization scheme

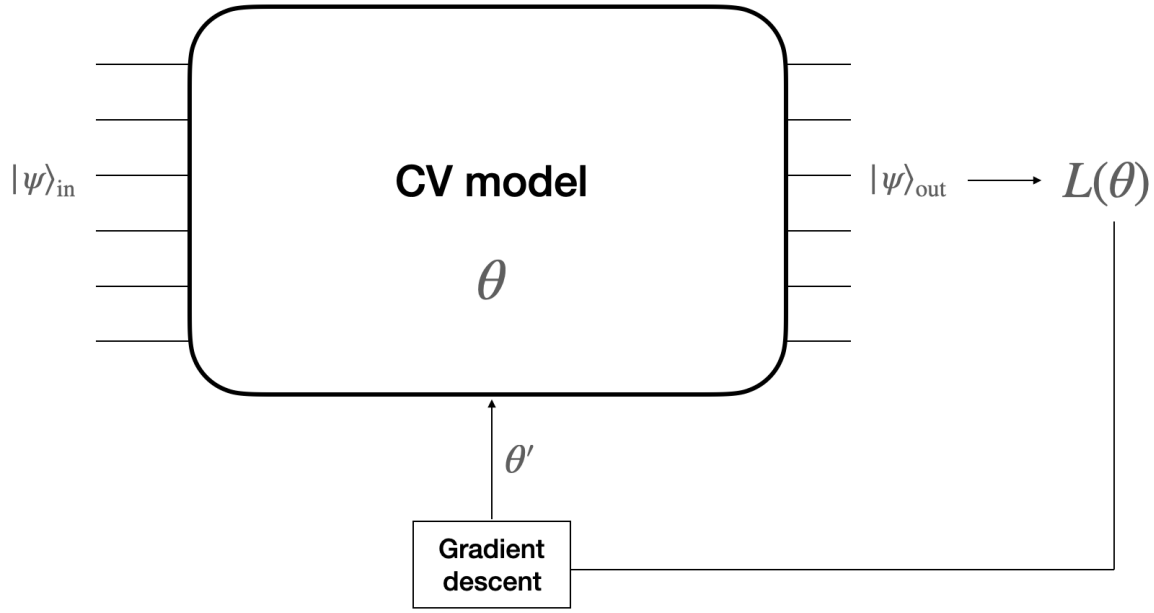


Figure 5.1: General optimization scheme with CV model.

In general, optimization refers to picking up the best elements regarding some criteria. A real function $L(\boldsymbol{\theta})$ is defined at the beginning with parameters $\boldsymbol{\theta}$. The idea is to minimize this function L by choosing different $\boldsymbol{\theta}$ in a given set Θ and computing the function L until one finds the *best* one locally:

$$\boldsymbol{\theta}_{\text{best}} = \min_{\boldsymbol{\theta}\in\Theta} L(\boldsymbol{\theta}). \tag{5.1.1}$$

This kind of optimization is local because the search space of the parameters $\boldsymbol{\theta}$ is in a given set. In contrast, global optimization focuses on finding the path to the best possible solution in the entire search space.

The general optimization scheme is shown in Fig. 5.1. We have a parametrized circuit model $\boldsymbol{\theta}$, the output of the circuit would be:

$$|\psi\rangle_{\text{out}} = U(\boldsymbol{\theta})|\psi\rangle_{\text{in}}. \tag{5.1.2}$$

The cost function L is a function of this output state, and it can be written as $L(\boldsymbol{\theta})$. The optimization in this circuit is to update parameters $\boldsymbol{\theta}$ by calculating the gradients $\frac{\partial L}{\partial \boldsymbol{\theta}}$ and updating the parameters $\boldsymbol{\theta}'$.

5.1.2 Non-Gaussian effects

Non-Gaussian operators

It is natural to obtain non-Gaussian effects from non-Gaussian operators. Typically, the cubic phase gate and the Kerr gate are used as the non-Gaussian ones, defined in Eq. (3.62) and Eq. (3.63), respectively.

For example, the CV neural network proposed in the paper [97] consists of Gaussian and non-Gaussian gates to mimic the affine transformation and nonlinear activation functions in neural networks. An interferometer, local squeezing gates, a second interferometer, local displacement gates, and local non-Gaussian

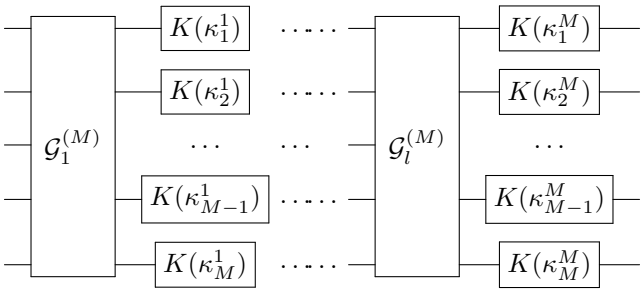


Figure 5.2: M-mode l-Layered architecture (the circuit is composed of Gaussian and non-Gaussian gates, and states get transformed from right to left).

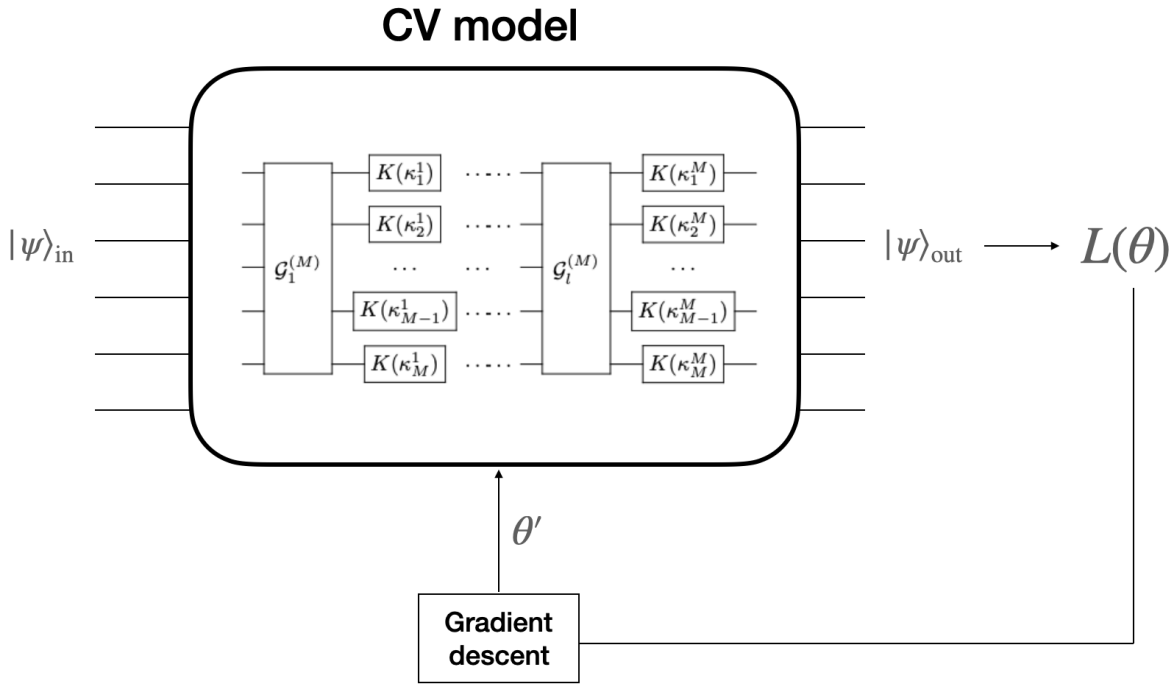


Figure 5.3: Optimization scheme with the circuit of Fig. 5.2.

gates construct one layer. And then, one can concatenate several layers to make a multi-layer CV quantum neural network, shown in Fig. 5.2.

The circuit shown in Fig. 5.3 consists of Gaussian and non-Gaussian gates in each layer. In this scheme, the non-Gaussian operator is easy to manipulate in numerical experiments but challenging to implement in the laboratory.

Measurement

The non-Gaussian effect can be introduced by measurement.

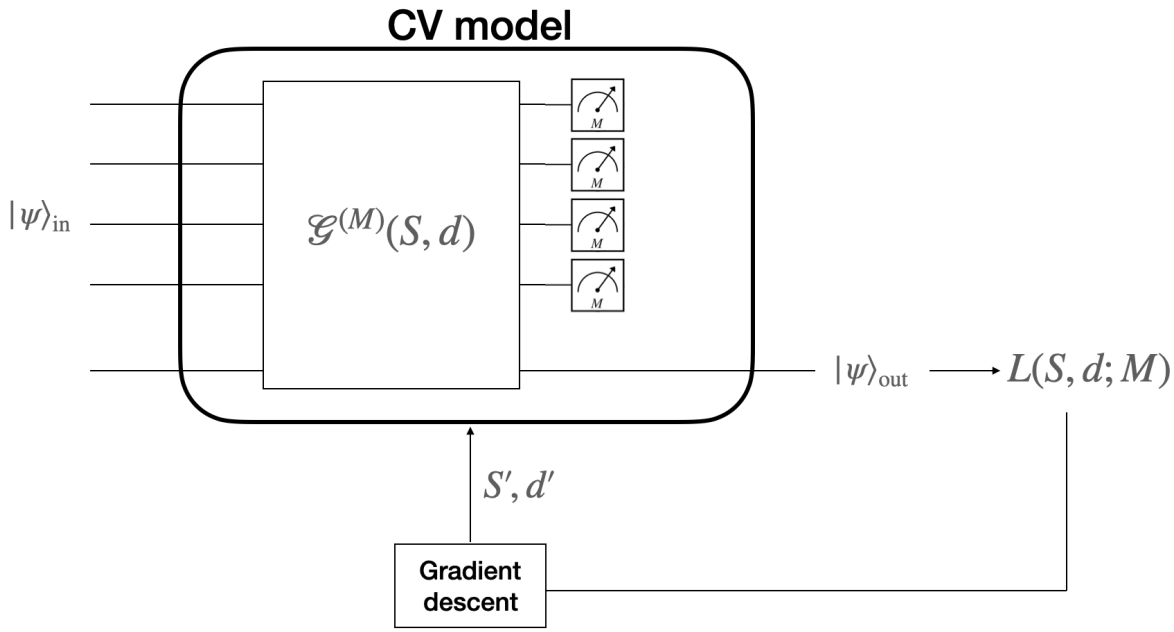


Figure 5.4: Optimization scheme with Gaussian gates and measurements.

It is difficult for photons to interact with each other, but it is needed if we want to reach universal quantum computing. The famous Hong-Ou-Mandel effect [99, 100] gives rise to the *photon bunching* (the incoming photons pair off together), which lies at the heart of linear optical quantum computing. However, the paper [101] claims that this is not enough to make deterministic linear optical quantum computing possible. The famous KLM scheme [102] induces the interaction between photons by making projective measurements with photon detectors, which gives non-deterministic quantum computation.

The circuit shown in Fig. 5.4 involves a multimode Gaussian state and measures all modes using photon-number-resolving (PNR) detectors except one that gives the desired state[103, 104]. The measurement induces a non-Gaussian effect to prepare the desired target state, conditioned on observing a particular measurement pattern. Such an output state is also known as "heralded" by the measurement outcome.

5.2 PQC optimization steps

The PQCs have an initial input state, a sequence of quantum gates that depend on free parameters, and the measurements at the end of the circuit. Someone also calls it a programmable quantum processor.

The PQCs (as shown in Fig. 5.1) consist of four steps:

- Prepare the initial state $|\psi\rangle_{\text{in}}$;

- Apply the quantum circuit (the corresponding of a unitary operator), parameterized by free parameters $\boldsymbol{\theta}$;
- Calculate the cost function $L(\boldsymbol{\theta})$ with the output state $|\psi\rangle_{\text{out}}$;
- Update the parameters $\boldsymbol{\theta}'$ until the cost function converges.

5.3 PQC optimization element

In this section, we explain the building blocks for PQCs: a cost function, a parameterized circuit, gradient calculation, and an optimizer to update the parameters.

5.3.1 Cost function

One important part of PQC is encoding the problem into a suitable cost function. The cost function defines the landscape of all the trainable parameters, and the optimizer navigates through it to find the global minimum.

Our general cost function $L(\boldsymbol{\theta})$ can be expressed as a function of all the elements inside the PQC.

On the one hand, if the cost function is related to the input states $\{\rho\}$, the parameters $\boldsymbol{\theta}$, and observable $\{O\}$, it can be expressed as [105]

$$L(\boldsymbol{\theta}) = f\left(\text{Tr}[OU(\boldsymbol{\theta})\rho U^{\dagger}(\boldsymbol{\theta})]\right), \tag{5.3.1}$$

for some cost functions f. This kind of cost function has been used intensively in lots of variational quantum algorithms, where the cost function depends on the measurement of the observables.

On the other hand, the cost function can be related to the output state directly:

$$L(\boldsymbol{\theta}) = f(U(\boldsymbol{\theta})\rho_{\text{in}}U^{\dagger}(\boldsymbol{\theta})). \tag{5.3.2}$$

In the case of the pure state, we have the following:

$$L(\boldsymbol{\theta}) = f(U(\boldsymbol{\theta})|\psi_{\text{in}}\rangle). \tag{5.3.3}$$

5.3.2 Parameterized circuits

These gates with free parameters sometimes are called ansatz. The specific structure of the ansatz depends on what is the purpose, such as the hardware-efficient ansatz aiming at reducing the circuit depth needed, the well-used problem-inspired unitary coupled clustered ansatz for quantum chemistry problems, or the hybrid ansatz.

In our work, we will not call these gates ansatz. Instead, we will use some specific structure depending on the character of quantum optical elements. The discussion of the structure is shown in section 5.1.

5.3.3 Gradient

Once we have defined the cost function and the circuit, the training of the circuit is to minimize the cost function by updating the parameters $\boldsymbol{\theta}$. The gradient of the cost function is calculated and applied in the optimization part for each parameter.

A hardware-friendly protocol is proposed in the paper [106] to evaluate the partial derivative of L with respect to $\boldsymbol{\theta}$: parameter-shift rule. And the paper [46] extended it. The main idea of the parameter-shift rule is that the partial derivative of a transformation can be calculated as a linear combination of the same transformation but with different parameters.

Another straightforward way to get the gradient of a transformation is to make the transformation differentiable. That is why our method explained in section 6.2 is a breakthrough for the simulation of the photonic quantum circuits: we can simulate and differentiate our transformations (shown in section 7.2.3) and get its gradient directly (shown in section 7.2.1).

5.3.4 Optimizer

The efficiency and reliability of the optimization method used in the PQC play an important role. Because the optimization is normally processed in the classical optimizer, we can try to adapt the optimizer used in the classical machine learning or deep learning task.

The most common optimization is to update the parameters by their gradient in the steepest direction, which corresponds the update rule shown in section 4.5.

Adam is a ubiquitous gradient-descent optimization method that has the ability to select a step size dynamically and has a good performance in a great many deep-learning applications.

Natural gradient (NG) has long been established as an approach to learning tasks that outperforms vanilla gradient descent at the expense of needing a matrix inversion depending on the underlying geometry parameter landscape[39, 40, 41]. We have also adapted the NG into the complex version in section 7.4.

Simultaneous perturbation stochastic approximation (SPSA) [107] is another approach. SPSA can be considered as an approximation to gradient descent along a randomly chosen direction. SPSA has thus been put forward as an efficient method as it avoids computing many gradient components at each iteration.

In addition, it has been shown that one can encounter new challenges when optimizing the PQC: such as the presence of "barren plateaus", where the variance of the gradients vanish exponentially with the increasing number of qubits [108], specific cost functions [109], entanglement [110] and noise [111].

5.4 PQC application

PQC provides the framework to tackle different tasks:

- Universal quantum computing. The *universal* means the ability to achieve any desired unitary. We can build any unitary from a set of basic gates. For a qubit-based circuit, one example of the gate sets consists of Clifford gates and non-Clifford gates [112], while for a CV circuit, one example of the gate sets consists of Gaussian gates and non-Gaussian gates[97].

- Finding ground and excited states in quantum chemistry. The Variational Quantum Eigensolver (VQE) is a flagship algorithm in quantum chemistry using near-term quantum computers to calculate the ground state [20] and excited states of molecules [113].

- Optimization problem. The Quantum Approximate Optimization Algorithm (QAOA) [21] is a method for combinatorial optimization problems, such as Max-Cut problems [21, 114], where the evolution inside the circuit is governed under an engineered Hamiltonian.

- Mathematical applications. PQCs are widely used in different mathematical problems such as solving linear [115, 116], or non-linear systems [117], and the famous Shor's algorithm for factoring [118].

- Machine Learning techniques. Some CV quantum neural networks are considered as the quantum machine learning [97], which refers to the tasks of using a quantum computer to learn patterns as classifiers [119, 120, 97], or of comprising quantum data into a quantum auto-encoder [121, 97]. It can also be used in deep reinforcement learning [122]. Generative adversarial networks (GANs) play an important role in classical machine learning to generate new data with the same distribution as the training set, and a quantum version of GANs is proposed using the PQCs as the neural networks [123].

- Quantum devices design. A Quantum device can be defined as a device that runs depending essentially on quantum mechanical effects. PQCs are also good candidates to design any quantum device. For example, one can implement the Quantum RAM [124] for generative applications. Moreover, the PQC-based quantum repeater (will be introduced in section 10.2.1) is also a good application. However, the practical implementation of PQC-based devices is still limited because of noise and errors.

- Quantum Tomography. Quantum Tomography is the process of reconstructing a quantum state using measurements on an ensemble of identical quantum states [125].

Part II

Automated design of photonic quantum circuits

Chapter 6

Converting Gaussian objects from phase space representation to Fock representation

6.1 Introduction

We introduce a unified and differentiable Fock space representation of Gaussian objects, namely, pure and mixed states, unitaries, and channels in terms of a *single* linear recurrence relation that can generate their Fock space amplitudes recursively. Here the Fock space amplitude refers to the elements of the matrix representation of the Gaussian objects.

Talking about Gaussian objects, the first easy-manipulated representation that comes into mind is the phase space representation. For example, a Gaussian state is characterized by its mean vector and the covariance matrix defined in phase space. On the other hand, its density matrix also has complete information: its characteristic function, then, is defined by tracing the product of the density matrix and the displacement operator, and the Wigner function is defined as the complex Fourier transform of the characteristic function.

In the previous work, the Fock representation of Gaussian objects has not been investigated intensively for a long time because the phase space representation is easier and more comfortable to deal with: the Fock representation is based on the tensor product of infinite-dimensional Hilbert space with the Fock basis, while the phase space representation is finite-dimensional with the coordinates of position and momentum variables. However, because we are working in the bosonic system, we need to use the Fock representation, such as the PNR detector that is widely used to introduce the non-Gaussian effects inside the system.

That is why we developed this recurrence relation for all Gaussian objects in this chapter, from its phase space representation to its Fock space representation. As we explained before, we consider the quantum state evolution in the circuit as the matrix-vector multiplication. However, because of the existence of the cutoff (we can not simulate the infinite-dimensional Hilbert space), there are imperfections in the numerical simulation. Our method can reduce these imperfections by treating the Gaussian operator as a whole instead of decomposing it into fundamental components and calculating their evolution successively.

In the end, we also find the composition rule of Gaussian operators expressed in the recurrent form, which allows us to obtain the correct global phase when composing Gaussian operators. Moreover, it extends our model to states that can be written as linear combinations of Gaussians.

6.2 The recurrence relation for Gaussian objects in Fock representation

In this section, the main problem is to give the way to write the Fock representation for Gaussian objects, which includes M-mode Gaussian pure states, mixed states, unitaries, and channels.

6.2.1 The general formula

We can write M-mode pure states, mixed states, unitaries, and channels in the Fock representation as

$$|\psi\rangle = \sum_{\boldsymbol{k}} \psi_{\boldsymbol{k}} |\boldsymbol{k}\rangle, \tag{6.2.1}$$

$$\rho = \sum_{\boldsymbol{j},\boldsymbol{k}} \rho_{\boldsymbol{j},\boldsymbol{k}} |\boldsymbol{j}\rangle\langle\boldsymbol{k}|, \tag{6.2.2}$$

$$U = \sum_{\boldsymbol{j},\boldsymbol{k}} U_{\boldsymbol{j},\boldsymbol{k}} |\boldsymbol{j}\rangle\langle\boldsymbol{k}|, \tag{6.2.3}$$

$$\Phi[|\boldsymbol{j}\rangle\langle\boldsymbol{l}|] = \sum_{\boldsymbol{i},\boldsymbol{k}} \Phi_{\boldsymbol{k},\boldsymbol{l},\boldsymbol{i},\boldsymbol{j}} |\boldsymbol{i}\rangle\langle\boldsymbol{k}|, \tag{6.2.4}$$

where the Fock representation indices are expressed as a multi-index $\boldsymbol{k} = (k_1, k_2, ..., k_M)$. We now simplify the notation by considering the collections of amplitudes $\psi_{\boldsymbol{k}}$, $\rho_{\boldsymbol{j},\boldsymbol{k}}$, $U_{\boldsymbol{j},\boldsymbol{k}}$ and $\Phi_{\boldsymbol{i},\boldsymbol{j},\boldsymbol{k},\boldsymbol{l}}$ as instances of a tensor $\mathcal{G}_{\boldsymbol{k}}$ where $\boldsymbol{k}$ is M-dimensional index vector for pure states, $2M$-dimensional index vector for mixed states and unitary transformations, and $4M$-dimensional index vector for channels.

One way to produce the Fock space amplitudes of a Gaussian object is to start from a generating function $\Gamma(\boldsymbol{\alpha})$ and then compute its derivatives. The generating function $\Gamma(\boldsymbol{\alpha})$ is also known as the stellar function [126] or the Bargmann function [75]. To obtain the generating function, one needs to contract each index of a Gaussian object with a *rescaled* multi-mode coherent state

$$e^{\frac{1}{2}||\boldsymbol{\alpha}||^2} |\boldsymbol{\alpha}\rangle. \tag{6.2.5}$$

For example, for a pure state, we have

$$\Gamma_\psi(\boldsymbol{\alpha}) = e^{\frac{1}{2}||\boldsymbol{\alpha}||^2} \sum_{\boldsymbol{k}} \psi_{\boldsymbol{k}} \langle\boldsymbol{\alpha}^*|\boldsymbol{k}\rangle = \sum_{\boldsymbol{k}} \psi_{\boldsymbol{k}} \frac{\boldsymbol{\alpha}^{\boldsymbol{k}}}{\sqrt{\boldsymbol{k}!}}, \tag{6.2.6}$$

$$= c_\psi \exp\left(\boldsymbol{\alpha}^T \boldsymbol{b}_\psi + \frac{1}{2}\boldsymbol{\alpha}^T \boldsymbol{A}_\psi \boldsymbol{\alpha}\right), \tag{6.2.7}$$

where $\boldsymbol{A}_\psi$ is an $M \times M$ complex symmetric matrix, $\boldsymbol{b}_\psi$ is an M-dimensional complex vector and c_ψ is the vacuum amplitude.

For unitaries, $\boldsymbol{A}_U$ and $\boldsymbol{b}_U$ are of size $2M \times 2M$ and $2M$. We can recall a previous work in the paper [35], the generating function for the matrix elements of the unitary is defined as

$$\Gamma_U(\boldsymbol{\alpha}, \boldsymbol{\beta}) = e^{\frac{1}{2}\left(||\boldsymbol{\alpha}||^2 + ||\boldsymbol{\beta}||^2\right)} \langle\boldsymbol{\alpha}^*|U|\boldsymbol{\beta}\rangle, \tag{6.2.8}$$

$$= C \exp\left(\boldsymbol{\mu}^T \boldsymbol{\nu} - \frac{1}{2}\boldsymbol{\nu}^T \boldsymbol{\Sigma} \boldsymbol{\nu}\right). \tag{6.2.9}$$

In the following chapter, we will compare our results $\boldsymbol{A}_U$ and $\boldsymbol{b}_U$ with $\boldsymbol{\Sigma}$ and $\boldsymbol{\mu}$. But it is clear that, for unitaries, the generating function is in exponential form and the exponent is a polynomial of degree 2.

In the case of density matrices, we obtain an analogous exponential as in Eq. (6.2.6), except that $\boldsymbol{A}_\rho$ and $\boldsymbol{b}_\rho$ are of size $2M \times 2M$ and $2M$ respectively. And for channels $\boldsymbol{A}_\Phi$ and $\boldsymbol{b}_\Phi$ are of size $4M \times 4M$ and $4M$, respectively.

Therefore, all Gaussian objects are characterized by a complex symmetric matrix $\boldsymbol{A}$, a complex vector $\boldsymbol{b}$ and a complex scalar $c = \mathcal{G}_{\boldsymbol{0}}$, or conversely given valid $\boldsymbol{A}$ and $\boldsymbol{b}$ and c, we can calculate the coefficients $\mathcal{G}_{\boldsymbol{k}}$ by computing derivatives of the appropriate order of the generating function $\Gamma(\boldsymbol{\alpha})$:

$$\mathcal{G}_{\boldsymbol{k}} = c\frac{\boldsymbol{\partial}_{\boldsymbol{\alpha}}^{k}}{\sqrt{\boldsymbol{k}!}} \exp\left(\boldsymbol{\alpha}^T\boldsymbol{b} + \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{A}\boldsymbol{\alpha}\right)\Big|_{\boldsymbol{\alpha}=\boldsymbol{0}}. \tag{6.2.10}$$

In this way, we unify the calculation of the amplitudes of Gaussian objects into a single method that works in all cases, depending on which triple $(\boldsymbol{A}, \boldsymbol{b}, c)$ one is considering.

Multivariate derivatives of the exponential function can be computed with a linear recurrence formula [35], and in case the function is a polynomial of degree D, the recurrence relation has order D. In our case, the polynomial has degree 2, which means we can write a linear recurrence relation of order 2 between the Fock space amplitudes:

$$\mathcal{G}_{\boldsymbol{k}+1_i} = \frac{1}{\sqrt{k_i+1}}\left(b_i\mathcal{G}_{\boldsymbol{k}} + \sum_j \sqrt{k_j}A_{ij}\mathcal{G}_{\boldsymbol{k}-1_j}\right), \tag{6.2.11}$$

with the vacuum amplitude initialized as $\mathcal{G}_{\boldsymbol{0}} = c$.

In this recurrence relation, $\boldsymbol{k}+1_i$ is like $\boldsymbol{k}$ but the i-th index has been increased by 1 (and similarly for $\boldsymbol{k}-1_j$, where it is decreased by 1). If we define $w = \sum_i k_i$ as the weight of the index, the recurrence relation allows us to write amplitudes of weight $w+1$ as linear combinations of amplitudes of weight w and $w-1$. By applying it repeatedly, one can reach any Fock space amplitude (in practice, one eventually reaches a numerical precision horizon [127]). More details will be discussed in chapter 8.

This is clear that in our recurrence relation formula, it is important to find out the three entities: a complex symmetric matrix $\boldsymbol{A}$, a complex vector $\boldsymbol{b}$ and a complex scalar $c = \mathcal{G}_{\boldsymbol{0}}$. In the following sections, we are going to give the $\boldsymbol{A}, \boldsymbol{b}$, and c for each Gaussian object.

To obtain the results ($\boldsymbol{A}_X, \boldsymbol{b}_X$, where $X = \{\psi, \rho, U, \Phi\}$), our work develops in two ways:

- The first one is the **extract** way: we can get $\boldsymbol{A}_\rho$ for mixed states and in the case of pure states, $\boldsymbol{A}_\rho$ becomes to a form of a diagonal block, and we can extract the block of $\boldsymbol{A}_\psi$ from $\boldsymbol{A}_\rho$. For transformations, using the Choi-Jamiołkowski duality, we can treat channels as mixed states and get $\boldsymbol{A}_\Psi$, and in case of a unitary channel, one obtains $\boldsymbol{A}_U$ of Gaussian unitaries by extracting the block of $\boldsymbol{A}_\Psi$.
- The second one is only related to the Gaussian unitaries: we can get $\boldsymbol{A}_U$ directly from its Fock-Bargmann representation.

Note that we will also compare our generalized results with the previous work of Gaussian unitaries $\boldsymbol{A}_U$ in [35], and they correspond well with each other.

6.2.2 Multidimensional Hermite Polynomials

We first recall the definition of the multidimensional Hermite polynomials as the Taylor series of a multidimensional Gaussian function

$$K^{\boldsymbol{A}}(\boldsymbol{y}, \boldsymbol{b}) = \exp\left(\boldsymbol{y}^T\boldsymbol{b} + \tfrac{1}{2}\boldsymbol{y}^T\boldsymbol{A}\boldsymbol{y}\right) = \sum_{\boldsymbol{k}\geq\boldsymbol{0}} \frac{G_{\boldsymbol{k}}^{\boldsymbol{A}}(\boldsymbol{b})}{\boldsymbol{k}!}\boldsymbol{y}^{\boldsymbol{k}}. \tag{6.2.12}$$

Note the sign of the quadratic term in the exponential, which can differ from other conventions. In the last equation $\boldsymbol{b} \in \mathbb{C}^\ell$ is a complex vector, $\boldsymbol{A} = \boldsymbol{A}^T \in \mathbb{C}^{\ell\times\ell}$ is a complex symmetric matrix and $\boldsymbol{k} \in \mathbb{Z}_0^\ell$ is a vector of non-negative integers. This notation makes it explicit that

$$\left[\prod_{i=1}^{\ell}\left(\frac{\partial}{\partial y_i}\right)^{k_i}\right] K^{\boldsymbol{A}}(\boldsymbol{y}, \boldsymbol{b})\Big|_{\boldsymbol{y}=\boldsymbol{0}} = G_{\boldsymbol{k}}^{\boldsymbol{A}}(\boldsymbol{b}). \tag{6.2.13}$$

These polynomials satisfy the recurrence relation

$$G^{\boldsymbol{A}}_{\boldsymbol{k}+1_i}(\boldsymbol{b}) - b_i G^{\boldsymbol{A}}_{\boldsymbol{k}}(\boldsymbol{b}) - \sum_{j=1}^{M} A_{i,j} p_j G^{\boldsymbol{A}}_{\boldsymbol{k}-1_j}(\boldsymbol{b}) = 0, \tag{6.2.14}$$

where 1_i is a vector that has a 1 in the i-th entry and 0s elsewhere. Note that $G^{\boldsymbol{A}}_{\boldsymbol{0}}(\boldsymbol{b}) = 1$, $G^{\boldsymbol{A}}_{1_i}(\boldsymbol{b}) = b_i$ and that $G^{\boldsymbol{A}}_{1_i+1_j}(\boldsymbol{b}) = b_i b_j + A_{ij}$.

The multidimensional Hermite polynomial is related to the loop-hafnian function introduced in Ref. [128], which counts the number of perfect matching of weighted graphs, including self-loops. Hence, our method can be converted into the calculation of the loop-hafnian function in the graph.

They are related as follows

$$G^{\boldsymbol{A}}_{\boldsymbol{k}}(\boldsymbol{b}) = \mathrm{lhaf}(\mathrm{fdiag}(\boldsymbol{A}_{\boldsymbol{k}}, \boldsymbol{b}_{\boldsymbol{k}})), \tag{6.2.15}$$

where fdiag fills the diagonal of the matrix in the first argument using the vector in the second argument. Note that $\boldsymbol{A}_{\boldsymbol{k}}$ is the matrix obtained from $\boldsymbol{A}$ by repeating its i-th row and column k_i times. Similarly, $\boldsymbol{b}_{\boldsymbol{k}}$ is the vector obtained from $\boldsymbol{b}$ by repeating its i-th entry k_i times. Note that when $k_i = 0$ the relevant row and column of $\boldsymbol{A}$ and entry of $\boldsymbol{b}$ are deleted. The best known methods to calculate the single loop-hafnian in Eq. (6.2.15) requires $O(C^3\sqrt{\prod_{i=1}^{\ell}(1+k_i)})$ steps where C is the number of nonzero entries in the vector $\boldsymbol{k}$ [129].

We will show below that the Fock representation of a pure Gaussian state, a mixed Gaussian state, a Gaussian unitary, or a Gaussian channel can all be written as

$$c \times \frac{G^{\boldsymbol{A}}_{\boldsymbol{k}}(\boldsymbol{b})}{\sqrt{\boldsymbol{k}!}}, \tag{6.2.16}$$

where c is a scalar, $\boldsymbol{b}$ is a vector of dimension ℓ, $\boldsymbol{A}$ is square matrix of size $\ell \times \ell$ and $\boldsymbol{k} \in \mathbb{Z}^{\ell}_{\geq 0}$. The integer ℓ equals $M, 2M, 2M, 4M$ for pure states, mixed states, unitaries or channels on M modes respectively.

Note that the quantity in Eq. (6.2.16) is potentially the ratio of two large numbers. In particular, since this quantity represents a probability or a probability amplitude, it should be bounded in absolute value by 1. Thus it is often convenient, especially for numerical purposes, to introduce renormalized multidimensional Hermite polynomials as

$$\mathcal{G}^{\boldsymbol{A}}_{\boldsymbol{k}}(\boldsymbol{b}) = c \times \frac{G^{\boldsymbol{A}}_{\boldsymbol{k}}(\boldsymbol{b})}{\sqrt{\boldsymbol{k}!}}, \tag{6.2.17}$$

which satisfy the recurrence relation in Eq. (6.2.11).

6.2.3 States

In this subsection, we show how to turn the symplectic representation of a Gaussian state into the Fock space representation of the same object. This follows the developments in Refs. [130, 131, 8, 132, 133, 134, 135]. Another work [136] has also developed similar results for Gaussian states from the point of view of Hermite-like polynomials to describe them.

To compute the Fock space amplitudes of a Gaussian pure state we need the triple $(\boldsymbol{A}_\psi, \boldsymbol{b}_\psi, c_\psi)$ where $\boldsymbol{A}_\psi$ and $\boldsymbol{b}_\psi$ are M-dimensional. If the state is mixed, we need the triple $(\boldsymbol{A}_\rho, \boldsymbol{b}_\rho, c_\rho)$ where $\boldsymbol{A}_\rho$ and $\boldsymbol{b}_\rho$ are $2M$-dimensional. We are now going to show we use the **extract** method to obtain these triples for mixed states first and then transfer them to triples for the pure states.

Mixed state

In this part, the mixed state ρ is characterized by the complex covariance matrix $\boldsymbol{\sigma}$ and the mean vector $\bar{\boldsymbol{\mu}}$.

As the complex covariance matrix $\boldsymbol{\sigma}$ introduced in Eq.(3.33), it is convenient to introduce the $s-$parametrized complex covariance matrix

$$\boldsymbol{\sigma}_s = \boldsymbol{\sigma} + \frac{s}{2}\mathbb{1}_{2M}, \tag{6.2.18}$$

by definition $\boldsymbol{\sigma}_0 \equiv \boldsymbol{\sigma}$ and moreover we use the shorthand notation $\boldsymbol{\sigma}_\pm \equiv \boldsymbol{\sigma}_{\pm 1}$. $\boldsymbol{\sigma}_+$ is also called the Husimi covariance matrix.

We recall the results derived in Ref. [134]. An expression for the Fock representation of the Gaussian state is

$$\langle \boldsymbol{m}|\rho|\boldsymbol{n}\rangle = c_\rho \times \prod_{s=1}^{M} \frac{\partial_{\alpha_s}^{n_s}\partial_{\alpha_s^*}^{m_s}}{\sqrt{n_s! m_s!}} \exp\left[\tfrac{1}{2}\boldsymbol{y}^T \boldsymbol{A}_\rho \boldsymbol{y} + \boldsymbol{y}^T \boldsymbol{b}_\rho\right], \tag{6.2.19}$$

where, relative to Eq. (6.2.13), we identified $\boldsymbol{y} = \left[\begin{smallmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\alpha}^*\end{smallmatrix}\right]$, $\boldsymbol{k} = \boldsymbol{n} \oplus \boldsymbol{m}$, $\ell = 2M$ and used the results from Refs. [8, 132, 135] to write together with the definitions in Eqs. (3.32)

$$\boldsymbol{A}_\rho = \boldsymbol{P}_M\left[\mathbb{1} - \boldsymbol{\sigma}_+^{-1}\right] = \boldsymbol{P}_M\boldsymbol{\sigma}_-\boldsymbol{\sigma}_+^{-1} = \boldsymbol{P}_M\boldsymbol{\sigma}_+^{-1}\boldsymbol{\sigma}_-, \tag{6.2.20}$$

$$\boldsymbol{b}_\rho = \left[\boldsymbol{\sigma}_+^{-1}\bar{\boldsymbol{\mu}}\right]^*, \tag{6.2.21}$$

$$c_\rho = \langle \boldsymbol{0}|\rho|\boldsymbol{0}\rangle = \frac{\exp\left[-\frac{1}{2}\bar{\boldsymbol{\mu}}^\dagger\boldsymbol{\sigma}_+^{-1}\bar{\boldsymbol{\mu}}\right]}{\sqrt{\det(\boldsymbol{\sigma}_+)}}, \tag{6.2.22}$$

$$\boldsymbol{P}_M = \left[\begin{smallmatrix}0_M & \mathbb{1}_M\\ \mathbb{1}_M & 0_M\end{smallmatrix}\right], \tag{6.2.23}$$

to finally write

$$\langle \boldsymbol{m}|\rho|\boldsymbol{n}\rangle = c_\rho \times \frac{G_{\boldsymbol{n}\oplus\boldsymbol{m}}^{\boldsymbol{A}_\rho}(\boldsymbol{b}_\rho)}{\sqrt{\boldsymbol{n}!\boldsymbol{m}!}}. \tag{6.2.24}$$

Pure state

In the case where $\rho = |\Psi\rangle\langle\Psi|$ is a pure state, it is easy to show that

$$\boldsymbol{A}_\rho = \boldsymbol{A}_\psi^* \oplus \boldsymbol{A}_\psi, \tag{6.2.25}$$

$$\boldsymbol{b}_\rho = \boldsymbol{b}_\psi^* \oplus \boldsymbol{b}_\psi, \tag{6.2.26}$$

and then we can write

$$G_{\boldsymbol{n}\oplus\boldsymbol{m}}^{\boldsymbol{A}_\rho}(\boldsymbol{b}_\rho) = G_{\boldsymbol{n}\oplus\boldsymbol{m}}^{\boldsymbol{A}_\psi^*\oplus\boldsymbol{A}_\psi}(\boldsymbol{b}_\psi^* \oplus \boldsymbol{b}_\psi) \tag{6.2.27}$$

$$= G_{\boldsymbol{n}}^{\boldsymbol{A}_\psi^*}(\boldsymbol{b}_\psi^*) \times G_{\boldsymbol{m}}^{\boldsymbol{A}_\psi}(\boldsymbol{b}_\psi) \tag{6.2.28}$$

$$= [G_{\boldsymbol{n}}^{\boldsymbol{A}_\psi}(\boldsymbol{b}_\psi)]^* \times G_{\boldsymbol{m}}^{\boldsymbol{A}_\psi}(\boldsymbol{b}_\psi), \tag{6.2.29}$$

which allows us to write the probability amplitude of a pure state

$$\langle \boldsymbol{m}|\Psi\rangle = c_\psi e^{i\varphi_\Psi}\frac{G_{\boldsymbol{m}}^{\boldsymbol{A}_\psi}(\boldsymbol{b}_\psi)}{\sqrt{\boldsymbol{m}!}}, \quad c_\psi = c_\rho^* c_\rho, \tag{6.2.30}$$

up to a global phase φ that cannot be determined from the covariance matrix and vector of means of the pure Gaussian state. This will be discussed in section 6.3.

Examples

We now give a few examples.

The recursive representation of a single-mode coherent state of amplitude α is given by $\boldsymbol{A}_\psi = 0$, $\boldsymbol{b}_\psi = \alpha$ and $c_\psi = \psi_0 = e^{-\frac{1}{2}|\alpha|^2}$:

$$\psi_{k+1}^{\text{coh}} = \frac{1}{\sqrt{k+1}} \alpha \, \psi_k^{\text{coh}}. \tag{6.2.31}$$

A squeezed state with squeezing parameter r and angle ϕ is given by $\boldsymbol{A}_\psi = \tanh(r) e^{i\phi}$, $\boldsymbol{b}_\psi = 0$ and $c_\psi = \sqrt{\text{sech}(r)}$:

$$\psi_{k+1}^{\text{sq}} = \sqrt{\frac{k}{k+1}} \tanh(r) e^{i\phi} \psi_{k-1}^{\text{sq}}. \tag{6.2.32}$$

Note that, as expected, this recurrence relation skips odd indices. A displaced squeezed state (which is the most general pure single-mode Gaussian state) is given by $\boldsymbol{A}_\psi = \tanh(r) e^{i\phi}$, $\boldsymbol{b}_\psi = \alpha$ and $c_\psi = \sqrt{\text{sech}(r)} e^{-\frac{1}{2}|\alpha|^2}$:

$$\psi_{k+1}^{\text{dsq}} = \frac{1}{\sqrt{k+1}} \left(\alpha \psi_k^{\text{dsq}} + \sqrt{k} \tanh(r) e^{i\phi} \psi_{k-1}^{\text{dsq}} \right). \tag{6.2.33}$$

For the simple case of M squeezed states with parameters r_i sent into an interferometer with unitary $\boldsymbol{U}$ we have that $\boldsymbol{A}_\psi = -\boldsymbol{U} \left[\bigoplus_{i=1}^M \tanh r_i \right] \boldsymbol{U}^T$.

The thermal state is given by $\boldsymbol{A}_\rho = \frac{\bar{n}}{\bar{n}+1} \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$, $\boldsymbol{b}_\rho = \boldsymbol{0}$ and $c = \frac{1}{1+\bar{n}}$, where $\bar{n}$ is the average photon number, giving rise to the recurrence relations:

$$\rho_{k_1+1,k_2}^{\text{th}} = \sqrt{\frac{k_2}{k_1+1}} \frac{\bar{n}}{\bar{n}+1} \rho_{k_1,k_2-1}^{\text{th}}, \tag{6.2.34}$$

$$\rho_{k_1,k_2+1}^{\text{th}} = \sqrt{\frac{k_1}{k_2+1}} \frac{\bar{n}}{\bar{n}+1} \rho_{k_1-1,k_2}^{\text{th}}. \tag{6.2.35}$$

For a squeezed state along the q-quadrature by $r > 0$ (the symplectic matrix $\boldsymbol{S}$ can be found in Eq. (3.46)) that undergoes loss by transmission factor η, we start from the vacuum state with $\boldsymbol{V} = \frac{\hbar}{2}\mathbb{1}$, we apply the squeezing operator $\boldsymbol{V}' = \boldsymbol{S}\boldsymbol{V}\boldsymbol{S}^T$, we make the state pass through the lossy channel $\boldsymbol{V}'' = \boldsymbol{X}\boldsymbol{V}'\boldsymbol{X}^T + \boldsymbol{Y}$, and we obtain its covariance matrix $\boldsymbol{\sigma} = \frac{1}{\hbar}\boldsymbol{W}^\dagger \boldsymbol{V}'' \boldsymbol{W}$. Then it is easy to find $\boldsymbol{A}_\rho$ from Eq. (6.2.20) that

$$\boldsymbol{A}_\rho = \frac{\eta}{\coth^2 r - (\eta-1)^2} \begin{bmatrix} -\coth r & 1-\eta \\ 1-\eta & -\coth r \end{bmatrix}. \tag{6.2.36}$$

In the limit of no loss we find $\boldsymbol{A}_\rho = -[\tanh r \oplus \tanh r]$ while in the limit of zero transmission we retrieve the single-mode vacuum, $\boldsymbol{A}_\rho = \boldsymbol{0}_2$.

6.2.4 Unitaries

The three parameters $\boldsymbol{A}_U$, $\boldsymbol{b}_U$ and c_U of the Gaussian unitaries can be found in two ways: the **extract** method by following the Choi-Jamiolkowski isomorphism in the next section to get the channel parameters first and then extract part of it; or use the Fock-Bargmann method. Here in this section, we will focus on the Fock-Bargmann method.

As we have already shown in section 2.4, in the Fock-Bargmann representation, we are working with the complex parameter $\boldsymbol{z} \in \mathbb{C}^n$.

If we have a symplectic matrix $\boldsymbol{M} \in \text{Sp}$ on $\mathbb{R}^{2n}$, the linear transformation to map it onto the $\mathbb{C}^n$ is:

$$\boldsymbol{M}_c = \boldsymbol{W}^\dagger \boldsymbol{M} \boldsymbol{W}, \tag{6.2.37}$$

where $\boldsymbol{W}$ is defined in Eq.(3.6). And now the $\boldsymbol{M}_c \in \mathrm{Sp}_c$.

If we have a unitary operator $\nu(\boldsymbol{M_c})$ and the $\boldsymbol{M_c} = \begin{bmatrix} \boldsymbol{P} & \boldsymbol{Q} \\ \boldsymbol{Q}^* & \boldsymbol{P}^* \end{bmatrix}$, then

$$K_{\boldsymbol{M}_c}(\boldsymbol{z}, \boldsymbol{w}^*) = C_{\boldsymbol{M}_c} \exp\left\{\frac{1}{2}(\boldsymbol{z}^{\boldsymbol{T}}\boldsymbol{Q}^*\boldsymbol{P}^{-1}\boldsymbol{z} + 2\boldsymbol{w}^\dagger\boldsymbol{P}^{-1}\boldsymbol{z} - \boldsymbol{w}^\dagger\boldsymbol{P}^{-1}\boldsymbol{Q}\boldsymbol{w}^*)\right\}, \tag{6.2.38}$$

where $C_{\boldsymbol{M}_c} = \frac{1}{\sqrt{\det P}}$ and $\boldsymbol{z}, \boldsymbol{w} \in \mathbb{C}^n$.

In addition, there is a special operation β, which can be seen as a corresponding displacement operator in $\mathcal{F}_n$:

$$\beta(\boldsymbol{w})F(\boldsymbol{z}) = e^{-\frac{1}{2}|\boldsymbol{w}|^2 - \boldsymbol{z}\boldsymbol{w}^*}F(\boldsymbol{z} + \boldsymbol{w}). \tag{6.2.39}$$

A M-mode Gaussian unitary generated by a second-degree polynomial in the quadratures can be decomposed into a M-mode displacement $\mathcal{D}_{\boldsymbol{d}}$ and a M-mode unitary so that we can write

$$U_G = \mathcal{D}_{\boldsymbol{d}}U(\boldsymbol{S}), \tag{6.2.40}$$

where $\mathcal{D}_{\boldsymbol{d}}$ is the displacement operator, parametrized by a real vector $\boldsymbol{d}$ of size $2M$. In the same way, we can write it in $\mathcal{F}_n$:

$$\beta(-\boldsymbol{\gamma}^*)\nu(\boldsymbol{M}_c)F(\boldsymbol{z}) = e^{-\frac{1}{2}|\boldsymbol{\gamma}|^2 + \boldsymbol{z}\boldsymbol{\gamma}}TF(\boldsymbol{z} - \boldsymbol{\gamma}^*) \tag{6.2.41}$$

$$= e^{-\frac{1}{2}|\boldsymbol{\gamma}|^2 + \boldsymbol{z}\boldsymbol{\gamma}} \int K_{\boldsymbol{M}}(\boldsymbol{z} - \boldsymbol{\gamma}^*, \boldsymbol{w}^*)F(\boldsymbol{w})e^{-|\boldsymbol{w}|^2}\mathrm{d}\boldsymbol{w} \tag{6.2.42}$$

$$\begin{aligned} &= e^{-\frac{1}{2}|\boldsymbol{\gamma}|^2 + \boldsymbol{z}\boldsymbol{\gamma}} \int \frac{1}{\det \boldsymbol{P}} \exp\frac{1}{2}\Big(\boldsymbol{z}^T\boldsymbol{Q}^*\boldsymbol{P}^{-1}\boldsymbol{z} - \boldsymbol{\gamma}^\dagger\boldsymbol{Q}^*\boldsymbol{P}^{-1}\boldsymbol{z} \\ &\quad - \boldsymbol{z}^T\boldsymbol{Q}^*\boldsymbol{P}^{-1}\boldsymbol{\gamma}^* + \boldsymbol{\gamma}^\dagger\boldsymbol{Q}^*\boldsymbol{P}^{-1}\boldsymbol{\gamma}^* + 2\boldsymbol{w}^\dagger\boldsymbol{P}^{-1}\boldsymbol{z} - 2\boldsymbol{w}^\dagger\boldsymbol{P}^{-1}\boldsymbol{\gamma}^* - \boldsymbol{w}^\dagger\boldsymbol{P}^{-1}\boldsymbol{Q}\boldsymbol{w}^*\Big)\mathrm{d}\boldsymbol{w} \end{aligned} \tag{6.2.43}$$

$$\begin{aligned} &= e^{-\frac{1}{2}|\boldsymbol{\gamma}|^2 + \boldsymbol{z}\boldsymbol{\gamma}} \int \frac{1}{\sqrt{\det \boldsymbol{P}}} \exp\left\{\frac{1}{2}\left(\boldsymbol{\gamma}^\dagger\boldsymbol{Q}^*\boldsymbol{P}^{-1}\boldsymbol{\gamma}^* - 2\boldsymbol{w}^\dagger\boldsymbol{P}^{-1}\boldsymbol{\gamma}^* - \boldsymbol{\gamma}^\dagger\boldsymbol{Q}^*\boldsymbol{P}^{-1}\boldsymbol{z}\right)\right\}\mathrm{d}\boldsymbol{w} \\ &\quad * \exp\left\{\frac{1}{2}\left(\boldsymbol{z}^T\boldsymbol{Q}^*\boldsymbol{P}^{-1}\boldsymbol{z} + 2\boldsymbol{w}^\dagger\boldsymbol{P}^{-1}\boldsymbol{z} - \boldsymbol{w}^\dagger\boldsymbol{P}^{-1}\boldsymbol{Q}\boldsymbol{w}^*\right)\right\} \end{aligned} \tag{6.2.44}$$

$$\begin{aligned} &= \int \frac{\exp\left\{\frac{1}{2}\left(-|\boldsymbol{\gamma}|^2 + \boldsymbol{\gamma}^\dagger\boldsymbol{Q}^*\boldsymbol{P}^{-1}\boldsymbol{\gamma}^*\right)\right\}}{\sqrt{\det \boldsymbol{P}}} \exp\left\{\begin{bmatrix} -\boldsymbol{\gamma}^\dagger\boldsymbol{Q}^*\boldsymbol{P}^{-1} + \boldsymbol{\gamma}^T & -\boldsymbol{\gamma}^\dagger\left(\boldsymbol{P}^{-1}\right)^T \end{bmatrix}\begin{bmatrix} \boldsymbol{z} \\ \boldsymbol{w}^* \end{bmatrix}\right\} \\ &\quad * \exp\left\{\frac{1}{2}\left(\begin{bmatrix} \boldsymbol{z} & \boldsymbol{w}^* \end{bmatrix}^T\begin{bmatrix} \boldsymbol{Q}^*\boldsymbol{P}^{-1} & \left(\boldsymbol{P}^{-1}\right)^T \\ \boldsymbol{P}^{-1} & -\boldsymbol{P}^{-1}\boldsymbol{Q} \end{bmatrix}\begin{bmatrix} \boldsymbol{z} \\ \boldsymbol{w}^* \end{bmatrix}\right)\right\}\mathrm{d}\boldsymbol{w}. \end{aligned} \tag{6.2.45}$$

Since the exponential term has a similar form with Eq.(6.2.10), if we let $\boldsymbol{\nu} = \begin{bmatrix} z \\ w^* \end{bmatrix}$, we get

$$\boldsymbol{A}'_U = \begin{bmatrix} \boldsymbol{Q}^*\boldsymbol{P}^{-1} & \left(\boldsymbol{P}^{-1}\right)^T \\ \boldsymbol{P}^{-1} & -\boldsymbol{P}^{-1}\boldsymbol{Q} \end{bmatrix}, \tag{6.2.46}$$

$$\boldsymbol{b}'_U = \begin{bmatrix} -\boldsymbol{\gamma}^\dagger\boldsymbol{Q}^*\boldsymbol{P}^{-1} + \gamma^T & -\boldsymbol{\gamma}^\dagger\left(\boldsymbol{P}^{-1}\right)^T \end{bmatrix}, \tag{6.2.47}$$

$$c'_U = \frac{\exp\left\{\frac{1}{2}\left(-|\boldsymbol{\gamma}|^2 + \boldsymbol{\gamma}^\dagger\boldsymbol{Q}^*\boldsymbol{P}^{-1}\boldsymbol{\gamma}^*\right)\right\}}{\sqrt{\det P}}, \tag{6.2.48}$$

where $\boldsymbol{P}$ and $\boldsymbol{Q}$ come from the complex matrix $\boldsymbol{M}_c$. Note that, in this paper, $\nu = \begin{bmatrix} z^* \\ w \end{bmatrix}$, so with this method, the final result should be the conjugate of what we have:

$$\boldsymbol{A}'_U = \begin{bmatrix} \boldsymbol{Q}\left(\boldsymbol{P}^{-1}\right)^* & \left(\boldsymbol{P}^{-1}\right)^\dagger \\ \left(\boldsymbol{P}^{-1}\right)^* & -\left(\boldsymbol{P}^{-1}\right)^*\boldsymbol{Q}^* \end{bmatrix}. \tag{6.2.49}$$

Now we have the three entities for Gaussian unitaries. The paper [35] gives us the value of these three entities as well in a special case where the Gaussian unitary is defined as

$$\mathcal{G}(\boldsymbol{\gamma}, \boldsymbol{W}, \boldsymbol{\zeta}, \boldsymbol{V}) = \mathcal{D}(\boldsymbol{\gamma}) U(\boldsymbol{W}) \mathcal{S}(\boldsymbol{\zeta}) U(\boldsymbol{V}). \tag{6.2.50}$$

In [35], C, μ, and Σ are defined as follows:

$$C = \frac{\exp\left(-\frac{1}{2}\left[||\boldsymbol{\gamma}||^2 + \boldsymbol{\gamma}^\dagger \boldsymbol{W} \operatorname{diag}(e^{i\boldsymbol{\delta}} \tanh \boldsymbol{r}) \boldsymbol{W}^T \boldsymbol{\gamma}^*\right]\right)}{\sqrt{\prod_{i=1}^{M} \cosh r_i}}, \tag{6.2.51}$$

$$\boldsymbol{\mu}^T = \left[\boldsymbol{\gamma}^\dagger \boldsymbol{W} \operatorname{diag}(e^{i\boldsymbol{\delta}} \tanh \boldsymbol{r}) \boldsymbol{W}^T + \boldsymbol{\gamma}^T, -\boldsymbol{\gamma}^\dagger \boldsymbol{W} \operatorname{diag}(\operatorname{sech} \boldsymbol{r}) \boldsymbol{V}\right], \tag{6.2.52}$$

$$\boldsymbol{\Sigma} = \left[\begin{array}{c|c} \boldsymbol{W} \operatorname{diag}(e^{i\boldsymbol{\delta}} \tanh \boldsymbol{r}) \boldsymbol{W}^T & -\boldsymbol{W} \operatorname{diag}(\operatorname{sech} \boldsymbol{r}) \boldsymbol{V} \\ \hline -\boldsymbol{V}^T \operatorname{diag}(\operatorname{sech} \boldsymbol{r}) \boldsymbol{W}^{\boldsymbol{T}} & -\boldsymbol{V}^T \operatorname{diag}(e^{-i\boldsymbol{\delta}} \tanh \boldsymbol{r}) \boldsymbol{V} \end{array}\right], \tag{6.2.53}$$

where $\boldsymbol{\gamma}$ is the vector of displacement parameters, $\boldsymbol{\delta}$, and $\boldsymbol{r}$ are the polar coordinates of the complex vector $\boldsymbol{\zeta}$ of squeezing parameters, and $\boldsymbol{W}$ and $\boldsymbol{V}$ are unitary covariance matrices describing the two interferometers.

Now we try to calculate three entities with our general method but using the same decomposition of Gaussian unitaries in Eq. (6.2.50).

We first write the symplectic matrix $\boldsymbol{M}$ of this specific Gaussian unitary (except the displacement):

$$\boldsymbol{M} = \begin{bmatrix} \Re(\boldsymbol{W}) & -\Im(\boldsymbol{W}) \\ \Im(\boldsymbol{W}) & \Re(\boldsymbol{W}) \end{bmatrix} \begin{bmatrix} \operatorname{diag}(\cosh \boldsymbol{r} + \cos \boldsymbol{\theta} \sinh \boldsymbol{r}) & \operatorname{diag}(\sin \boldsymbol{\theta} \sinh \boldsymbol{r}) \\ \operatorname{diag}(\sin \boldsymbol{\theta} \sinh \boldsymbol{r}) & \operatorname{diag}(\cosh \boldsymbol{r} - \cos \boldsymbol{\theta} \sinh \boldsymbol{r}) \end{bmatrix} \begin{bmatrix} \Re(\boldsymbol{V}) & -\Im(\boldsymbol{V}) \\ \Im(\boldsymbol{V}) & \Re(\boldsymbol{V}) \end{bmatrix}, \tag{6.2.54}$$

and we can compute the matrix $\boldsymbol{M}_c$ by sandwiching it with $\boldsymbol{W}^\dagger$ and $\boldsymbol{W}$:

$$\boldsymbol{M}_c = \begin{bmatrix} \boldsymbol{P} & \boldsymbol{Q} \\ \boldsymbol{Q}^* & \boldsymbol{P}^* \end{bmatrix} = \begin{bmatrix} \boldsymbol{W} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{W}^* \end{bmatrix} \begin{bmatrix} \operatorname{diag}(\cosh \boldsymbol{r}) & \operatorname{diag}\left(e^{i\boldsymbol{\delta}} \sinh \boldsymbol{r}\right) \\ \operatorname{diag}(e^{-i\boldsymbol{\delta}} \sinh \boldsymbol{r}) & \operatorname{diag}(\cosh \boldsymbol{r}) \end{bmatrix} \begin{bmatrix} \boldsymbol{V} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{V}^* \end{bmatrix} \tag{6.2.55}$$

$$= \begin{bmatrix} \boldsymbol{W} \operatorname{diag}(\cosh \boldsymbol{r}) \boldsymbol{V} & \boldsymbol{W} \operatorname{diag}(e^{i\boldsymbol{\delta}} \sinh \boldsymbol{r}) \boldsymbol{V}^* \\ \boldsymbol{W}^* \operatorname{diag}(e^{-i\boldsymbol{\delta}} \sinh \boldsymbol{r}) \boldsymbol{V} & \boldsymbol{W}^* \operatorname{diag}(\cosh \boldsymbol{r}) \boldsymbol{V}^* \end{bmatrix}. \tag{6.2.56}$$

Then we can calculate

$$\left(\boldsymbol{P}^{-1}\right)^* = \boldsymbol{V}^T \operatorname{diag}(\operatorname{sech} \boldsymbol{r}) \boldsymbol{W}^T, \tag{6.2.57}$$

$$(\boldsymbol{P}^{-1})^\dagger = \boldsymbol{W} \operatorname{diag}(\operatorname{sech} \boldsymbol{r}) \boldsymbol{V}, \tag{6.2.58}$$

$$\boldsymbol{Q}\left(\boldsymbol{P}^{-1}\right)^* = -\boldsymbol{W} \operatorname{diag}(e^{i\boldsymbol{\delta}} \sinh \boldsymbol{r}) \boldsymbol{V}^* \left(\boldsymbol{V}^T \operatorname{diag}(\operatorname{sech} \boldsymbol{r}) \boldsymbol{W}^T\right) = -\boldsymbol{W} \operatorname{diag}(e^{i\boldsymbol{\delta}} \tanh \boldsymbol{r}) \boldsymbol{W}^T. \tag{6.2.59}$$

We will have

$$\boldsymbol{A}'_U = -\left[\begin{array}{c|c} \boldsymbol{W} \operatorname{diag}(e^{i\boldsymbol{\delta}} \tanh \boldsymbol{r}) \boldsymbol{W}^T & -\boldsymbol{W} \operatorname{diag}(\operatorname{sech} \boldsymbol{r}) \boldsymbol{V} \\ \hline -\boldsymbol{V}^T \operatorname{diag}(\operatorname{sech} \boldsymbol{r}) \boldsymbol{W}^T & -\boldsymbol{V}^T \operatorname{diag}(e^{-i\boldsymbol{\delta}} \tanh \boldsymbol{r}) \boldsymbol{V} \end{array}\right], \tag{6.2.60}$$

which corresponds to Eq. (6.2.53) with a minus sign.

6.2.5 Channel

A Gaussian channel $\Phi[\cdot]$ is uniquely determined by the triplet $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{d}$ and acts on a Gaussian state as $(\boldsymbol{V}, \bar{\boldsymbol{r}}) \mapsto (\boldsymbol{X}\boldsymbol{V}\boldsymbol{X}^T + \boldsymbol{Y}, \boldsymbol{X}\bar{\boldsymbol{r}} + \boldsymbol{d})$. We are going to find out the $\boldsymbol{A}_\Phi$, $\boldsymbol{b}_\Phi$ and c_Φ:

$$\langle \boldsymbol{i}| \left(\Phi\left[|\boldsymbol{j}\rangle\langle \boldsymbol{l}|\right]\right) |\boldsymbol{k}\rangle = c_\Phi \times \frac{G^{\boldsymbol{A}_\Phi}_{\boldsymbol{k} \oplus \boldsymbol{l} \oplus \boldsymbol{i} \oplus \boldsymbol{j}}(\boldsymbol{b}_\Phi)}{\sqrt{\boldsymbol{i}!\boldsymbol{j}!\boldsymbol{k}!\boldsymbol{l}!}}, \tag{6.2.61}$$

In this section, we employ the Choi-Jamiołkowski isomorphism [137, 138, 78] to reduce the calculation of the matrix elements of an arbitrary Gaussian channel in M to the calculation of the matrix element of a Gaussian state with $2M$. And then we will give the expressions of $\boldsymbol{A}_\Phi$, $\boldsymbol{b}_\Phi$ and c_Φ. For the unitary operator, $\boldsymbol{A}_U$, $\boldsymbol{b}_U$ and c_U can be found through the Choi-Jamiołkowski isomorphism.

Choi-Jamiołkowski isomorphism

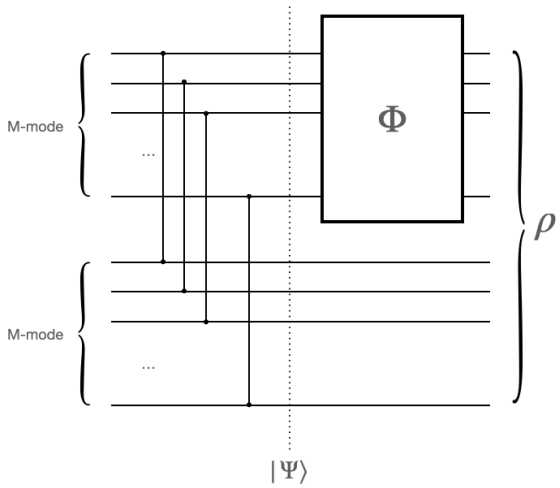


Figure 6.1: $2M$-mode circuit for implementing the Choi-Jamiołkowski isomorphism. Φ is the channel (parametrized by the triplet $\boldsymbol{X}$,$\boldsymbol{Y}$ and $\boldsymbol{d}$) applied on the first half M modes, and the two dots represent a two-mode squeezing operator connecting two modes: one comes from the first M modes, and the other one comes from the second M modes.

The Choi-Jamiołkowski isomorphism refers to the correspondence between the quantum channels and the quantum states. It is also called the channel-state duality [139]. This correspondence would help us to get the channel parameters from the corresponding states.

We first consider a collection of systems with arbitrary but identical dimensionality N.

We write the state right before the channel Φ is applied to the first half of the modes in Fig. 6.1 as

$$|\Psi\rangle = \sqrt{\mathcal{N}} \sum_{\boldsymbol{n}=\boldsymbol{0}}^{\boldsymbol{N}-1} \boldsymbol{\tau}^{\boldsymbol{n}} |\boldsymbol{n}\rangle \otimes |\boldsymbol{n}\rangle, \tag{6.2.62}$$

where $\sum_{\boldsymbol{n}=\boldsymbol{0}}^{\boldsymbol{N}-1} \equiv \sum_{n_1=0}^{N-1} \cdots \sum_{n_M=0}^{N-1}$, $\mathcal{N}$ is a normalization constant to be determined in a moment and $\boldsymbol{\tau}$ is the squeezing parameter of the two-mode squeezing operator connecting the first M modes and the second M modes. The density matrix of the state $|\Psi\rangle$ is simply

$$|\Psi\rangle\langle\Psi| = \mathcal{N} \sum_{\boldsymbol{m}=\boldsymbol{0}}^{\boldsymbol{N}-1} \sum_{\boldsymbol{n}=\boldsymbol{0}}^{\boldsymbol{N}-1} \boldsymbol{\tau}^{\boldsymbol{n}+\boldsymbol{m}} |\boldsymbol{n}\rangle\langle\boldsymbol{m}| \otimes |\boldsymbol{n}\rangle\langle\boldsymbol{m}|. \tag{6.2.63}$$

We can now write the output of the circuit after the application of the channel Φ as

$$\rho = (\Phi \otimes \mathbb{I}) \left[|\Psi\rangle\langle\Psi|\right] = \mathcal{N} \sum_{\boldsymbol{m}=\boldsymbol{0}}^{\boldsymbol{N}-1} \sum_{\boldsymbol{n}=\boldsymbol{0}}^{\boldsymbol{N}-1} \boldsymbol{\tau}^{\boldsymbol{n}+\boldsymbol{m}} \Phi \left[|\boldsymbol{n}\rangle\langle\boldsymbol{m}|\right] \otimes |\boldsymbol{n}\rangle\langle\boldsymbol{m}|. \tag{6.2.64}$$

We can premultiply the equation above by $\langle\boldsymbol{i}| \otimes \langle\boldsymbol{j}|$ and postmultiply by $|\boldsymbol{k}\rangle \otimes |\boldsymbol{l}\rangle$ to obtain

$$\left(\langle\boldsymbol{i}| \otimes \langle\boldsymbol{j}|\right) \rho \left(|\boldsymbol{k}\rangle \otimes |\boldsymbol{l}\rangle\right) = \mathcal{N}\, \boldsymbol{\tau}^{\boldsymbol{j}+\boldsymbol{l}} \langle\boldsymbol{i}| \left(\Phi \left[|\boldsymbol{j}\rangle\langle\boldsymbol{l}|\right]\right) |\boldsymbol{k}\rangle. \tag{6.2.65}$$

In finite-dimensional systems it is convenient to pick $\boldsymbol{\tau} = (1, \ldots, 1)$ and the normalization $\mathcal{N}$ is simply given by the dimensionality of the system N^M. For infinite dimensional systems, if one were to try to pick the

same normalization as for a finite-dimensional, one would obtain a non-normalizable state $|\Psi\rangle$. Thus it is convenient to pick $\boldsymbol{\tau} = (\tau, \ldots, \tau)$ with $\tau = \tanh t < 1$ and then

$$\mathcal{N} = (1-\tau^2)^M = (1-\tanh^2 t)^M, \tag{6.2.66}$$

$$\boldsymbol{\tau}^{\boldsymbol{l}+\boldsymbol{j}} = (\tanh t)^{\sum_{i=1}^M l_i + j_i}. \tag{6.2.67}$$

For a rigorous justification of this derivation, see sec 5.5 of Serafini [78]. Now consider the case where the channel Φ is Gaussian parametrized by

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{X}_{qq} & \boldsymbol{X}_{qp} \\ \boldsymbol{X}_{pq} & \boldsymbol{X}_{pp} \end{bmatrix}, \quad \boldsymbol{Y} = \begin{bmatrix} \boldsymbol{Y}_{qq} & \boldsymbol{Y}_{qp} \\ \boldsymbol{Y}_{pq} & \boldsymbol{Y}_{pp} \end{bmatrix}, \quad \boldsymbol{d} = \begin{bmatrix} \boldsymbol{d}_q \\ \boldsymbol{d}_p \end{bmatrix}. \tag{6.2.68}$$

Then the output state is also Gaussian since the input state to the channel is nothing but one-half of a two-mode squeezed state. In this case, we can write the quadrature covariance matrix and vector of means of the output state as

$$\boldsymbol{V} = \tilde{\boldsymbol{X}}\boldsymbol{\mathcal{T}}(t)\left(\frac{\hbar}{2}\mathbb{1}_{4M}\right)\boldsymbol{\mathcal{T}}(t)^T\tilde{\boldsymbol{X}}^T + \tilde{\boldsymbol{Y}} = \frac{\hbar}{2}\tilde{\boldsymbol{X}}\boldsymbol{\mathcal{T}}(2t)\tilde{\boldsymbol{X}}^T + \tilde{\boldsymbol{Y}}, \quad \bar{\boldsymbol{r}} = \begin{bmatrix} \boldsymbol{d}_q \\ \boldsymbol{0} \\ \boldsymbol{d}_p \\ \boldsymbol{0} \end{bmatrix}, \tag{6.2.69}$$

where

$$\tilde{\boldsymbol{X}} = \begin{bmatrix} \boldsymbol{X}_{qq} & 0_M & \boldsymbol{X}_{qp} & 0_M \\ 0_M & \mathbb{1}_M & 0_M & 0_M \\ \boldsymbol{X}_{pq} & 0_M & \boldsymbol{X}_{pp} & 0_M \\ 0_M & 0_M & 0_M & \mathbb{1}_M \end{bmatrix}, \quad \tilde{\boldsymbol{Y}} = \begin{bmatrix} \boldsymbol{Y}_{qq} & 0_M & \boldsymbol{Y}_{qp} & 0_M \\ 0_M & 0_M & 0_M & 0_M \\ \boldsymbol{Y}_{pq} & 0_M & \boldsymbol{Y}_{pp} & 0_M \\ 0_M & 0_M & 0_M & 0_M \end{bmatrix}, \tag{6.2.70}$$

$$\boldsymbol{\mathcal{T}}(t) = \begin{bmatrix} \cosh t\mathbb{1}_M & \sinh t\mathbb{1}_M & 0_M & 0_M \\ \sinh t\mathbb{1}_M & \cosh t\mathbb{1}_M & 0_M & 0_M \\ 0_M & 0_M & \cosh t\mathbb{1}_M & -\sinh t\mathbb{1}_M \\ 0_M & 0_M & -\sinh t\mathbb{1}_M & \cosh t\mathbb{1}_M \end{bmatrix}, \tag{6.2.71}$$

and we used the fact that $\boldsymbol{\mathcal{T}}(t)\boldsymbol{\mathcal{T}}(t)^T = \boldsymbol{\mathcal{T}}(t)\boldsymbol{\mathcal{T}}(t) = \boldsymbol{\mathcal{T}}(2t)$ and $\boldsymbol{\mathcal{T}}$ is the symplectic matrix of two-mode squeezing operator defined in Eq. (3.48).

Now, we are going to calculate the $\boldsymbol{A}_\rho, \boldsymbol{b}_\rho, c_\rho$ for the output state, and take out three entities $\boldsymbol{A}_\Phi, \boldsymbol{b}_\Phi, c_\Phi$ of the channels from them.

In turn, the complex covariance matrix $\boldsymbol{\sigma}$ of the output state is

$$\boldsymbol{\sigma} = \boldsymbol{W}\left(\frac{1}{2}\tilde{\boldsymbol{X}}\boldsymbol{\mathcal{T}}(2t)\tilde{\boldsymbol{X}}^T + \frac{\tilde{\boldsymbol{Y}}}{\hbar}\right)\boldsymbol{W}^\dagger, \tag{6.2.72}$$

where $\boldsymbol{W}$ is defined in Eq.(3.6) and it is unitary. Note that $(\boldsymbol{\mathcal{T}}(t))^T = \boldsymbol{\mathcal{T}}(t)$ is symmetric, $\tilde{\boldsymbol{X}}$ is symplectic if $X = \left[\begin{smallmatrix} X_{qq} & X_{qp} \\ X_{pq} & X_{pp} \end{smallmatrix}\right]$ is symplectic.

Let

$$\boldsymbol{Q}' = \left(\frac{\mathbb{1}_{4M}}{2} + \frac{1}{2}\tilde{\boldsymbol{X}}\boldsymbol{\mathcal{T}}(2t)\tilde{\boldsymbol{X}}^T + \frac{\tilde{\boldsymbol{Y}}}{\hbar}\right), \tag{6.2.73}$$

then $(\boldsymbol{\sigma} + \frac{\mathbb{1}_{4M}}{2})^{-1} = \boldsymbol{W}(\boldsymbol{Q}')^{-1}\boldsymbol{W}^\dagger$. Now we define

$$\boldsymbol{Q} = \boldsymbol{L}\boldsymbol{Q}'\boldsymbol{L}^T, \tag{6.2.74}$$

with

$$\boldsymbol{L} = \begin{bmatrix} \mathbb{1}_M & 0_M & 0_M & 0_M \\ 0_M & 0_M & \mathbb{1}_M & 0_M \\ 0_M & \mathbb{1}_M & 0_M & 0_M \\ 0_M & 0_M & 0_M & \mathbb{1}_M \end{bmatrix}. \tag{6.2.75}$$

Then we have that $\boldsymbol{Q}^{-1} = \boldsymbol{L}(Q')^{-1}\boldsymbol{L}^T$, which implies that $\boldsymbol{L}^T\boldsymbol{Q}^{-1}\boldsymbol{L} = (\boldsymbol{Q}')^{-1}$. So calculating $\boldsymbol{Q}^{-1}$ gives $(\boldsymbol{Q}')^{-1}$ and therefore $(\boldsymbol{\sigma} + \frac{\mathbb{1}_{4M}}{2})^{-1}$.

Expressing $\boldsymbol{Q}$ as a block matrix $\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix}$, we can write $\boldsymbol{Q}^{-1}$ using Schur complements as [78]

$$\boldsymbol{Q}^{-1} = \begin{bmatrix} \boldsymbol{\xi}^{-1} & -\boldsymbol{\xi}^{-1}\boldsymbol{B}\boldsymbol{D}^{-1} \\ -\boldsymbol{D}^{-1}\boldsymbol{C}\boldsymbol{\xi}^{-1} & \boldsymbol{D}^{-1} + \boldsymbol{D}^{-1}\boldsymbol{C}\boldsymbol{\xi}^{-1}\boldsymbol{B}\boldsymbol{D}^{-1} \end{bmatrix}, \tag{6.2.76}$$

where the Schur complement is $\boldsymbol{\xi} = \boldsymbol{A} - \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C}$. The blocks $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, and $\boldsymbol{D}$, are given by

$$\boldsymbol{A} = \frac{\boldsymbol{Y}}{\hbar} + \frac{\mathbb{1}_{2M}}{2} + \frac{1}{2}\cosh(2t)\boldsymbol{X}\boldsymbol{X}^T, \tag{6.2.77}$$

$$\boldsymbol{B} = \frac{1}{2}\sinh 2t \begin{bmatrix} \boldsymbol{X}_{qq} & -\boldsymbol{X}_{qp} \\ \boldsymbol{X}_{pq} & -\boldsymbol{X}_{pp} \end{bmatrix} = \frac{1}{2}\sinh 2t\boldsymbol{X}\boldsymbol{Z}, \tag{6.2.78}$$

$$\boldsymbol{C} = \frac{1}{2}\sinh 2t \begin{bmatrix} \boldsymbol{X}_{qq}^T & \boldsymbol{X}_{pq}^T \\ -\boldsymbol{X}_{qp}^T & -\boldsymbol{X}_{pp}^T \end{bmatrix} = \boldsymbol{B}^T = \frac{1}{2}\sinh 2t\boldsymbol{Z}\boldsymbol{X}^T, \tag{6.2.79}$$

$$\boldsymbol{D} = \cosh^2(t) \begin{bmatrix} \mathbb{1}_M & 0_M \\ 0_M & \mathbb{1}_M \end{bmatrix}, \tag{6.2.80}$$

where $\boldsymbol{Z} = \left[\begin{smallmatrix} \mathbb{1}_M & 0_M \\ 0_M & -\mathbb{1}_M \end{smallmatrix}\right]$. We now use these to calculate the blocks of $\boldsymbol{Q}^{-1}$ starting with $\boldsymbol{\xi}$,

$$\boldsymbol{\xi} = \boldsymbol{A} - \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C} = \frac{1}{2}\left(\mathbb{1}_{2M} + \boldsymbol{X}\boldsymbol{X}^T + \frac{2\boldsymbol{Y}}{\hbar}\right) = \boldsymbol{\xi}^T, \tag{6.2.81}$$

which turns out to be *independent* of t. Next, we find

$$-\boldsymbol{\xi}^{-1}\boldsymbol{B}\boldsymbol{D}^{-1} = -\tanh(t)\boldsymbol{\xi}^{-1}\boldsymbol{X}\boldsymbol{Z}, \tag{6.2.82}$$

$$-\boldsymbol{D}^{-1}\boldsymbol{C}\boldsymbol{\xi}^{-1} = -\tanh(t)\boldsymbol{Z}\boldsymbol{X}^T\boldsymbol{\xi}^{-1}. \tag{6.2.83}$$

Finally, the bottom right block, which can be simplified by substituting the other three blocks, is given by

$$\boldsymbol{D}^{-1} + \boldsymbol{D}^{-1}\boldsymbol{C}\boldsymbol{\xi}^{-1}\boldsymbol{B}\boldsymbol{D}^{-1} = \left(1 - \tanh^2(t)\right)\mathbb{1}_{2M} + \tanh^2(t)\boldsymbol{Z}\boldsymbol{X}^T\boldsymbol{\xi}^{-1}\boldsymbol{X}\boldsymbol{Z}, \tag{6.2.84}$$

$$= \mathbb{1}_{2M} + \tanh^2(t)\boldsymbol{Z}\left(\boldsymbol{X}^T\boldsymbol{\xi}^{-1}\boldsymbol{X} - \mathbb{1}_{2M}\right)\boldsymbol{Z}. \tag{6.2.85}$$

Putting these blocks together, we get the expanded form of $\boldsymbol{Q}^{-1}$

$$\boldsymbol{Q}^{-1} = \begin{bmatrix} \boldsymbol{\xi}^{-1} & -\tanh(t)\boldsymbol{\xi}^{-1}\boldsymbol{X}\boldsymbol{Z} \\ -\tanh(t)\boldsymbol{Z}\boldsymbol{X}^T\boldsymbol{\xi}^{-1} & \mathbb{1}_{2M} + \tanh^2(t)Z\left(\boldsymbol{X}^T\boldsymbol{\xi}^{-1}\boldsymbol{X} - \mathbb{1}_{2M}\right)\boldsymbol{Z} \end{bmatrix}. \tag{6.2.86}$$

Now we known $\boldsymbol{Q}^{-1}$ and then $\left(\boldsymbol{\sigma} + \frac{\mathbb{1}_{4M}}{2}\right)^{-1}$, we can use Eq. (6.2.20) and Eq. (6.2.21) to write down $\boldsymbol{A}_\rho$ and $\boldsymbol{b}_\rho$ for the output state.

We can write

$$\begin{aligned}\boldsymbol{A}_\rho = \boldsymbol{P}_{2M}\underbrace{\left(\mathbb{1}_{4M}-\boldsymbol{\sigma}_+^{-1}\right)}&\\ \mathbb{1}_{4M}-\left(\boldsymbol{\sigma}+\frac{\mathbb{1}_{4M}}{2}\right)^{-1}&\\ =\boldsymbol{W}\boldsymbol{L}^T\left(\mathbb{1}_{4M}-\begin{bmatrix}\boldsymbol{\xi}^{-1} & -\tanh(t)\boldsymbol{\xi}^{-1}\boldsymbol{X}\boldsymbol{Z}\\ -\tanh(t)\boldsymbol{Z}\boldsymbol{X}^T\boldsymbol{\xi}^{-1} & \mathbb{1}_{2M}+\tanh^2(t)\boldsymbol{Z}\left(\boldsymbol{X}^T\boldsymbol{\xi}^{-1}\boldsymbol{X}-\mathbb{1}_{2M}\right)\boldsymbol{Z}\end{bmatrix}\right)\boldsymbol{L}\boldsymbol{W}^\dagger,&\\ =\boldsymbol{W}\boldsymbol{L}^T\begin{bmatrix}\mathbb{1}_{2M}-\boldsymbol{\xi}^{-1} & \tanh(t)\boldsymbol{\xi}^{-1}\boldsymbol{X}\boldsymbol{Z}\\ \tanh(t)\boldsymbol{Z}\boldsymbol{X}^T\boldsymbol{\xi}^{-1} & \tanh^2(t)\boldsymbol{Z}\left[\mathbb{1}_{2M}-\boldsymbol{X}^T\boldsymbol{\xi}^{-1}\boldsymbol{X}\right]\boldsymbol{Z}\end{bmatrix}\boldsymbol{L}\boldsymbol{W}^\dagger.&\quad(6.2.87)\end{aligned}$$

Defining the matrix $\boldsymbol{F}=\left[\begin{smallmatrix}\mathbb{1}_{2M} & 0_{2M}\\ 0_{2M} & \boldsymbol{Z}\tanh(t)\end{smallmatrix}\right]$, we can rewrite the last equation as

$$\begin{aligned}\mathbb{1}_{4M}-\left(\boldsymbol{\sigma}+\frac{\mathbb{1}_{4M}}{2}\right)^{-1}&=\boldsymbol{W}\boldsymbol{L}^T\boldsymbol{F}\begin{bmatrix}\mathbb{1}_{2M}-\boldsymbol{\xi}^{-1} & \boldsymbol{\xi}^{-1}\boldsymbol{X}\\ \boldsymbol{X}^T\boldsymbol{\xi}^{-1} & \mathbb{1}_{2M}-\boldsymbol{X}^T\boldsymbol{\xi}^{-1}\boldsymbol{X}\end{bmatrix}\boldsymbol{F}^T\boldsymbol{L}\boldsymbol{W}^\dagger,&(6.2.88)\\ &=\boldsymbol{E}(t)\boldsymbol{R}\begin{bmatrix}\mathbb{1}_{2M}-\boldsymbol{\xi}^{-1} & \boldsymbol{\xi}^{-1}X\\ X^T\boldsymbol{\xi}^{-1} & \mathbb{1}_{2M}-\boldsymbol{X}^T\boldsymbol{\xi}^{-1}\boldsymbol{X}\end{bmatrix}\boldsymbol{R}^\dagger\boldsymbol{E}(t),&(6.2.89)\end{aligned}$$

where we noted that $\boldsymbol{W}\boldsymbol{L}^T\boldsymbol{F}=\boldsymbol{W}\boldsymbol{L}\boldsymbol{F}=\boldsymbol{E}(t)\boldsymbol{R}$, where

$$\boldsymbol{E}(t)=\mathbb{1}_M\oplus\left(\tanh t\mathbb{1}_M\right)\oplus\mathbb{1}_M\oplus\left(\tanh t\mathbb{1}_M\right),\tag{6.2.90}$$

and

$$\boldsymbol{R}=\frac{1}{\sqrt{2}}\begin{bmatrix}\mathbb{1}_M & i\mathbb{1}_M & 0_M & 0_M\\ 0_M & 0 & \mathbb{1}_M & -i\mathbb{1}_M\\ \mathbb{1}_M & -i\mathbb{1}_M & 0_M & 0_M\\ 0_M & 0_M & \mathbb{1}_M & i\mathbb{1}_M\end{bmatrix},\tag{6.2.91}$$

$$\boldsymbol{P}_M=\left[\begin{smallmatrix}0_M & \mathbb{1}_M\\ \mathbb{1}_M & 0_M\end{smallmatrix}\right].\tag{6.2.92}$$

Then we have

$$\boldsymbol{A}_\rho=\boldsymbol{P}_{2M}\left(\mathbb{1}_{4M}-\boldsymbol{\sigma}_+^{-1}\right)=\boldsymbol{P}_{2M}\boldsymbol{E}(t)\boldsymbol{R}\begin{bmatrix}\mathbb{1}_{2M}-\boldsymbol{\xi}^{-1} & \boldsymbol{\xi}^{-1}X\\ X^T\boldsymbol{\xi}^{-1} & \mathbb{1}_{2M}-\boldsymbol{X}^T\boldsymbol{\xi}^{-1}\boldsymbol{X}\end{bmatrix}\boldsymbol{R}^\dagger\boldsymbol{E}(t).\tag{6.2.93}$$

Note that $\boldsymbol{E}(t)$ and $\boldsymbol{P}_{2M}$ commute with each other so that we can write it like:

$$\boldsymbol{A}_\rho=\boldsymbol{E}(t)\boldsymbol{P}_{2M}\boldsymbol{R}\begin{bmatrix}\mathbb{1}_{2M}-\boldsymbol{\xi}^{-1} & \boldsymbol{\xi}^{-1}X\\ X^T\boldsymbol{\xi}^{-1} & \mathbb{1}_{2M}-\boldsymbol{X}^T\boldsymbol{\xi}^{-1}\boldsymbol{X}\end{bmatrix}\boldsymbol{R}^\dagger\boldsymbol{E}(t)=\boldsymbol{E}(t)\boldsymbol{A}_\Phi\boldsymbol{E}(t).\tag{6.2.94}$$

We would also like to find

$$\begin{aligned}\boldsymbol{b}_\rho=\left(\boldsymbol{\sigma}_+^{-1}\bar{\boldsymbol{\mu}}\right)^*=\left(\boldsymbol{W}\boldsymbol{L}\boldsymbol{Q}^{-1}\boldsymbol{L}\boldsymbol{W}^\dagger\begin{bmatrix}\frac{1}{\sqrt{\hbar}}\boldsymbol{W}\bar{\boldsymbol{r}}\end{bmatrix}\right)^*&=\frac{1}{\sqrt{\hbar}}\left(\boldsymbol{W}\boldsymbol{L}\boldsymbol{Q}^{-1}\boldsymbol{L}\bar{\boldsymbol{r}}\right)^*=\frac{1}{\sqrt{\hbar}}(\boldsymbol{W}\boldsymbol{L})^*\boldsymbol{Q}^{-1}\begin{bmatrix}\boldsymbol{d}\\ \boldsymbol{0}\end{bmatrix},&(6.2.95)\\ =\frac{1}{\sqrt{\hbar}}(\boldsymbol{W}\boldsymbol{L})^*\begin{bmatrix}\boldsymbol{\xi}^{-1}\boldsymbol{d}\\ -\tanh t\boldsymbol{Z}\boldsymbol{X}^T\boldsymbol{\xi}^{-1}\boldsymbol{d}\end{bmatrix}&\\ =\frac{1}{\sqrt{\hbar}}(\boldsymbol{W}\boldsymbol{L}\boldsymbol{F})^*\begin{bmatrix}\boldsymbol{\xi}^{-1}\boldsymbol{d}\\ -\boldsymbol{X}^T\boldsymbol{\xi}^{-1}\boldsymbol{d}\end{bmatrix}&\\ =\frac{1}{\sqrt{\hbar}}\boldsymbol{E}(t)\boldsymbol{R}^*\begin{bmatrix}\boldsymbol{\xi}^{-1}\boldsymbol{d}\\ -\boldsymbol{X}^T\boldsymbol{\xi}^{-1}\boldsymbol{d}\end{bmatrix}&\\ =\boldsymbol{E}(t)\boldsymbol{b}_\Phi.&(6.2.96)\end{aligned}$$

We can also obtain the expression for the scalar c

$$c_\rho = \left(\langle \mathbf{0}| \otimes \langle \mathbf{0}|\right) \rho \left(|\mathbf{0}\rangle \otimes |\mathbf{0}\rangle\right) = \mathcal{N} \ \langle \mathbf{0}| \left(\Phi \left[|\mathbf{0}\rangle\langle \mathbf{0}|\right]\right) |\mathbf{0}\rangle = \mathcal{N} c_\Phi. \tag{6.2.97}$$

The Husimi covariance matrix of the state $\Phi\left[|\mathbf{0}\rangle\langle \mathbf{0}|\right]$ is simply $\hbar\boldsymbol{\xi}$ and its vector of means is $\boldsymbol{d}$ and thus we can write

$$\langle \mathbf{0}| \left(\Phi \left[|\mathbf{0}\rangle\langle \mathbf{0}|\right]\right) |\mathbf{0}\rangle = \frac{\exp\left[-\frac{1}{2}\boldsymbol{d}^T(\hbar\boldsymbol{\xi})^{-1}\boldsymbol{d}\right]}{\sqrt{\det\left(\boldsymbol{\xi}\right)}}. \tag{6.2.98}$$

In conclusion, we use the Choi-Jamiołkowski isomorphism to get three entities of the output state and then extract the expression by getting rid of the input two-mode squeezing part $\boldsymbol{E}(t)$ in order to write the three entities of the channel:

$$\boldsymbol{A}_\rho = \boldsymbol{E}(t)\boldsymbol{A}_\Phi \boldsymbol{E}(t), \tag{6.2.99}$$

$$\boldsymbol{A}_\Phi = \boldsymbol{P}_{2M}\boldsymbol{R}\begin{bmatrix} \mathbb{1}_{2M} - \boldsymbol{\xi}^{-1} & \boldsymbol{\xi}^{-1}\boldsymbol{X} \\ \boldsymbol{X}^T\boldsymbol{\xi}^{-1} & \mathbb{1}_{2M} - \boldsymbol{X}^T\boldsymbol{\xi}^{-1}\boldsymbol{X}\end{bmatrix}\boldsymbol{R}^\dagger, \tag{6.2.100}$$

$$= \boldsymbol{P}_{2M}\boldsymbol{R}\left(\mathbb{1}_{4M} - \begin{bmatrix} \boldsymbol{\xi}^{-1} & -\boldsymbol{\xi}^{-1}\boldsymbol{X} \\ -\boldsymbol{X}^T\boldsymbol{\xi}^{-1} & \boldsymbol{X}^T\boldsymbol{\xi}^{-1}\boldsymbol{X}\end{bmatrix}\right)\boldsymbol{R}^\dagger, \tag{6.2.101}$$

$$\boldsymbol{b}_\rho = \boldsymbol{E}(t)\boldsymbol{b}_\Phi, \tag{6.2.102}$$

$$\boldsymbol{b}_\Phi = \frac{1}{\sqrt{\hbar}}\boldsymbol{R}^*\begin{bmatrix} \boldsymbol{\xi}^{-1}\boldsymbol{d} \\ -\boldsymbol{X}^T\boldsymbol{\xi}^{-1}\boldsymbol{d}\end{bmatrix}, \tag{6.2.103}$$

$$c_\rho = (1-\tanh^2 t)^M c_\Phi, \tag{6.2.104}$$

$$c_\Phi = \frac{\exp\left[-\frac{1}{2\hbar}\boldsymbol{d}^T\boldsymbol{\xi}^{-1}\boldsymbol{d}\right]}{\sqrt{\det(\boldsymbol{\xi})}}, \tag{6.2.105}$$

where $\boldsymbol{\xi} = \frac{1}{2}\left(\mathbb{1}_{2M} + \boldsymbol{X}\boldsymbol{X}^T + \frac{2\boldsymbol{Y}}{\hbar}\right)$, and $\boldsymbol{E}(t)$,$\boldsymbol{R}$, $\boldsymbol{P}$ are defined in Eqs. (6.2.90) (6.2.91) and (6.2.92).

Note that $\boldsymbol{\xi}$ is nothing but the qp-Husimi covariance matrix (in units where $\hbar = 1$) of the state obtained by sending the M mode vacuum state in the process specified by $\boldsymbol{X}$ and $\boldsymbol{Y}$.

With these results we can write

$$\left(\langle \boldsymbol{i}| \otimes \langle \boldsymbol{j}|\right) \rho \left(|\boldsymbol{k}\rangle \otimes |\boldsymbol{l}\rangle\right) = c_\rho \times \frac{G^{\boldsymbol{A}_\rho}_{\boldsymbol{k}\oplus\boldsymbol{l}\oplus\boldsymbol{i}\oplus\boldsymbol{j}}(\boldsymbol{b}_\rho)}{\sqrt{\boldsymbol{i}!\boldsymbol{j}!\boldsymbol{k}!\boldsymbol{l}!}}. \tag{6.2.106}$$

Now we recall a fundamental property that multidimensional Hermite polynomials inherit from loop-hafnians [128], namely that if $\boldsymbol{E} = \oplus_{i=1}^{\ell} E_i$ is a diagonal matrix then

$$G^{\boldsymbol{EAE}}_{\boldsymbol{n}}(\boldsymbol{Eb}) = \left(\prod_{i=1}^{\ell} E_i^{n_i}\right) G^{\boldsymbol{A}}_{\boldsymbol{n}}(\boldsymbol{b}), \tag{6.2.107}$$

We can use the definitions from Eq. (6.2.99) to Eq. (6.2.105) together with the Eq. (6.2.65) and the relation Eq. (6.2.106) to find

$$\langle \boldsymbol{i}| \left(\Phi \left[|\boldsymbol{j}\rangle\langle \boldsymbol{l}|\right]\right) |\boldsymbol{k}\rangle = \frac{\left(\langle \boldsymbol{i}| \otimes \langle \boldsymbol{j}|\right) \rho \left(|\boldsymbol{k}\rangle \otimes |\boldsymbol{l}\rangle\right)}{\mathcal{N}\ \boldsymbol{\tau}^{\boldsymbol{j}+\boldsymbol{l}}} = \frac{c_\rho}{\mathcal{N}\ \boldsymbol{\tau}^{\boldsymbol{j}+\boldsymbol{l}}} \times \frac{G^{\boldsymbol{A}_\rho}_{\boldsymbol{k}\oplus\boldsymbol{l}\oplus\boldsymbol{i}\oplus\boldsymbol{j}}(\boldsymbol{b}_\rho)}{\sqrt{\boldsymbol{i}!\boldsymbol{j}!\boldsymbol{k}!\boldsymbol{l}!}} = c_\Phi \times \frac{G^{\boldsymbol{A}_\Phi}_{\boldsymbol{k}\oplus\boldsymbol{l}\oplus\boldsymbol{i}\oplus\boldsymbol{j}}(\boldsymbol{b}_\Phi)}{\sqrt{\boldsymbol{i}!\boldsymbol{j}!\boldsymbol{k}!\boldsymbol{l}!}}, \tag{6.2.108}$$

which allows us to find the matrix elements of the channel without any reference to the specific amount of squeezing used to create the two-mode squeezed vacuum.

Channel $\boldsymbol{A}, \boldsymbol{b}, c$

To summarize, we can then write

$$\boldsymbol{A}_\Phi = \boldsymbol{P}_{2M}\boldsymbol{R}\begin{bmatrix} \mathbb{1}_{2M} - \boldsymbol{\xi}^{-1} & \boldsymbol{\xi}^{-1}\boldsymbol{X} \\ \boldsymbol{X}^T\boldsymbol{\xi}^{-1} & \mathbb{1}_{2M} - \boldsymbol{X}^T\boldsymbol{\xi}^{-1}\boldsymbol{X} \end{bmatrix}\boldsymbol{R}^\dagger, \tag{6.2.109}$$

$$\boldsymbol{b}_\Phi = \frac{1}{\sqrt{\hbar}}\boldsymbol{R}^*\begin{bmatrix} \boldsymbol{\xi}^{-1}\boldsymbol{d} \\ -\boldsymbol{X}^T\boldsymbol{\xi}^{-1}\boldsymbol{d} \end{bmatrix}, \tag{6.2.110}$$

$$c_\Phi = \frac{\exp\left[-\frac{1}{2\hbar}\boldsymbol{d}^T\boldsymbol{\xi}^{-1}\boldsymbol{d}\right]}{\sqrt{\det(\boldsymbol{\xi})}}, \tag{6.2.111}$$

and

$$\boldsymbol{R} = \frac{1}{\sqrt{2}}\begin{bmatrix} \mathbb{1}_M & i\mathbb{1}_M & 0_M & 0_M \\ 0_M & 0 & \mathbb{1}_M & -i\mathbb{1}_M \\ \mathbb{1}_M & -i\mathbb{1}_M & 0_M & 0_M \\ 0_M & 0_M & \mathbb{1}_M & i\mathbb{1}_M \end{bmatrix}, \tag{6.2.112}$$

$$\boldsymbol{\xi} = \frac{1}{2}\left(\mathbb{1}_{2M} + \boldsymbol{X}\boldsymbol{X}^T + \frac{2\boldsymbol{Y}}{\hbar}\right). \tag{6.2.113}$$

Examples

For example, for a single-mode amplifier channel with gain $g \geq 1$ we find

$$\boldsymbol{A}_\Phi = \begin{bmatrix} 0 & \frac{1}{\sqrt{g}} & \frac{g-1}{g} & 0 \\ \frac{1}{\sqrt{g}} & 0 & 0 & 0 \\ \frac{g-1}{g} & 0 & 0 & \frac{1}{\sqrt{g}} \\ 0 & 0 & \frac{1}{\sqrt{g}} & 0 \end{bmatrix}, \ \boldsymbol{b}_\Phi = \boldsymbol{0}, \ c_\Phi = 1/g. \tag{6.2.114}$$

For the case of the M-mode lossy interferometer with transmission matrix $\boldsymbol{T}$ we find

$$\boldsymbol{A}_\Phi = \begin{bmatrix} 0_M & \boldsymbol{T}^* & 0_M & 0_M \\ \boldsymbol{T}^\dagger & 0_M & 0_M & \mathbb{1}_M - \boldsymbol{T}^\dagger\boldsymbol{T} \\ 0_M & 0_M & 0_M & \boldsymbol{T} \\ 0_M & \mathbb{1}_M - \boldsymbol{T}^T\boldsymbol{T}^* & \boldsymbol{T}^T & 0_M \end{bmatrix}, \tag{6.2.115}$$

$$\boldsymbol{b}_\Phi = \boldsymbol{0}, \tag{6.2.116}$$

$$c_\Phi = 1. \tag{6.2.117}$$

Unitary operator $\boldsymbol{A}, \boldsymbol{b}, c$ via the Choi isomorphism

In the case where the channel is unitary, we can write $\Phi[\cdot] = U\{\cdot\}U^\dagger$ and then we obtain

$$\langle \boldsymbol{i}| \left(\Phi\left[|\boldsymbol{j}\rangle\langle \boldsymbol{l}|\right]\right)|\boldsymbol{k}\rangle = \langle \boldsymbol{i}|U|\boldsymbol{j}\rangle\langle \boldsymbol{l}|U^\dagger|\boldsymbol{k}\rangle. \tag{6.2.118}$$

This corresponds to the case where $\boldsymbol{Y} = 0_{2M}$ and $\boldsymbol{X} = \boldsymbol{S}$ is symplectic.

Since $\boldsymbol{S}$ is symplectic, then we can write a symplectic singular-value decomposition

$$\boldsymbol{S} = \begin{bmatrix} \Re(\boldsymbol{U}_1) & -\Im(\boldsymbol{U}_1) \\ \Im(\boldsymbol{U}_1) & \Re(\boldsymbol{U}_1) \end{bmatrix}\underbrace{\begin{bmatrix} e^{-\boldsymbol{r}} & 0_M \\ 0_M & e^{\boldsymbol{r}} \end{bmatrix}}_{\equiv\boldsymbol{\lambda}}\begin{bmatrix} \Re(\boldsymbol{U}_2) & -\Im(\boldsymbol{U}_2) \\ \Im(\boldsymbol{U}_2) & \Re(\boldsymbol{U}_2) \end{bmatrix} = \boldsymbol{O}_1\boldsymbol{\lambda}\boldsymbol{O}_2, \tag{6.2.119}$$

where $\boldsymbol{U}_1, \boldsymbol{U}_2$ are $M \times M$ unitaries and $\boldsymbol{r} = \oplus_{i=1}^M r_i$ represents squeezing. We can now calculate the Schur complement to find

$$\boldsymbol{\xi} = \frac{1}{2}\left(\mathbb{1}_{2M} + \boldsymbol{S}\boldsymbol{S}^T\right), \tag{6.2.120}$$

$$\boldsymbol{\xi}^{-1} = 2\boldsymbol{O}_1\frac{\mathbb{1}_{2M}}{\mathbb{1}_{2M} + \boldsymbol{\lambda}^2}\boldsymbol{O}_1^T, \tag{6.2.121}$$

and then we find

$$\begin{bmatrix} \mathbb{1}_{2M} - \boldsymbol{\xi}^{-1} & \boldsymbol{\xi}^{-1}\boldsymbol{X} \\ \boldsymbol{X}^T\boldsymbol{\xi}^{-1} & \mathbb{1}_{2M} - \boldsymbol{X}^T\boldsymbol{\xi}^{-1}\boldsymbol{X} \end{bmatrix} = \begin{bmatrix} \boldsymbol{O}_1 \frac{\boldsymbol{\lambda}^2 - \mathbb{1}_{2M}}{\boldsymbol{\lambda}^2 + \mathbb{1}_{2M}} \boldsymbol{O}_1^T & \boldsymbol{O}_1 \frac{2\boldsymbol{\lambda}}{\boldsymbol{\lambda}^2 + \mathbb{1}_{2M}} \boldsymbol{O}_2 \\ \boldsymbol{O}_2^T \frac{2\boldsymbol{\lambda}}{\boldsymbol{\lambda}^2 + \mathbb{1}_{2M}} \boldsymbol{O}_1^T & -\boldsymbol{O}_2^T \frac{\boldsymbol{\lambda}^2 - \mathbb{1}_{2M}}{\boldsymbol{\lambda}^2 + \mathbb{1}_{2M}} \boldsymbol{O}_2 \end{bmatrix} \tag{6.2.122}$$

$$= \begin{bmatrix} \boldsymbol{O}_1 & 0_{2M} \\ 0_{2M} & \boldsymbol{O}_2^T \end{bmatrix} \begin{bmatrix} \frac{\boldsymbol{\lambda}^2 - \mathbb{1}_{2M}}{\boldsymbol{\lambda}^2 + \mathbb{1}_{2M}} & \frac{2\boldsymbol{\lambda}}{\boldsymbol{\lambda}^2 + \mathbb{1}_{2M}} \\ \frac{2\boldsymbol{\lambda}}{\boldsymbol{\lambda}^2 + \mathbb{1}_{2M}} & -\frac{\boldsymbol{\lambda}^2 - \mathbb{1}_{2M}}{\boldsymbol{\lambda}^2 + \mathbb{1}_{2M}} \end{bmatrix} \begin{bmatrix} \boldsymbol{O}_1^T & 0_{2M} \\ 0_{2M} & \boldsymbol{O}_2 \end{bmatrix}. \tag{6.2.123}$$

Note that

$$\frac{\boldsymbol{\lambda}^2 - \mathbb{1}_{2M}}{\boldsymbol{\lambda}^2 + \mathbb{1}_{2M}} = \begin{bmatrix} -\tanh \boldsymbol{r} & 0_M \\ 0_M & \tanh \boldsymbol{r} \end{bmatrix}, \quad \frac{2\boldsymbol{\lambda}}{\boldsymbol{\lambda}^2 + \mathbb{1}_{2M}} = \begin{bmatrix} \operatorname{sech} \boldsymbol{r} & 0_M \\ 0_M & \operatorname{sech} \boldsymbol{r} \end{bmatrix}. \tag{6.2.124}$$

We can now calculate

$$\boldsymbol{R} \begin{bmatrix} \mathbb{1}_{2M} - \boldsymbol{\xi}^{-1} & \boldsymbol{\xi}^{-1}X \\ X^T\boldsymbol{\xi}^{-1} & \mathbb{1}_{2M} - X^T\boldsymbol{\xi}^{-1}X \end{bmatrix} \boldsymbol{R}^\dagger = \boldsymbol{R} \begin{bmatrix} \boldsymbol{O}_1 & 0_M \\ 0_M & \boldsymbol{O}_2^T \end{bmatrix} \boldsymbol{R}^\dagger \boldsymbol{R} \begin{bmatrix} \frac{\boldsymbol{\lambda}^2 - \mathbb{1}_{2M}}{\boldsymbol{\lambda}^2 + \mathbb{1}_{2M}} & \frac{2\boldsymbol{\lambda}}{\boldsymbol{\lambda}^2 + \mathbb{1}_{2M}} \\ \frac{2\boldsymbol{\lambda}}{\boldsymbol{\lambda}^2 + \mathbb{1}_{2M}} & -\frac{\boldsymbol{\lambda}^2 - \mathbb{1}_{2M}}{\boldsymbol{\lambda}^2 + \mathbb{1}_{2M}} \end{bmatrix} \boldsymbol{R}^\dagger \boldsymbol{R} \begin{bmatrix} \boldsymbol{O}_1^T & 0_M \\ 0_M & \boldsymbol{O}_2 \end{bmatrix} \boldsymbol{R}^\dagger \tag{6.2.125}$$

$$= \begin{bmatrix} \boldsymbol{U}_1 & 0_M & 0_M & 0_M \\ 0_M & \boldsymbol{U}_2^T & 0_M & 0_M \\ 0_M & 0_M & \boldsymbol{U}_1^* & 0_M \\ 0_M & 0_M & 0_M & \boldsymbol{U}_2^\dagger \end{bmatrix} \begin{bmatrix} 0_M & 0_M & -\tanh \boldsymbol{r} & \operatorname{sech} \boldsymbol{r} \\ 0_M & 0_M & \operatorname{sech} \boldsymbol{r} & \tanh \boldsymbol{r} \\ -\tanh \boldsymbol{r} & \operatorname{sech} \boldsymbol{r} & 0_M & 0_M \\ \operatorname{sech} \boldsymbol{r} & \tanh \boldsymbol{r} & 0_M & 0_M \end{bmatrix} \begin{bmatrix} \boldsymbol{U}_1 & 0_M & 0_M & 0_M \\ 0_M & \boldsymbol{U}_2^T & 0_M & 0_M \\ 0_M & 0_M & \boldsymbol{U}_1^* & 0_M \\ 0_M & 0_M & 0_M & \boldsymbol{U}_2^\dagger \end{bmatrix}^\dagger \tag{6.2.126}$$

$$= -\begin{bmatrix} 0_M & \boldsymbol{A}_U \\ \boldsymbol{A}_U^* & 0_M \end{bmatrix}, \tag{6.2.127}$$

where

$$\boldsymbol{A}_U = \begin{bmatrix} \boldsymbol{U}_1 & 0_M \\ 0_M & \boldsymbol{U}_2^T \end{bmatrix} \begin{bmatrix} \tanh \boldsymbol{r} & -\operatorname{sech} \boldsymbol{r} \\ -\operatorname{sech} \boldsymbol{r} & -\tanh \boldsymbol{r} \end{bmatrix} \begin{bmatrix} \boldsymbol{U}_1 & 0_M \\ 0_M & \boldsymbol{U}_2^T \end{bmatrix}^T = \boldsymbol{A}_U^T. \tag{6.2.128}$$

So we then have

$$\boldsymbol{A}_\Phi = \begin{pmatrix} \boldsymbol{A}_U^* & 0_M \\ 0_M & \boldsymbol{A}_U \end{pmatrix}, \tag{6.2.129}$$

$$\boldsymbol{b}_\Phi = \begin{pmatrix} \boldsymbol{b}_U^* & \boldsymbol{b}_U \end{pmatrix}^T, \tag{6.2.130}$$

that is

$$\boldsymbol{A}_\Phi = \boldsymbol{A}_U^* \oplus \boldsymbol{A}_U, \tag{6.2.131}$$

$$\boldsymbol{b}_\Phi = \boldsymbol{b}_U^* \oplus \boldsymbol{b}_U, \tag{6.2.132}$$

and then we have

$$\langle \boldsymbol{i} | \left(\Phi \left[|\boldsymbol{j}\rangle\langle \boldsymbol{l}| \right] \right) | \boldsymbol{k} \rangle = \frac{G^{\boldsymbol{A}_U^* \oplus \boldsymbol{A}_U}_{\boldsymbol{k} \oplus \boldsymbol{l} \oplus \boldsymbol{i} \oplus \boldsymbol{j}} (\boldsymbol{b}_U \oplus \boldsymbol{b}_U^*)}{\sqrt{\boldsymbol{i}!\boldsymbol{j}!\boldsymbol{k}!\boldsymbol{l}!}} \tag{6.2.133}$$

$$= \frac{G^{\boldsymbol{A}_U^*}_{\boldsymbol{k} \oplus \boldsymbol{l}} (\boldsymbol{b}_U^*)}{\sqrt{\boldsymbol{k}!\boldsymbol{l}!}} \times \frac{G^{\boldsymbol{A}_U}_{\boldsymbol{i} \oplus \boldsymbol{j}} (\boldsymbol{b}_U)}{\sqrt{\boldsymbol{i}!\boldsymbol{j}!}} \tag{6.2.134}$$

$$= \frac{\left[G^{\boldsymbol{A}_U}_{\boldsymbol{k} \oplus \boldsymbol{l}} (\boldsymbol{b}_U) \right]^*}{\sqrt{\boldsymbol{k}!\boldsymbol{l}!}} \times \frac{G^{\boldsymbol{A}_U}_{\boldsymbol{i} \oplus \boldsymbol{j}} (\boldsymbol{b}_U)}{\sqrt{\boldsymbol{i}!\boldsymbol{j}!}}. \tag{6.2.135}$$

Comparing Eq. (6.2.118) and the last equation we easily identify

$$\langle \boldsymbol{i} | U | \boldsymbol{j} \rangle = c_U e^{i\varphi_U} \frac{G^{\boldsymbol{A}_U}_{\boldsymbol{i} \oplus \boldsymbol{j}} (\boldsymbol{b}_U)}{\sqrt{\boldsymbol{i}!\boldsymbol{j}!}}, c_\Phi = c_U^* c_U, \tag{6.2.136}$$

Operator	$\boldsymbol{A}$	$\boldsymbol{b}$	c	$\boldsymbol{\Sigma}$	$\boldsymbol{\mu}^T$	c
$\mathcal{D}$	$\left[\begin{array}{c\|c} 0 & 1 \\ \hline 1 & 0 \end{array}\right]$	$\begin{bmatrix} \gamma \\ -\gamma^* \end{bmatrix}$	$e^{-\frac{1}{2}\lvert\gamma\rvert^2}$	$\left[\begin{array}{c\|c} 0 & -1 \\ \hline -1 & 0 \end{array}\right]$	$[\gamma, -\gamma^*]$	$e^{-\frac{1}{2}\lvert\gamma\rvert^2}$
$\mathcal{S}$	$\left[\begin{array}{c\|c} -\tanh r & \operatorname{sech} r \\ \hline \operatorname{sech} r & \tanh r \end{array}\right]$	$\mathbf{0}$	$\operatorname{sech} r$	$\left[\begin{array}{c\|c} \tanh r & -\operatorname{sech} r \\ \hline -\operatorname{sech} r & -\tanh r \end{array}\right]$	$\mathbf{0}$	$\operatorname{sech} r$

Table 6.1: Relation of $\boldsymbol{A}, \boldsymbol{b}, c$ and $\boldsymbol{\Sigma}, \boldsymbol{\mu}, c$.

where φ_U is a phase that will be discussed in the next section.

Note that the quantities $c_U, \boldsymbol{b}_U$ and $\boldsymbol{A}_U$ correspond to the $C, \boldsymbol{\mu}, -\boldsymbol{\Sigma}$ introduced in Eq. (26) of Ref. [35]. And we show examples in Tab. 6.1 with the displacement operator $\mathcal{D}$ and squeezing operator $\mathcal{S}$.

Non-unitary operator $\boldsymbol{A}, \boldsymbol{b}, c$ via the Choi isomorphism

Now consider the case of non-unitary passive process specified by a transfer matrix $\boldsymbol{T}$, $\boldsymbol{T}^\dagger\boldsymbol{T} \leq \mathbb{1}_M$. For this process $\boldsymbol{X} = \begin{bmatrix} \Re(\boldsymbol{T}) & -\Im(\boldsymbol{T}) \\ \Im(\boldsymbol{T}) & \Re(\boldsymbol{T}) \end{bmatrix}$ and $\boldsymbol{Y} = \frac{\hbar}{2}\left(\mathbb{1}_{2M} - \boldsymbol{X}\boldsymbol{X}^T\right)$.

Since the process is passive we know that $\boldsymbol{\xi} = \mathbb{1}_{2M}$. We can simplify the expression to obtain

$$\boldsymbol{P}_{2M}\left[\mathbb{1}_{4M} - \left(\boldsymbol{\sigma} + \frac{\mathbb{1}_{4M}}{2}\right)^{-1}\right] = \boldsymbol{E}(t)\begin{bmatrix} 0_M & \boldsymbol{T}^* & 0_M & 0_M \\ \boldsymbol{T}^\dagger & 0_M & 0_M & \mathbb{1}_M - \boldsymbol{T}^\dagger\boldsymbol{T} \\ 0_M & 0_M & 0_M & \boldsymbol{T} \\ 0_M & \mathbb{1}_M - \boldsymbol{T}^T\boldsymbol{T}^* & \boldsymbol{T}^T & 0_M \end{bmatrix}\boldsymbol{E}(t) \tag{6.2.137}$$

.

Following the Choi-Jamiołkowski relation we gave in Eq. (6.2.99), the $\boldsymbol{A}_\Phi$ for the lossy interferometer is

$$\boldsymbol{A}_\Phi = \begin{bmatrix} 0_M & \boldsymbol{T}^* & 0_M & 0_M \\ \boldsymbol{T}^\dagger & 0_M & 0_M & \mathbb{1}_M - \boldsymbol{T}^\dagger\boldsymbol{T} \\ 0_M & 0_M & 0_M & \boldsymbol{T} \\ 0_M & \mathbb{1}_M - \boldsymbol{T}^T\boldsymbol{T}^* & \boldsymbol{T}^T & 0_M \end{bmatrix}. \tag{6.2.138}$$

If we sandwich $\boldsymbol{A}_\Phi$ with a permutation matrix $\boldsymbol{P}_{4123}$, we would have:

$$\boldsymbol{P}_{4123}\boldsymbol{A}_\Phi\boldsymbol{P}_{4123}^T = \begin{bmatrix} \mathbf{0} & \begin{matrix} \mathbb{1}_M - \boldsymbol{T}^\dagger\boldsymbol{T} & \boldsymbol{T}^\dagger \\ \boldsymbol{T} & 0 \end{matrix} \\ \begin{matrix} \mathbb{1}_M - \boldsymbol{T}^T\boldsymbol{T}^* & \boldsymbol{T}^T \\ \boldsymbol{T}^* & 0 \end{matrix} & \mathbf{0} \end{bmatrix}. \tag{6.2.139}$$

To get the probability with a measurement of photon number pattern $\boldsymbol{j} = (j_1, \ldots, j_M)$ given the input $\boldsymbol{i} = (i_1, \ldots, i_M)$, we let $\boldsymbol{k} = \boldsymbol{i}, \boldsymbol{l} = \boldsymbol{j}$ in (6.2.61):

$$G^{\boldsymbol{A}_\Phi}_{\boldsymbol{i}\oplus\boldsymbol{j}\oplus\boldsymbol{i}\oplus\boldsymbol{j}}(\mathbf{0}) = G^{\boldsymbol{P}_{4123}\boldsymbol{A}_\Phi\boldsymbol{P}_{4123}^T}_{\boldsymbol{P}_{4123}(\boldsymbol{i}\oplus\boldsymbol{j}\oplus\boldsymbol{i}\oplus\boldsymbol{j})}(\mathbf{0}) \tag{6.2.140}$$

$$= \frac{1}{\boldsymbol{i}!\boldsymbol{j}!}\operatorname{haf}(\boldsymbol{P}_{4123}\boldsymbol{A}_\Phi\boldsymbol{P}_{4123}^T)_{\boldsymbol{j}\oplus\boldsymbol{i}\oplus\boldsymbol{j}\oplus\boldsymbol{i}} \tag{6.2.141}$$

$$= \frac{1}{\boldsymbol{i}!\boldsymbol{j}!}\operatorname{perm}\begin{bmatrix} \mathbb{1}_M - \boldsymbol{T}^\dagger\boldsymbol{T} & \boldsymbol{T}^\dagger \\ \boldsymbol{T} & 0 \end{bmatrix}_{\boldsymbol{j}\oplus\boldsymbol{i}}. \tag{6.2.142}$$

6.2.6 Conclusion for recurrence relation of Gaussian objects

The summary of all recurrence relations is shown in Table 6.2.

Tab. 6.3 gives the characteristics of typical Gaussian objects (note that $\boldsymbol{A}$ is 1×1 for single-mode pure states).

Object	$\boldsymbol{A}$	$\boldsymbol{b}$	c	
Mixed state ρ	$\boldsymbol{P}_M(\mathbb{1} - \boldsymbol{\sigma}_+^{-1})$	$\boldsymbol{P}_M \boldsymbol{\sigma}_+^{-1} \bar{\boldsymbol{\mu}}$	$\frac{\exp\left[-\frac{1}{2}\bar{\boldsymbol{\mu}}^\dagger \boldsymbol{\sigma}_+^{-1} \bar{\boldsymbol{\mu}}\right]}{\sqrt{\det(\boldsymbol{\sigma}_+)}}$	$\boldsymbol{A}_\rho = \boldsymbol{A}_\psi^* \oplus \boldsymbol{A}_\psi,$
Pure state ψ	*	*	$c_\rho = c_\psi^* c_\psi$	$\boldsymbol{b}_\rho = \boldsymbol{b}_\psi^* \oplus \boldsymbol{b}_\psi$
Channel Φ	$\boldsymbol{P}_{2M} \boldsymbol{R} \begin{bmatrix} \mathbb{1}_{2M} - \boldsymbol{\xi}^{-1} & \boldsymbol{\xi}^{-1}\boldsymbol{X} \\ \boldsymbol{X}^T\boldsymbol{\xi}^{-1} & \mathbb{1}_{2M} - \boldsymbol{X}^T\boldsymbol{\xi}^{-1}\boldsymbol{X} \end{bmatrix} \boldsymbol{R}^\dagger$	$\frac{1}{\sqrt{\hbar}} \boldsymbol{R}^* \begin{bmatrix} \boldsymbol{\xi}^{-1}\boldsymbol{d} \\ -\boldsymbol{X}^T\boldsymbol{\xi}^{-1}\boldsymbol{d} \end{bmatrix}$	$\frac{\exp\left[-\frac{1}{2\hbar}\boldsymbol{d}^T\boldsymbol{\xi}^{-1}\boldsymbol{d}\right]}{\sqrt{\det(\boldsymbol{\xi})}}$	$\boldsymbol{A}_\Phi = \boldsymbol{A}_U^* \oplus \boldsymbol{A}_U,$
Transformation U	$\begin{bmatrix} \boldsymbol{Q}^*\boldsymbol{P}^{-1} & (\boldsymbol{P}^T)^{-1} \\ \boldsymbol{P}^{-1} & \boldsymbol{P}^{-1}\boldsymbol{Q} \end{bmatrix}$	*	$c_\Phi = c_U^* c_U$	$\boldsymbol{b}_\Phi = \boldsymbol{b}_U^* \oplus \boldsymbol{b}_U$

Table 6.2: $\boldsymbol{A}, \boldsymbol{b}, c$ for Gaussian objects. The * means we do not give the equation to get the value directly.

Pure State	$\boldsymbol{A}_\psi$	$\boldsymbol{b}_\psi$	c_ψ
single-mode coherent	0	α	$e^{-\lvert\alpha\rvert^2}$
single-mode squeezed vacuum	$-\tanh(r)e^{i\phi}$	0	$\sqrt{\operatorname{sech}(r)}$
single-mode displaced squeezed	$-\tanh(r)e^{i\phi}$	α	$\sqrt{\operatorname{sech}(r)}e^{-\lvert\alpha\rvert^2}$
thermal state	$\frac{\bar{n}}{1+\bar{n}} \left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$	0	$\frac{1}{1+\bar{n}}$
Unitary	$\boldsymbol{A}_U$	$\boldsymbol{b}_U$	c_U
single-mode displacement	$\left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$	$\left[\begin{smallmatrix} \gamma \\ -\gamma^* \end{smallmatrix}\right]$	$e^{-\frac{1}{2}\lvert\gamma\rvert^2}$
single-mode squeezing	$\begin{bmatrix} -e^{i\delta}\tanh(r) & \operatorname{sech}(r) \\ \operatorname{sech}(r) & e^{-i\delta}\tanh(r) \end{bmatrix}$	0_2	$\sqrt{\operatorname{sech}(r)}$
single-mode rotation	$\begin{bmatrix} 0 & e^{i\phi} \\ e^{i\phi} & 0 \end{bmatrix}$	0_2	1
Channel	$\boldsymbol{A}_\Phi$	$\boldsymbol{b}_\Phi$	c_Φ
single-mode amplifier	$\begin{bmatrix} 0 & \frac{1}{\sqrt{g}} & \frac{g-1}{g} & 0 \\ \frac{1}{\sqrt{g}} & 0 & 0 & 0 \\ \frac{g-1}{g} & 0 & 0 & \frac{1}{\sqrt{g}} \\ 0 & 0 & \frac{1}{\sqrt{g}} & 0 \end{bmatrix}$	0_4	$\frac{1}{g}$
M-mode lossy interferometer	$\begin{bmatrix} \boldsymbol{0}_M & \boldsymbol{T}^* & \boldsymbol{0}_M & \boldsymbol{0}_M \\ \boldsymbol{T}^\dagger & \boldsymbol{0}_M & \boldsymbol{0}_M & \mathbb{1}_M - \boldsymbol{T}^\dagger\boldsymbol{T} \\ \boldsymbol{0}_M & \boldsymbol{0}_M & \boldsymbol{0}_M & \boldsymbol{T} \\ \boldsymbol{0}_M & \mathbb{1}_M - \boldsymbol{T}^T\boldsymbol{T}^* & \boldsymbol{T}^T & \boldsymbol{0}_M \end{bmatrix}$	0_4	1

Table 6.3: Useful results for Gaussian states, unitaries, and channels.

6.3 Explicit global phase formula for Gaussian operators

In the phase space representation, transformations are specified by a symplectic matrix and a displacement vector. However, these two quantities do not uniquely specify the evolution of a quantum state. For example, when two displacement operators with parameters $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ are composed in the Gaussian representation, their effect is just another displacement with parameter $\boldsymbol{d} = \boldsymbol{d}_1 + \boldsymbol{d}_2$. However, the unitary representation acquires a global phase:

$$\mathcal{D}(\alpha)\mathcal{D}(\beta) = e^{(\alpha\beta^* - \alpha^*\beta)/2}\mathcal{D}\left(\alpha + \beta\right), \tag{6.3.1}$$

i.e. we do not only add up both displacement parameters $\mathcal{D}(\alpha + \beta)$ here, but also get an extra part $e^{(\alpha\beta^* - \alpha^*\beta)/2}$, which is a *global phase*. Such a global phase is important when evolving linear combinations of Gaussian states with Gaussian operators [140]. This section will compute this global phase and

provide some examples.

Eq. (23) of Ref. [35] shows that the Husimi Q function of an arbitrary Gaussian unitary can be characterized by three quantities $C, \boldsymbol{\mu}$, and $\boldsymbol{\Sigma}$. As we already know the relation between $C, \boldsymbol{\mu}, \boldsymbol{\Sigma}$ and $c_U, \boldsymbol{b}_U, \boldsymbol{A}_U$ in the previous section, now we will rewrite the Husimi Q function for an arbitrary Gaussian unitary as:

$$\langle \boldsymbol{\alpha}^* | \mathcal{G} | \boldsymbol{\beta} \rangle = \exp\left(-\tfrac{1}{2}\left[||\boldsymbol{\alpha}||^2 + ||\boldsymbol{\beta}||^2\right]\right) c_U \exp\left(\boldsymbol{b}_U^T \boldsymbol{\nu} + \frac{1}{2}\boldsymbol{\nu}^T \boldsymbol{A_U} \boldsymbol{\nu}\right), \tag{6.3.2}$$

where

$$\boldsymbol{\nu} = \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix}. \tag{6.3.3}$$

If we compose two Gaussian operators, they should be able to write as a new Gaussian operator with an extra *global phase*:

$$\mathcal{G}_1(\mathcal{A}_{U_1}, \boldsymbol{b}_{U_1}, c_{U_1}) \mathcal{G}_2(\mathcal{A}_{U_2}, \boldsymbol{b}_{U_2}, c_{U_2}) = e^{?} \mathcal{G}_f(\mathcal{A}_{U_f}, \boldsymbol{b}_{U_f}, c_{U_f}). \tag{6.3.4}$$

The question now is how to find a phase φ containing inside c_{U_f} as a function of $A_{U_1}, \boldsymbol{b}_{U_1}, c_{U_1}$ and $A_{U_2}, \boldsymbol{b}_{U_2}, c_{U_2}$.

6.3.1 How to compute the global phase

Completeness of the coherent states So we first extend the Husimi Q function of the two successive M-mode Gaussian operators $\mathcal{G}_1, \mathcal{G}_2$ by inserting an identity between them and using the completeness of the coherent states, and we combine them into two Husimi Q functions:

$$\langle \boldsymbol{\beta}^* | \mathcal{G}_1 \mathcal{G}_2 | \boldsymbol{\beta'} \rangle = \langle \boldsymbol{\beta}^* | \mathcal{G}_1 I \mathcal{G}_2 | \boldsymbol{\beta'} \rangle = \frac{1}{\pi^M} \int_{-\infty}^{+\infty} d^{2M}\boldsymbol{\alpha} \langle \boldsymbol{\beta}^* | \mathcal{G}_1 | \boldsymbol{\alpha} \rangle \langle \boldsymbol{\alpha} | \mathcal{G}_2 | \boldsymbol{\beta'} \rangle, \tag{6.3.5}$$

where $d^{2M}\boldsymbol{\alpha} = d\mathrm{Re}(\boldsymbol{\alpha}) d\mathrm{Im}(\boldsymbol{\alpha})$.

And then we extend the expressions of $\langle \boldsymbol{\beta}^* | \mathcal{G}_1 | \boldsymbol{\alpha} \rangle, \langle \boldsymbol{\alpha} | \mathcal{G}_2 | \boldsymbol{\beta'} \rangle$:

$$\frac{1}{\pi^M} \int_{-\infty}^{+\infty} d^{2M}\boldsymbol{\alpha} \exp\left(-\tfrac{1}{2}\left[||\boldsymbol{\beta}||^2 + 2||\boldsymbol{\alpha}||^2 + ||\boldsymbol{\beta'}||^2\right]\right) c_{U_1} c_{U_2} \exp\left(\boldsymbol{b}_{U_1}^T \boldsymbol{\nu}_1 + \frac{1}{2}\boldsymbol{\nu}_1^T \boldsymbol{A}_{U_1} \boldsymbol{\nu_1} + \boldsymbol{b}_{U_2}^T \boldsymbol{\nu}_2 + \frac{1}{2}\boldsymbol{\nu}_2^T \boldsymbol{A}_{U_2} \boldsymbol{\nu_2}\right), \tag{6.3.6}$$

where

$$\boldsymbol{\nu}_{\mathbf{1}}^T = [\boldsymbol{\beta}, \boldsymbol{\alpha}], \tag{6.3.7}$$

$$\boldsymbol{\nu}_{\mathbf{2}}^T = [\boldsymbol{\alpha^*}, \boldsymbol{\beta'}]. \tag{6.3.8}$$

Integral $\boldsymbol{\alpha}$ In order to integrate the $\boldsymbol{\alpha}$ with its real and imaginary parts, we introduce the following vectors because the integral of the multi-dimensional Gaussian expression is based on the real parameters:

$$\boldsymbol{x}_r^T = [\mathrm{Re}(\boldsymbol{\alpha}), \mathrm{Im}(\boldsymbol{\alpha}), \mathrm{Re}(\boldsymbol{\beta}), \mathrm{Im}(\boldsymbol{\beta}), \mathrm{Re}(\boldsymbol{\beta'}), \mathrm{Im}(\boldsymbol{\beta'})], \tag{6.3.9}$$

$$\boldsymbol{x}_c^T = [\boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\beta}, \boldsymbol{\beta}^*, \boldsymbol{\beta'}, \boldsymbol{\beta'}^*], \tag{6.3.10}$$

and it is clear to transfer from $\boldsymbol{x}_r$ to $\boldsymbol{x}_c$ with the matrix $\boldsymbol{M}_0$:

$$\boldsymbol{x}_c = (\boldsymbol{M}_0 \oplus \boldsymbol{M}_0 \oplus \boldsymbol{M}_0) \boldsymbol{x}_r, \tag{6.3.11}$$

where

$$\boldsymbol{M}_0 = \begin{bmatrix} \mathbb{1} & i\mathbb{1} \\ \mathbb{1} & -i\mathbb{1} \end{bmatrix}. \tag{6.3.12}$$

Next step, we replace all elements in Eq. (6.3.6) in terms of $\boldsymbol{x}_c^T$.

So we have:

$$\begin{aligned} ||\boldsymbol{\alpha}||^2 &= \frac{1}{2}\boldsymbol{x}_c^T \boldsymbol{M}_2 \boldsymbol{x}_c, \\ \boldsymbol{\nu}_1 &= \boldsymbol{M}_3 \boldsymbol{x}_c, \\ \boldsymbol{\nu}_2 &= \boldsymbol{M}_4 \boldsymbol{x}_c, \end{aligned} \tag{6.3.13}$$

where

$$\boldsymbol{M}_2 = \begin{bmatrix} 0 & \mathbb{1} & 0 & 0 & 0 & 0 \\ \mathbb{1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{6.3.14}$$

$$\boldsymbol{M}_3 = \begin{bmatrix} 0 & 0 & \mathbb{1} & 0 & 0 & 0 \\ \mathbb{1} & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{6.3.15}$$

$$\boldsymbol{M}_4 = \begin{bmatrix} 0 & \mathbb{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbb{1} & 0 \end{bmatrix}. \tag{6.3.16}$$

We can now rewrite Eq. (6.3.6) as a function of $\boldsymbol{x}_c$:

$$\begin{aligned} \langle \boldsymbol{\beta}^* | \mathcal{G}_1 I \mathcal{G}_2 | \boldsymbol{\beta'} \rangle =& \frac{1}{\pi^M} \int_{-\infty}^{+\infty} d^{2M}\boldsymbol{\alpha} \exp\left(-\tfrac{1}{2}\left[||\boldsymbol{\beta}||^2 + ||\boldsymbol{\beta'}||^2\right]\right) c_{U_1} c_{U_2} \\ & \exp\left[(\boldsymbol{b}_{U_1}^T \boldsymbol{M}_3 + \boldsymbol{b}_{U_2}^T \boldsymbol{M}_4)\boldsymbol{x}_c + \frac{1}{2}\boldsymbol{x}_c^T (\boldsymbol{M}_3^T \boldsymbol{A}_{U_1} \boldsymbol{M}_3 + \boldsymbol{M}_4^T \boldsymbol{A}_{U_2} \boldsymbol{M}_4 - \boldsymbol{M}_2)\boldsymbol{x}_c\right] \\ =& \frac{1}{\pi^M} \int_{-\infty}^{+\infty} d^{2M}\boldsymbol{\alpha} \exp\left(-\tfrac{1}{2}\left[||\boldsymbol{\beta}||^2 + ||\boldsymbol{\beta'}||^2\right]\right) c_{U_1} c_{U_2} \exp\left(\boldsymbol{\mathcal{B}}^T \boldsymbol{x}_c + \frac{1}{2}\boldsymbol{x}_c^T \boldsymbol{\mathcal{A}} \boldsymbol{x}_c\right), \end{aligned} \tag{6.3.17}$$

where

$$\begin{aligned} \boldsymbol{\mathcal{B}}^T &= \boldsymbol{b}_{U_1}^T \boldsymbol{M}_3 + \boldsymbol{b}_{U_2}^T \boldsymbol{M}_4 \\ &= \left[\begin{array}{cc|ccc} \boldsymbol{d}_1^T & \boldsymbol{c}_2^T & \boldsymbol{c}_1^T & 0 & \boldsymbol{d}_2^T & 0 \end{array}\right] \\ &= \left[\boldsymbol{\mathcal{B}}_l^T, \boldsymbol{\mathcal{B}}_r^T\right], \end{aligned} \tag{6.3.18}$$

$$\begin{aligned} \boldsymbol{\mathcal{A}} &= \boldsymbol{M}_3^T \boldsymbol{A}_{U_1} \boldsymbol{M}_3 + \boldsymbol{M}_4^T \boldsymbol{A}_{U_2} \boldsymbol{M}_4 - \boldsymbol{M}_2 \\ &= \left[\begin{array}{cc|ccc|c} \boldsymbol{D}_1 & -\mathbb{1} & \boldsymbol{C}_1^T & 0 & 0 & 0 \\ -\mathbb{1} & \boldsymbol{B}_2 & 0 & 0 & \boldsymbol{C}_2 & 0 \\ \hline \boldsymbol{C}_1 & 0 & \boldsymbol{B}_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \boldsymbol{C}_2^T & 0 & 0 & \boldsymbol{D}_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] = \left[\begin{array}{c|c} \boldsymbol{\mathcal{A}}_1 & \boldsymbol{\mathcal{A}}_2 \\ \hline \boldsymbol{\mathcal{A}}_3 & \boldsymbol{\mathcal{A}}_4 \end{array}\right], \end{aligned} \tag{6.3.19}$$

and we write $\boldsymbol{b}_{U_i}$ and $\boldsymbol{A}_{U_i}$ in block:

$$\boldsymbol{b}_{U_i}^T = \left[\boldsymbol{c}_i^T, \boldsymbol{d}_i^T\right], \tag{6.3.20}$$

$$\boldsymbol{A}_{U_i} = \left[\begin{array}{c|c} \boldsymbol{B}_i & \boldsymbol{C}_i \\ \hline \boldsymbol{C}_i^T & \boldsymbol{D}_i \end{array}\right]. \tag{6.3.21}$$

It is obvious that $\boldsymbol{\mathcal{A}}$, $\boldsymbol{\mathcal{A}}_1$ and $\boldsymbol{\mathcal{A}}_4$ are all symmetric because of the symmetric of $\boldsymbol{A}_{U_1}$ and $\boldsymbol{A}_{U_2}$.

Now we want to separate out the components with $\boldsymbol{\alpha}$ from $\exp\left(\boldsymbol{\mathcal{B}}^T\boldsymbol{x}_c + \frac{1}{2}\boldsymbol{x}_c^T\boldsymbol{\mathcal{A}}\boldsymbol{x}_c\right)$ in order to calculate the integral. We first redefine the vector as:

$$\boldsymbol{x}_r^T = \left[\begin{array}{cc|cccc} \mathrm{Re}(\boldsymbol{\alpha}) & \mathrm{Im}(\boldsymbol{\alpha}) & \mathrm{Re}(\boldsymbol{\beta}) & \mathrm{Im}(\boldsymbol{\beta}) & \mathrm{Re}(\boldsymbol{\beta'}) & \mathrm{Im}(\boldsymbol{\beta'}) \end{array}\right] = [\boldsymbol{x}_{rl}, \boldsymbol{x}_{rr}]^T, \tag{6.3.22}$$

where $\boldsymbol{x}_{rl}$ is the term related to $\boldsymbol{\alpha}$. So that we have:

$$\boldsymbol{x}_c = (\boldsymbol{M}_0 \oplus \boldsymbol{M}_0 \oplus \boldsymbol{M}_0)\boldsymbol{x}_r = \begin{bmatrix} \boldsymbol{M}_0\boldsymbol{x}_{rl} \\ (\boldsymbol{M}_0 \oplus \boldsymbol{M}_0)\boldsymbol{x}_{rr} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_{cl} \\ \boldsymbol{x}_{cr} \end{bmatrix}. \tag{6.3.23}$$

Then we rewrite Eq. (6.3.17) with $\boldsymbol{x}_{cl}$ and $\boldsymbol{x}_{cr}$:

$$\begin{aligned}\langle\boldsymbol{\beta}^*|\mathcal{G}_1 I \mathcal{G}_2|\boldsymbol{\beta'}\rangle =& \frac{1}{\pi^M}\int_{-\infty}^{+\infty} d^{2M}\boldsymbol{\alpha}\exp\left(-\tfrac{1}{2}\left[||\boldsymbol{\beta}||^2 + ||\boldsymbol{\beta'}||^2\right]\right) c_{U_1}c_{U_2} \\ & \exp\left(\boldsymbol{\mathcal{B}}_r^T\boldsymbol{x}_{cr} + \boldsymbol{\mathcal{B}}_l^T\boldsymbol{x}_{cl} + \frac{1}{2}\left(\boldsymbol{x}_{cl}^T\boldsymbol{\mathcal{A}}_1\boldsymbol{x}_{cl} + \boldsymbol{x}_{cl}^T\boldsymbol{\mathcal{A}}_2\boldsymbol{x}_{cr} + \boldsymbol{x}_{cr}^T\boldsymbol{\mathcal{A}}_3\boldsymbol{x}_{cl} + \boldsymbol{x}_{cr}^T\boldsymbol{\mathcal{A}}_4\boldsymbol{x}_{cr}\right)\right),\end{aligned} \tag{6.3.24}$$

because of the symmetry of $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{x}_{cl} = \boldsymbol{M}_0\boldsymbol{x}_{rl}$, we have:

$$\begin{aligned}\langle\boldsymbol{\beta}^*|\mathcal{G}_1 I \mathcal{G}_2|\boldsymbol{\beta'}\rangle =& \exp\left(-\tfrac{1}{2}\left[||\boldsymbol{\beta}||^2 + ||\boldsymbol{\beta'}||^2\right]\right) c_{U_1}c_{U_2}\exp\left(\boldsymbol{\mathcal{B}}_r^T\boldsymbol{x}_{cr} + \frac{1}{2}\boldsymbol{x}_{cr}^T\boldsymbol{\mathcal{A}}_4\boldsymbol{x}_{cr}\right) \\ & * \frac{1}{\pi^M}\int_{-\infty}^{+\infty} d^{2M}\boldsymbol{x_{rl}}\exp\left(\left(\boldsymbol{\mathcal{B}}_l^T + \boldsymbol{x}_{cr}^T\boldsymbol{\mathcal{A}}_3\right)\boldsymbol{M}_0\boldsymbol{x}_{rl} + \frac{1}{2}\boldsymbol{x}_{rl}^T\boldsymbol{M}_0^T\boldsymbol{\mathcal{A}}_1\boldsymbol{M}_0\boldsymbol{x}_{rl}\right).\end{aligned} \tag{6.3.25}$$

We can integrate the multi-dimensional Gaussian expression with $\boldsymbol{x}_{rl} = [\mathrm{Re}(\boldsymbol{\alpha}), \mathrm{Im}(\boldsymbol{\alpha})]$:

$$\exp\left(-\tfrac{1}{2}\left[||\boldsymbol{\beta}||^2 + ||\boldsymbol{\beta'}||^2\right]\right) c_{U_1}c_{U_2}\exp\left(\boldsymbol{\mathcal{B}}_r^T\boldsymbol{x}_{cr} + \frac{1}{2}\boldsymbol{x}_{cr}^T\boldsymbol{\mathcal{A}}_4\boldsymbol{x}_{cr}\right) * \frac{1}{\pi^M}\frac{(2\pi)^M}{\sqrt{\det(\boldsymbol{\mathcal{A}'})}}\exp\left(-\frac{1}{2}\boldsymbol{\mathcal{B}'}^T\left(\boldsymbol{\mathcal{A}'}\right)^{-1}\boldsymbol{\mathcal{B}'}\right), \tag{6.3.26}$$

where

$$\boldsymbol{\mathcal{B}'}^T = \left(\boldsymbol{\mathcal{B}}_l^T + \boldsymbol{x}_{cr}^T\boldsymbol{\mathcal{A}}_3\right)\boldsymbol{M}_0, \tag{6.3.27}$$

$$\boldsymbol{\mathcal{A}'} = \boldsymbol{M}_0^T\boldsymbol{\mathcal{A}}_1\boldsymbol{M}_0. \tag{6.3.28}$$

Derivition of $c_{U_f}, \boldsymbol{b}_{U_f}$ and $\boldsymbol{A}_{U_f}$ So we now write the Husimi Q function of two Gaussian unitaries as:

$$\begin{aligned}\langle\boldsymbol{\beta}^*|\mathcal{G}_1 I \mathcal{G}_2|\boldsymbol{\beta'}\rangle =& \exp\left(-\tfrac{1}{2}\left[||\boldsymbol{\beta}||^2 + ||\boldsymbol{\beta'}||^2\right]\right)\frac{2^M c_{U_1}c_{U_2}}{\sqrt{\det(\boldsymbol{\mathcal{A}'})}} \\ & \exp\left(\boldsymbol{\mathcal{B}}_r^T\boldsymbol{x}_{cr} + \frac{1}{2}\boldsymbol{x}_{cr}^T\boldsymbol{\mathcal{A}}_4\boldsymbol{x}_{cr} - \frac{1}{2}\left(\boldsymbol{\mathcal{B}}_l^T + \boldsymbol{x}_{cr}^T\boldsymbol{\mathcal{A}}_3\right)\boldsymbol{\mathcal{A}}_1^{-1}\left(\boldsymbol{\mathcal{B}}_l^T + \boldsymbol{x}_{cr}^T\boldsymbol{\mathcal{A}}_3\right)^T\right) \\ =& \exp\left(-\tfrac{1}{2}\left[||\boldsymbol{\beta}||^2 + ||\boldsymbol{\beta'}||^2\right]\right)\frac{c_{U_1}c_{U_2}}{\sqrt{(-1)^M\det(\boldsymbol{\mathcal{A}}_1)}} \\ & \exp\left(-\frac{1}{2}\boldsymbol{\mathcal{B}}_l^T\boldsymbol{\mathcal{A}}_1^{-1}\boldsymbol{\mathcal{B}}_l + \left(\boldsymbol{\mathcal{B}}_r^T - \boldsymbol{\mathcal{B}}_l^T\boldsymbol{\mathcal{A}}_1^{-1}\boldsymbol{\mathcal{A}}_3^T\right)\boldsymbol{x}_{cr} + \frac{1}{2}\boldsymbol{x}_{cr}^T\left(\boldsymbol{\mathcal{A}}_4 - \boldsymbol{\mathcal{A}}_3\boldsymbol{\mathcal{A}}_1^{-1}\boldsymbol{\mathcal{A}}_3^T\right)\boldsymbol{x}_{cr}\right).\end{aligned} \tag{6.3.29}$$

We need to notice that, because of the integral, we are working with $\boldsymbol{x}_{cr}^T = (\boldsymbol{M}_0 \oplus \boldsymbol{M}_0)[\mathrm{Re}(\boldsymbol{\beta}), \mathrm{Im}(\boldsymbol{\beta}), \mathrm{Re}(\boldsymbol{\beta'}), \mathrm{Im}(\boldsymbol{\beta'})]$,

however, we need to go back to the vector with complex expressions $\boldsymbol{\nu'}^T = [\boldsymbol{\beta}, \boldsymbol{\beta'}, \boldsymbol{\beta}^*, \boldsymbol{\beta'}^*]$. We have:

$$\boldsymbol{\nu'} = \begin{bmatrix} \mathbb{1} & i\mathbb{1} & 0 & 0 \\ 0 & 0 & \mathbb{1} & i\mathbb{1} \\ \mathbb{1} & -i\mathbb{1} & 0 & 0 \\ 0 & 0 & \mathbb{1} & -i\mathbb{1} \end{bmatrix} \boldsymbol{x}_{rr} = \boldsymbol{M}_5 \boldsymbol{x}_{rr} = \boldsymbol{M}_5 (\boldsymbol{M}_0 \oplus \boldsymbol{M}_0)^{-1} \boldsymbol{x}_{cr}. \tag{6.3.30}$$

Then we rewrite the expression with $\boldsymbol{\nu'}$:

$$\langle \boldsymbol{\beta}^* | \mathcal{G}_1 \mathcal{G}_2 | \boldsymbol{\beta'} \rangle = \exp\left(-\tfrac{1}{2}\left[||\boldsymbol{\beta}||^2 + ||\boldsymbol{\beta'}||^2\right]\right) c_{U_f} \exp(\boldsymbol{b'_{U_f}}^T \boldsymbol{\nu'} + \frac{1}{2} \boldsymbol{\nu'}^T \boldsymbol{A'_{U_f}} \boldsymbol{\nu'}), \tag{6.3.31}$$

where

$$\begin{aligned} c_{U_f} &= \frac{c_{U_1} c_{U_2}}{\sqrt{(-1)^M \det(\boldsymbol{\mathcal{A}}_1)}} \exp(-\frac{1}{2} \boldsymbol{\mathcal{B}}_l^T \boldsymbol{\mathcal{A}}_1^{-1} \boldsymbol{\mathcal{B}}_l), \\ \boldsymbol{b'_{U_f}}^T &= (\boldsymbol{\mathcal{B}}_r^T - \boldsymbol{\mathcal{B}}_l^T \boldsymbol{\mathcal{A}}_1^{-1} \boldsymbol{\mathcal{A}}_3^T)(\boldsymbol{M}_0 \oplus \boldsymbol{M}_0) \boldsymbol{M}_5^{-1}, \\ \boldsymbol{A'_{U_f}} &= (\boldsymbol{M}_5^{-1})^T (\boldsymbol{M}_0 \oplus \boldsymbol{M}_0)^T (\boldsymbol{\mathcal{A}}_4 - \boldsymbol{\mathcal{A}}_3 \boldsymbol{\mathcal{A}}_1^{-1} \boldsymbol{\mathcal{A}}_3^T)(\boldsymbol{M}_0 \oplus \boldsymbol{M}_0) \boldsymbol{M}_5^{-1}. \end{aligned} \tag{6.3.32}$$

In order to get the explicit formula, we compute:

$$(\boldsymbol{M}_0 \oplus \boldsymbol{M}_0) \boldsymbol{M}_5^{-1} = \begin{bmatrix} \mathbb{1} & 0 & 0 & 0 \\ 0 & 0 & \mathbb{1} & 0 \\ 0 & \mathbb{1} & 0 & 0 \\ 0 & 0 & 0 & \mathbb{1} \end{bmatrix}, \tag{6.3.33}$$

and by using the Schur complement, since $\boldsymbol{\mathcal{A}}_1$ is symmetric, $\boldsymbol{D}_1$ and $\boldsymbol{B}_2$ are both symmetric, we can define the inverse of $\boldsymbol{\mathcal{A}}_1$:

$$\boldsymbol{\mathcal{A}}_1^{-1} = \begin{bmatrix} \boldsymbol{D}_1 \boldsymbol{\mathcal{X}}^{-1} & \boldsymbol{\mathcal{X}}^{-1} \\ \left(\boldsymbol{\mathcal{X}}^T\right)^{-1} & \boldsymbol{D}_1 \left(\boldsymbol{\mathcal{X}}^T\right)^{-1} \end{bmatrix}, \tag{6.3.34}$$

where

$$\boldsymbol{\mathcal{X}} = \boldsymbol{B}_2 \boldsymbol{D}_1 - \mathbb{1}. \tag{6.3.35}$$

Then we can get:

$$c_{U_f} = \frac{c_{U_1} c_{U_2}}{\sqrt{(-1)^M \det(\boldsymbol{\mathcal{A}}_1)}} \exp\left[-\frac{1}{2}\left(\boldsymbol{d}_1^T \boldsymbol{D}_1 \boldsymbol{\mathcal{X}}^{-1} \boldsymbol{d}_1 + \boldsymbol{c}_2^T \left(\boldsymbol{\mathcal{X}}^{\boldsymbol{T}}\right)^{-1} \boldsymbol{d}_1 + \boldsymbol{d}_1^T \boldsymbol{\mathcal{X}}^{-1} \boldsymbol{c}_2 + \boldsymbol{c}_2^T \boldsymbol{D}_1 \left(\boldsymbol{\mathcal{X}}^{\boldsymbol{T}}\right)^{-1} \boldsymbol{c}_2\right)\right], \tag{6.3.36}$$

$$\boldsymbol{b}'_{U_f} = \begin{bmatrix} \boldsymbol{c}_1 - \boldsymbol{C}_1 \boldsymbol{D}_1 \boldsymbol{\mathcal{X}}^{-1} \boldsymbol{d}_1 - \boldsymbol{C}_1 \left(\boldsymbol{\mathcal{X}}^{\boldsymbol{T}}\right)^{-1} \boldsymbol{c}_2 \\ \boldsymbol{d}_2 - \boldsymbol{C}_2^T \boldsymbol{\mathcal{X}}^{-1} \boldsymbol{d}_1 - \boldsymbol{C}_2^T \boldsymbol{D}_1 \left(\boldsymbol{\mathcal{X}}^{\boldsymbol{T}}\right)^{-1} \boldsymbol{c}_2 \\ 0_M \\ 0_M \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_{U_f} \\ \boldsymbol{0}_{2M} \end{bmatrix}, \tag{6.3.37}$$

$$\boldsymbol{A}'_{U_f} = \begin{bmatrix} \boldsymbol{B}_1 - \boldsymbol{C}_1 \boldsymbol{D}_1 \boldsymbol{\mathcal{X}}^{-1} \boldsymbol{C}_1^T & -\boldsymbol{C}_1 \boldsymbol{\mathcal{X}}^{-1} \boldsymbol{C}_2 & 0_M & 0_M \\ -\boldsymbol{C}_2^T \left(\boldsymbol{\mathcal{X}}^{\boldsymbol{T}}\right)^{-1} \boldsymbol{B}_1^T & \boldsymbol{D}_2 - \boldsymbol{C}_2^T \boldsymbol{D}_1 \left(\boldsymbol{\mathcal{X}}^{\boldsymbol{T}}\right)^{-1} \boldsymbol{C}_2 & 0_M & 0_M \\ 0_M & 0_M & 0_M & 0_M \\ 0_M & 0_M & 0_M & 0_M \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{U_f} & 0_{2M} \\ 0_{2M} & 0_{2M} \end{bmatrix}. \tag{6.3.38}$$

So that finally, we can write:

$$\langle \boldsymbol{\beta}^* | \mathcal{G}_1 \mathcal{G}_2 | \boldsymbol{\beta'} \rangle = \exp\left(-\tfrac{1}{2}\left[||\boldsymbol{\beta}||^2 + ||\boldsymbol{\beta'}||^2\right]\right) c_{U_f} \exp\left(\boldsymbol{b}_{U_f}^T \boldsymbol{\nu} + \frac{1}{2} \boldsymbol{\nu}^T \boldsymbol{A}_{U_f} \boldsymbol{\nu}\right), \tag{6.3.39}$$

c_{U_f} contains the global phase we want.

Final result We can now write down the explicit formula:

$$\boldsymbol{A}_{U_f} = \begin{bmatrix} \boldsymbol{B}_1 - \boldsymbol{C}_1 \boldsymbol{D}_1 \boldsymbol{\mathcal{X}}^{-1} \boldsymbol{C}_1^T & -\boldsymbol{C}_1 \boldsymbol{\mathcal{X}}^{-1} \boldsymbol{C}_2 \\ -\boldsymbol{C}_2^T \left(\boldsymbol{\mathcal{X}}^{\boldsymbol{T}}\right)^{-1} \boldsymbol{C}_1^T & \boldsymbol{D}_2 - \boldsymbol{C}_2^T \boldsymbol{D}_1 \left(\boldsymbol{\mathcal{X}}^{\boldsymbol{T}}\right)^{-1} \boldsymbol{C}_2 \end{bmatrix}, \tag{6.3.40}$$

$$\boldsymbol{b}_{U_f} = \begin{bmatrix} \boldsymbol{c}_1 - \boldsymbol{C}_1 \boldsymbol{D}_1 \boldsymbol{\mathcal{X}}^{-1} \boldsymbol{d}_1 - \boldsymbol{C}_1 \left(\boldsymbol{\mathcal{X}}^{\boldsymbol{T}}\right)^{-1} \boldsymbol{c}_2 \\ \boldsymbol{d}_2 - \boldsymbol{C}_2^T \boldsymbol{\mathcal{X}}^{-1} \boldsymbol{d}_1 - \boldsymbol{C}_2^T \boldsymbol{D}_1 \left(\boldsymbol{\mathcal{X}}^{\boldsymbol{T}}\right)^{-1} \boldsymbol{c}_2 \end{bmatrix}, \tag{6.3.41}$$

$$c_{U_f} = \frac{c_{U_1} c_{U_2}}{\sqrt{(-1)^M \det(\boldsymbol{\mathcal{A}}_1)}} \exp\Big[-\frac{1}{2}\Big(\boldsymbol{d}_1^T \boldsymbol{D}_1 \boldsymbol{\mathcal{X}}^{-1} \boldsymbol{d}_1 + \boldsymbol{c}_2^T \left(\boldsymbol{\mathcal{X}}^{\boldsymbol{T}}\right)^{-1} \boldsymbol{d}_1 + \boldsymbol{d}_1^T \boldsymbol{\mathcal{X}}^{-1} \boldsymbol{c}_2 + \boldsymbol{c}_2^T \boldsymbol{D}_1 \left(\boldsymbol{\mathcal{X}}^{\boldsymbol{T}}\right)^{-1} \boldsymbol{c}_2\Big)\Big], \tag{6.3.42}$$

where $\boldsymbol{b}_{U_i}^T$ and $\boldsymbol{A}_{U_i}$ are written in block form:

$$\boldsymbol{b}_{U_i}^T = \left[\boldsymbol{c}_i^T, \boldsymbol{d}_i^T\right], \tag{6.3.43}$$

$$\boldsymbol{A}_{U_i} = \left[\begin{array}{c|c} \boldsymbol{B}_i & \boldsymbol{C}_i \\ \hline \boldsymbol{C}_i^T & \boldsymbol{D}_i \end{array}\right], \tag{6.3.44}$$

And we define a new matrix $\boldsymbol{\mathcal{X}}$:

$$\boldsymbol{\mathcal{X}} = \boldsymbol{B}_2 \boldsymbol{D}_1 - \mathbb{1}. \tag{6.3.45}$$

Eq. (6.3.42) is the term of *global phase* for the composite Gaussian operators.

6.3.2 Some examples

As examples, we show the composition of two single-mode displacements and the composition of two single-mode squeezers.

For displacement operators $\mathcal{D}(\alpha), \mathcal{D}(\beta)$, we have

$$\det(\boldsymbol{D}_1\boldsymbol{B}_2 - \mathbb{1}) = -1, \quad \boldsymbol{\mathcal{X}} = -1. \tag{6.3.46}$$

So we obtain the global phase:

$$c_{U_f} = c_{U_\alpha} c_{U_\beta} \exp\left(-\alpha^*\beta\right) \tag{6.3.47}$$

$$= c_{U_{(\alpha+\beta)}} \exp\left(\tfrac{1}{2}\alpha\beta^* - \tfrac{1}{2}\beta\alpha^*\right), \tag{6.3.48}$$

where the global phase is the term $\varphi = \frac{i}{2}\left(\alpha\beta^* - \beta\alpha^*\right)$, which corresponds to Eq. (6.3.1).

For two squeezers $\mathcal{S}(\zeta_1), \mathcal{S}(\zeta_2)$, since $\boldsymbol{b}_U$ is zero, we have

$$\det(\boldsymbol{D}_1\boldsymbol{B}_2 - \mathbb{1}) = -1 - e^{i(\delta_2-\delta_1)} \tanh r_1 \tanh r_2, \tag{6.3.49}$$

and in turn, we get

$$c_{U_f} = \frac{c_{U_1} c_{U_2}}{\sqrt{-\det(\boldsymbol{D}_1\boldsymbol{B}_2 - \mathbb{1})}} \tag{6.3.50}$$

$$= \frac{\sqrt{\operatorname{sech} r_1 \operatorname{sech} r_2}}{\sqrt{1 + e^{i(\delta_2-\delta_1)} \tanh r_1 \tanh r_2}}, \tag{6.3.51}$$

which coincides with the results from Refs [141, 142].

6.4 Conclusion

In this chapter, a single recurrence relation for all Gaussian objects is defined by a complex symmetric matrix $\boldsymbol{A}$, a complex vector $\boldsymbol{b}$, and a scalar c, which are used to construct their Fock representations. Our work bridges the phase space representation and Fock space representation for all Gaussian objects, allowing us to include more elements in Fock space representation in numerical simulations (such as the PNR detectors) and the non-Gaussian gates (Kerr gates).

In addition, the explicit global phase formula is also derived, following the composition rule of Gaussian operators. This inspires the simulation of some non-Gaussian states, which can be written as linear combinations of Gaussian states.

This linear recurrence relation is recursive and differentiable, so it is simple to implement in the simulation and calculate its gradient of the Fock amplitudes directly, which will be explained in the following chapter.

Chapter 7

Optimization of parameterized quantum optical circuits

7.1 Introduction

In our optimization, the gradient descent is used to update the parameters $\boldsymbol{\theta}$ in PQCs. The cost function L is defined according to the different tasks, and concrete examples will be detailed in the next chapter.

The calculation of gradients is introduced first in this chapter, together with how the differentiability of our recurrence relation facilitates the gradient calculations. Two methods are proposed to use the differentiability: the chain rule and the generating function.

Next, we illustrate Euclidean and Riemannian optimizations of Gaussian objects in PQCs. These two types of optimization differ in how we treat the Gaussian operators:

- Euclidean optimization: we decompose the Gaussian operator into fundamental optical components and optimize each of them.
- Riemannian optimization: we take all the Gaussian operators as one global object first, optimize it and then decompose it into fundamental optical components. Note that the special case of the Gaussian operator is the interferometer. The unitary optimization is derived for the interferometer as well.

If the PQC is small, Euclidean optimization is the best choice to get the values for each component, while if the PQC is large, optimizing each parameter is redundant, so it is better to take them as a global Gaussian operator and use the Riemannian optimization.

Moreover, regarding the complex parameters in our quantum optical circuits, we propose the complex natural gradient algorithm to adapt the natural gradient with complex parameters instead of separating the complex parameters into the real and complex parts, which speeds up the gradient calculations. In addition, our adaptation version of the natural gradient is a holomorphic function that is differentiable and easy to use in different analytical techniques.

7.2 Gradient calculation with AD and differentiability of recurrence relation for Gaussian objects

In this thesis, we do numerical simulations to construct quantum optical circuits and optimize them automatically with the help of the gradient descent algorithm. As introduced in chapter 4, we use gradient descent as our optimization method. Thus, we need to find out how to calculate the gradient of operators with respect to their parameters in quantum optical circuits.

To avoid lots of unnecessary gradient calculations manually, we want to use AD. TensorFlow (TF), which is an open-source library to construct the machine learning model and train it, uses the AD technique. And

AD plays an important role in our numerical simulations to compute the gradients automatically. So we will first introduce how to calculate the gradient in mathematical forms in section 7.2.1 and how we realize it with TF in section 7.2.2.

Since we use the recurrence relation to simulate the Gaussian objects inside the circuits, we then discuss the differentiability of our recurrence relation in section 7.2.3. Thanks to this differentiability, we can compute the gradients directly.

7.2.1 Calculation of the gradient for quantum optical circuits

As explained in chapter 4, the update rule of each parameter for the gradient descent algorithm depends on the gradient calculation.

Let's take an example into consideration.

If we concatenate a squeezer $\mathcal{S}$ on the first mode, then a beam-splitter $\mathcal{B}$ on both the first and second modes, and after that, a rotation operator $\mathcal{R}$ on the second mode, we would have a circuit like this:

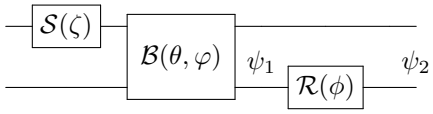


Figure 7.1: A quantum optical circuit (states move through the circuit from left to right).

In Fig. 7.1, this operator $\mathcal{G}$ is characterized by a vector of parameters $\boldsymbol{x} = [\zeta, \theta, \varphi, \phi]$, and we define the cost function L with the output of the circuit.

The chain rule to write the gradient with respect to the real parameter θ would be:

$$\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial \psi_2}\frac{\partial \psi_2}{\partial \psi_1}\frac{\partial \psi_1}{\partial \theta}. \tag{7.2.1}$$

first two terms are called the upstream, which comes from the backward pass of TF's calculation. The upstream can be obtained directly once the circuit is built. The last term $\frac{\partial \psi_1}{\partial \theta}$ is the derivative provided by our recurrence relation. All details about how to get this derivative are shown in section 7.2.3. We only need to use the tf.custom_gradient in TF to define this term with our equations.

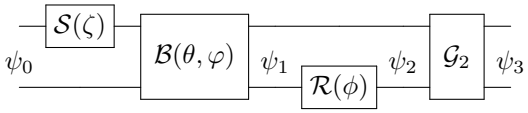


Figure 7.2: A quantum optical circuit (states move through the circuit from left to right).

If we concatenate one more operator $\mathcal{G}_2$ as shown in Fig. 7.2, since they are all independent variables, the chain rule of the parameter θ looks like:

$$\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial \psi_3}\frac{\partial \psi_3}{\partial \psi_2}\left(\frac{\partial \psi_2}{\partial \psi_1}\frac{\partial \psi_1}{\partial \theta}\right) = \frac{\partial L}{\partial \psi_3}\frac{\partial \psi_3}{\partial \psi_2}\left(\frac{\partial \psi_2}{\partial \theta}\right) \tag{7.2.2}$$

$$= \frac{\partial L}{\partial \psi_3}\frac{\partial(\mathcal{G}_2\psi_2)}{\partial \psi_2}\frac{\partial(\mathcal{G}\psi_0)}{\partial \theta} = \frac{\partial L}{\partial \psi_2}\mathcal{G}_2\left(\frac{\partial \psi_2}{\partial \theta}\right), \tag{7.2.3}$$

the operator $\mathcal{G}_2$ is integrated into the upstream part and the derivative part $\frac{\partial \psi_2}{\partial \theta}$ is always the same according to the same operator.

Therefore, if we want to update the variable vector $\boldsymbol{x}$, by using the upstream provided by TF, we only need to compute the Jacobian matrix for the Gaussian operator according to each variable and execute the JVP.

7.2.2 Library implementation based on TF

TF is an open-source library used to develop and train Machine Learning models. One can either create models by plugging building blocks together or create a model and write the forward pass.

TF implements AD using symbol-to-symbol derivatives. In this case, gradients are computed by constructing a graph with all the operations. This graph is built by listing the successive operations in the forward pass. Then, in the backward pass, the calculation of derivatives traverses the list of operations and uses the chain rule to connect each operation.

TF functions

TF provides the *tf.GradientTape* API for AD. No matter a function, or a model, we can get the gradient by putting the function or the model inside the "tape" and obtaining the gradient directly. One example is shown in Fig. 7.3.

```
x = tf.Variable(3.0)

with tf.GradientTape() as tape:
  y = x**2

dy_dx = tape.gradient(y, x)
```

Figure 7.3: Example code for GradienTape function of TF.

As we explained, we have two parts to calculate the gradient: the upstream part and the derivative to get the gradient of the parameter inside the optical quantum circuits. The derivative can be realized by defining the custom gradient function *tf.custom_gradient*. One example code of a custom gradient function is shown in Fig. 7.4.

```
@tf.custom_gradient
def log1pexp(x):
  e = tf.exp(x)
  def grad(dy):
    return dy * (1 - 1 / (1 + e))
  return tf.math.log(1 + e), grad
```

Figure 7.4: Example code for CustomGradient inside TF.

The variable dy is defined as the upstream gradient. By chain rule, we know that

$$\frac{dy}{dx} = \frac{dy}{dx_0}\frac{dx_0}{dx_1}...\frac{dx_i}{dx_{i+1}}...\frac{dx_n}{dx}, \tag{7.2.4}$$

where $\frac{dx_i}{dx_{i+1}}$ is the gradient of current function and it is defined inside the function with the name grad. Here we know that dy is $\frac{dx_{i+1}}{dx_{i+2}}...\frac{dx_n}{dx}$. The upstream gradient multiplied by the current gradient is then passed downstream.

Complex parameter gradient update

In addition, for most Machine Learning models, the parameters are real. However, in our case, we have both real parameters and complex parameters in our quantum optical circuits. It is a big challenge to update the complex gradients and realize them inside TF.

In order to update the complex parameter, we need to find out the gradient with respect to the conjugate of the parameter. Then, we write the chain rule by

$$\frac{\partial L}{\partial \xi^*} = \frac{\partial L}{\partial \psi_{out_M}} \frac{\partial \psi_{out_M}}{\partial \psi_{out_{M-1}}} ... \frac{\partial \psi_{out_n}}{\partial \xi^*} + \frac{\partial L}{\partial \psi^*_{out_M}} \frac{\partial \psi^*_{out_M}}{\partial \psi^*_{out_{M-1}}} ... \frac{\partial \psi^*_{out_n}}{\partial \xi^*}, \tag{7.2.5}$$

In this case, the downstream for gradient calculation with respect to the complex parameter ξ would be written as:

$$grad_{down} = dy(\frac{\partial \psi_{out_n}}{\partial \xi})^* + dy^* \frac{\partial \psi_{out_n}}{\partial \xi^*}. \tag{7.2.6}$$

As part of our work, we show how to calculate the partial derivative of complex functions with respect to complex variables in Appendix A.1.

7.2.3 Differentiability of the recurrence relation

This part is to explain how to get the partial derivative of each element inside a Gaussian object, which is $\frac{\partial \mathcal{G}}{\partial \boldsymbol{\theta}}$ (or $\frac{\partial \mathcal{G}}{\partial \boldsymbol{\theta}^*}$ for complex parameters).

Here we recall our general recurrence relation in Eq. (6.2.11) for all Gaussian objects:

$$\mathcal{G}_{\boldsymbol{k}+1_i} = \frac{1}{\sqrt{k_i+1}} \left(\boldsymbol{b}_i \mathcal{G}_{\boldsymbol{k}} + \sum_j \sqrt{k_j} \boldsymbol{A}_{ij} \mathcal{G}_{\boldsymbol{k}-1_j} \right), \tag{7.2.7}$$

$$\mathcal{G}_{\boldsymbol{0}} = c. \tag{7.2.8}$$

One critical insight of this kind of recursive function coming from the generating function method is that when we want to compute the tensor elements' gradients, we can directly differentiate this equation with respect to the parameters.

This differentiability can be developed in two ways:

- using the chain rule to connect the three entities $\boldsymbol{A}, \boldsymbol{b}, c$ with any parameters;
- evaluating the generating function gives a direct way to write the derivatives of the parameters.

Chain rule method

The derivative of L with respect to an arbitrary parameter ξ can be obtained by the chain rule:

$$\frac{\partial L}{\partial \xi} = \frac{\partial L}{\partial \mathcal{G}} \frac{\partial \mathcal{G}}{\partial \xi}, \tag{7.2.9}$$

where $\frac{\partial L}{\partial \mathcal{G}}$ is the upstream gradient which can be obtained from AD, so we need to know how to get the second part $\frac{\partial \mathcal{G}}{\partial \xi}$.

Since our recurrence relation for all Gaussian objects is parametrized by triplet $\boldsymbol{A}, \boldsymbol{b}, c$, we apply the chain rule again here:

$$\frac{\partial \mathcal{G}}{\partial \xi} = \sum_{X \in \{\boldsymbol{A}, \boldsymbol{b}, c\}} \frac{\partial \mathcal{G}}{\partial X} \frac{\partial X}{\partial \xi}. \tag{7.2.10}$$

The first term $\frac{\partial \mathcal{G}}{\partial X}$ in Eq. (7.2.10) can be obtained by differentiating the recurrence relation in Eq. (6.2.11) with respect to $\boldsymbol{A}, \boldsymbol{b}, c$.

The derivative $\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{Y}}$ with respect to a matrix $\boldsymbol{Y}$ needs to use **matrix calculus**. Give $\boldsymbol{X} \in \mathbb{R}^{p\times q}$ and $\boldsymbol{Y} \in \mathbb{R}^{m\times n}$, the derivative $\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{Y}}$ is a matrix with the size of $mn \times pq$, which looks like to concatenate the shape of two objects and calculate the derivatives. The "shape" of one object is one vector to describe its dimension and the cutoffs on each dimension. If $\boldsymbol{X}$ and/or $\boldsymbol{Y}$ are vectors or scalars, $\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{Y}}$ degenerates to the Jacobian matrix of $\boldsymbol{Y}$.

So now we know that $\partial_c \mathcal{G}$ has the same shape as $\mathcal{G}$ and it is simply a derivative with respect to a scalar c:

$$\partial_c \mathcal{G}_{\boldsymbol{k}} = \frac{\mathcal{G}_{\boldsymbol{k}}}{c}. \tag{7.2.11}$$

$\partial_{\boldsymbol{b}} \mathcal{G}$ has the shape ($\mathcal{G}$.shape + $\boldsymbol{b}$.shape) and the derivative can be calculated to fill in the $\mathcal{G}$ shape part and the $\boldsymbol{b}$ shape part:

$$\partial_{\boldsymbol{b}} \mathcal{G}_{\boldsymbol{k}+1_i} = \frac{1}{\sqrt{k_i+1}} \left(\boldsymbol{b}_i \partial_{\boldsymbol{b}} \mathcal{G}_{\boldsymbol{k}_i} + \sum_l \sqrt{\boldsymbol{k}_l} \boldsymbol{A}_{il} \partial_{\boldsymbol{b}} \mathcal{G}_{\boldsymbol{k}-1_l} \right), \tag{7.2.12}$$

$$\partial_{\boldsymbol{b}_i} \mathcal{G}_{\boldsymbol{k}+1i} = \mathcal{G}_{\boldsymbol{k}_i}. \tag{7.2.13}$$

The same idea to get two parts of $\partial_{\boldsymbol{A}} \mathcal{G}$, which has the shape ($\mathcal{G}$.shape + $\boldsymbol{A}$.shape):

$$\partial_{\boldsymbol{A}} \mathcal{G}_{\boldsymbol{k}+1_i} = \frac{1}{\sqrt{k_i+1}} \left(\boldsymbol{b}_i \partial_{\boldsymbol{A}} \mathcal{G}_{\boldsymbol{k}_i} + \sum_l \sqrt{\boldsymbol{k}_l} \boldsymbol{A}_{il} \partial_{\boldsymbol{A}} \mathcal{G}_{\boldsymbol{k}-1_l} \right), \tag{7.2.14}$$

$$\partial_{\boldsymbol{A}_{il}} \mathcal{G}_{\boldsymbol{k}+1_i} = \frac{\sqrt{(\boldsymbol{k}+1_i)_l}}{\sqrt{k_i+1}} \mathcal{G}_{\boldsymbol{k}-1_l}. \tag{7.2.15}$$

The second term $\frac{\partial X}{\partial \xi}$ in Eq. (7.2.10) depends on ξ, which can be two different types of parameters:

- the exact parameter of each optical component if the Gaussian object is decomposed, such as the angle of the rotation operator;
- the symplectic matrix $\boldsymbol{S}$ and the displacement vector $\boldsymbol{d}$ of the Gaussian object.

ξ as the exact parameter Let us take the single-mode Gaussian operator $\mathcal{G}_{m,n}$ as an example, which is defined in Eq.(43) of [35] as the product of displacement, rotation, and squeezing operators:

$$\mathcal{G}(\gamma, \phi, \zeta) = \mathcal{D}(\gamma)\mathcal{R}(\phi)\mathcal{S}(\zeta), \tag{7.2.16}$$

and the three entities $c, \boldsymbol{b}, \boldsymbol{A}$ are

$$c = \frac{\exp\left(-\frac{1}{2}\left[|\gamma|^2 + \gamma^{*2} e^{i(\delta+2\phi)} \tanh r\right]\right)}{\sqrt{\cosh r}}, \tag{7.2.17}$$

$$\boldsymbol{b} = \begin{bmatrix} \gamma^* e^{i(\delta+2\phi)} \tanh r + \gamma \\ -\gamma^* e^{i\phi} \operatorname{sech} r \end{bmatrix}, \tag{7.2.18}$$

$$\boldsymbol{A} = \left[\begin{array}{c|c} -e^{i(\delta+2\phi)} \tanh r & e^{i\phi} \operatorname{sech} r \\ \hline e^{i\phi} \operatorname{sech} r & e^{-i\delta} \tanh r \end{array}\right]. \tag{7.2.19}$$

To differentiate the Gaussian object $\mathcal{G}$ with respect to ϕ, one must calculate the derivatives $\frac{\partial c}{\partial \phi}$, $\frac{\partial \boldsymbol{b}}{\partial \phi}$ and $\frac{\partial \boldsymbol{A}}{\partial \phi}$:

$$\frac{\partial c}{\partial \phi} = -ic\gamma^{*2} e^{i(\delta+2\phi)} \tanh r, \tag{7.2.20}$$

$$\frac{\partial \boldsymbol{b}}{\partial \phi} = \begin{bmatrix} 2i\gamma^* e^{i(\delta+2\phi)} \tanh r \\ -i\gamma^* e^{i\phi} \operatorname{sech} r \end{bmatrix}, \tag{7.2.21}$$

$$\frac{\partial \boldsymbol{A}}{\partial \phi} = \left[\begin{array}{c|c} -2ie^{i(\delta+2\phi)} \tanh r & ie^{i\phi} \operatorname{sech} r \\ \hline ie^{i\phi} \operatorname{sech} r & 0 \end{array}\right]. \tag{7.2.22}$$

Then plugging the expressions of $\frac{\partial X}{\partial \xi}$ in Eq. (7.2.10) and then in Eq. (7.2.9), we are able to get the partial derivative with respect to any parameter ξ then get the full gradient.

ξ as $\boldsymbol{S}, \boldsymbol{d}$ Now we can use all equations that we have developed in chapter 6 to go from $\boldsymbol{S}, \boldsymbol{d}$ to $\boldsymbol{A}, \boldsymbol{b}, c$. As we explained about the AD framework, we can simply code all the relations within the TF functions, and then the derivatives of $\boldsymbol{S}, \boldsymbol{d}$ with respect to $\boldsymbol{A}, \boldsymbol{b}, c$ are obtained automatically.

Generating function method

The generation function of the Gaussian tensor $\mathcal{G}$ with its Fock indices $\boldsymbol{k} = (k_1, \ldots, k_\ell)$ (where ℓ can be M-dimensional vector, $2M$-dimensional vector, and $4M$-dimensional vector) is:

$$\Gamma(\boldsymbol{\alpha}) = \sum_{\boldsymbol{k}=\boldsymbol{0}}^{\infty} \frac{\boldsymbol{\alpha}^{\boldsymbol{k}}}{\boldsymbol{k}!} \mathcal{G}_{\boldsymbol{k}}, \tag{7.2.23}$$

where we use the shorthand notation $\sum_{\boldsymbol{k}=\boldsymbol{0}}^{\boldsymbol{\infty}} = \sum_{k_1=0}^{\infty} \cdots \sum_{k_\ell=0}^{\infty}$, $\boldsymbol{\alpha}$ is the rescaled multimode coherent state and $\boldsymbol{k}$ denotes the Fock indices of the tensor. So it is clear that if we want to get the Gaussian tensor $\mathcal{G}_{\boldsymbol{k}}$ from this generating function, we can differentiate it $\boldsymbol{k}$ times and set $\boldsymbol{\alpha}$ as $\boldsymbol{0}$:

$$\mathcal{G}_{\boldsymbol{k}} = \frac{\partial_{\boldsymbol{\alpha}}^{\boldsymbol{k}}}{\sqrt{\boldsymbol{k}!}} \Gamma(\boldsymbol{\alpha}) \bigg|_{\boldsymbol{\alpha}=\boldsymbol{0}}. \tag{7.2.24}$$

This generating function is in exponential form for Gaussian objects:

$$\Gamma(\boldsymbol{\alpha}) = c \exp\left(\boldsymbol{\alpha}^T \boldsymbol{b} + \frac{1}{2} \boldsymbol{\alpha}^T \boldsymbol{A} \boldsymbol{\alpha}\right), \tag{7.2.25}$$

because of this exponential, we could recursively express the high-order derivatives of Gaussian objects.

If we want to get the derivative with respect to the parameter ξ, we need to derive the Gaussian object's generating function and set $\boldsymbol{\alpha}$ as $\boldsymbol{0}$:

$$\frac{\partial \mathcal{G}_{\boldsymbol{k}}}{\partial \xi} = \frac{\partial_{\boldsymbol{\alpha}}^{\boldsymbol{k}}}{\sqrt{\boldsymbol{k}!}} \partial_\xi \Gamma(\boldsymbol{\alpha}) \bigg|_{\boldsymbol{\alpha}=\boldsymbol{0}}. \tag{7.2.26}$$

By inserting the exponential form in Eq. (7.2.25), we have:

$$\frac{\partial \mathcal{G}_{\boldsymbol{k}}}{\partial \xi} = \frac{\partial_{\boldsymbol{\alpha}}^{\boldsymbol{k}}}{\sqrt{\boldsymbol{k}!}} \partial_\xi c e^{(\boldsymbol{\alpha}^T \boldsymbol{b} + \frac{1}{2} \boldsymbol{\alpha}^T \boldsymbol{A} \boldsymbol{\alpha})} \bigg|_{\boldsymbol{\alpha}=\boldsymbol{0}} \tag{7.2.27}$$

$$= \frac{\partial_{\boldsymbol{\alpha}}^{\boldsymbol{k}}}{\sqrt{\boldsymbol{k}!}} \Gamma(\boldsymbol{\alpha}) \left(\frac{\partial_\xi c}{c} + \partial_\xi (\boldsymbol{\alpha}^T \boldsymbol{b} + \frac{1}{2} \boldsymbol{\alpha}^T \boldsymbol{A} \boldsymbol{\alpha})\right) \bigg|_{\boldsymbol{\alpha}=\boldsymbol{0}} \tag{7.2.28}$$

$$= \frac{\partial_{\boldsymbol{\alpha}}^{\boldsymbol{k}}}{\sqrt{\boldsymbol{k}!}} \Gamma(\boldsymbol{\alpha}) \left(\frac{\partial_\xi c}{c} + \boldsymbol{\alpha}^T \partial_\xi \boldsymbol{b} + \frac{1}{2} \boldsymbol{\alpha}^T \partial_\xi \boldsymbol{A} \boldsymbol{\alpha}\right) \bigg|_{\boldsymbol{\alpha}=\boldsymbol{0}}. \tag{7.2.29}$$

In this equation, we also have the following:

$$\frac{\partial_{\boldsymbol{\alpha}}^{\boldsymbol{k}}}{\sqrt{\boldsymbol{k}!}}\boldsymbol{\alpha}^{\boldsymbol{n}}\Gamma(\boldsymbol{\alpha})\Big|_{\boldsymbol{\alpha}=\boldsymbol{0}} = \sqrt{(\boldsymbol{k})_{\boldsymbol{n}}}\mathcal{G}_{\boldsymbol{k}-\boldsymbol{n}}, \tag{7.2.30}$$

where $(\boldsymbol{k})_{\boldsymbol{n}} = k(k-1)(k-2)\dots(k-n+1)$. Using Eq. (7.2.30), Eq. (7.2.24) to replace the term in Eq. (7.2.29), we have:

$$\frac{\partial\mathcal{G}_{\boldsymbol{k}}}{\partial\xi} = \frac{1}{c}\frac{\partial c}{\partial c}\mathcal{G}_{\boldsymbol{k}} + \sum_i \partial_\xi \boldsymbol{b}\sqrt{k_i}\mathcal{G}_{\boldsymbol{k}-1_i} + \sum_{i>j}\frac{\partial \boldsymbol{A}_{ij}}{\partial\xi}\sqrt{k_i k_j}\mathcal{G}_{\boldsymbol{k}-1_i-1_j} + \frac{1}{2}\sum_i \frac{\partial \boldsymbol{A}_{ii}}{\partial\xi}\sqrt{k_i(k_i-1)}\mathcal{G}_{\boldsymbol{k}-2_i}. \tag{7.2.31}$$

This equation provides the derivative of our recurrence relation for all Gaussian objects with respect to the parameter ξ. It is a more direct way to compute the derivatives and the gradient without passing through the chain rule.

In the example of the single-mode Gaussian operator, we only need to insert the derivatives, and we obtain::

$$\begin{aligned}\frac{\partial\mathcal{G}_{m,n}}{\partial\phi} = &-i\gamma^{*2}e^{i(\delta+2\phi)}\tanh r\mathcal{G}_{m,n} + 2i\gamma^* e^{i(\delta+2\phi)}\tanh r\sqrt{m}\mathcal{G}_{m-1,n} - i\gamma^* e^{i\phi}\operatorname{sech} r\sqrt{n}\mathcal{G}_{m,n-1}\\ &+ie^{i\phi}\operatorname{sech} r\sqrt{mn}\mathcal{G}_{m-1,n-1} + \frac{1}{2}\left[-2ie^{i(\delta+2\phi)}\tanh r\sqrt{m(m-1)}\mathcal{G}_{m-2,n}\right].\end{aligned} \tag{7.2.32}$$

7.3 Two ways to optimize Gaussian operators

Depending on how to treat the Gaussian operators, we have two optimizations:

- decomposition of the Gaussian operators into fundamental quantum optical components corresponds to the Euclidean optimization. This method is considered as hardware-friendly because one can directly obtain the values for each component.
- update of the Gaussian operators and decomposes it afterward, corresponding to the Riemannian optimization. Moreover, this method is theory-friendly because only the update of the symplectic matrix needs to be computed in this method.

7.3.1 Euclidean optimization, hardware-friendly

The hardware-friendly optimization refers to the gradient update of each optical component we use in the quantum circuits. All the elements need to be updated during each optimization iteration, where each component has its CV parameters.

This section explains how to decompose the Gaussian operators. The update rule of real and complex parameters is shown in section 4.5.

For example, we have a M-mode circuit characterized by M squeezing gates $\mathcal{S}(\boldsymbol{\zeta})$, M displacement gates $\mathcal{D}(\boldsymbol{\gamma})$, M Kerr gates $K(\boldsymbol{\kappa})$ and two interferometers V_1, V_2, there are

$$2M + 2M + M + 2 * M^2 = 2M^2 + 5M \tag{7.3.1}$$

real parameters inside the circuit; therefore, to update all the real or complex parameters inside this circuit, we need to compute $2M^2 + 5M$ derivatives each run.

Gaussian operators decomposition

Rather than working on single-mode Gaussian objects, the multi-mode Gaussian object is more interesting because it can exhibit entanglement, which is a key for various quantum applications, such as quantum computing and quantum cryptography.

Here we present the decomposition of an arbitrary Gaussian operation $\mathcal{G}$ based on the Bloch-Messiah decomposition [143, 144]. The Bloch-Messiah reduction allows decomposing of the Gaussian operation into a very simple scheme with fundamental optical components, such as rotation operators, displacement operators, squeezing operators, beam-splitters, and interferometers. In this way, one can take an arbitrary Gaussian operator as a whole and decompose it with implementable optical components.

We note that the rotation gate $\mathcal{R}$, the beam-splitter $\mathcal{B}$, and the interferometer $\mathcal{W}$ are the real-valued parameterized gates, whereas the squeezing gate $\mathcal{S}$ and the displacement gate $\mathcal{D}$ are complex-valued parameterized gates.

Decomposition of single-mode Gaussian operators In the single-mode case (as in Fig. 7.5), we use a sequence of single-mode squeezer $\mathcal{S}(\zeta)$, single-mode phase rotation $\mathcal{R}(\phi)$ and single-mode displacement gates $\mathcal{D}(\gamma)$:

$$\mathcal{G}^{(1)}(\gamma, \phi, \zeta) = \mathcal{D}(\gamma)\mathcal{R}(\phi)\mathcal{S}(\zeta). \tag{7.3.2}$$

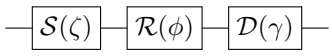


Figure 7.5: Single-mode Gaussian transformation architecture (states move through the circuit from left to right).

Decomposition of two-mode Gaussian operators In the two-mode case (as in Fig. 7.6), we have a first beam-splitter $\mathcal{B}(\theta', \varphi')$, two single-mode squeezers $\mathcal{S}(\zeta_1), \mathcal{S}(\zeta_2)$, another second beam-splitter $\mathcal{B}(\theta, \varphi)$, two single-mode phase rotations $\mathcal{R}(\phi_1), \mathcal{R}(\phi_2)$ and two single-mode displacement gates $\mathcal{D}(\gamma_1), \mathcal{D}(\gamma_2)$:

$$\mathcal{G}^{(2)}(\boldsymbol{\gamma}, \boldsymbol{\phi}, \theta', \varphi', \boldsymbol{\zeta}, \theta, \varphi) = \mathcal{D}(\boldsymbol{\gamma})\mathcal{R}(\boldsymbol{\phi})\mathcal{B}(\theta', \varphi')\mathcal{S}(\boldsymbol{\zeta})\mathcal{B}(\theta, \varphi). \tag{7.3.3}$$

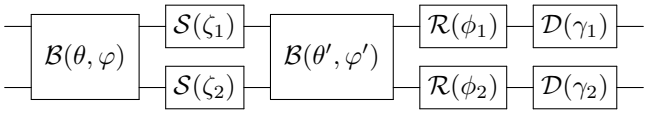


Figure 7.6: Two-mode Gaussian transformation architecture (states move through the circuit left to right).

Decomposition of multi-mode Gaussian operators For M-modes (as in Fig. 7.7), we have the structure composed by an M-mode interferometer $\mathcal{W}(\boldsymbol{V}_2)$, M single-mode squeezers $\mathcal{S}(\zeta_i)$, a second M-mode interferometer $\mathcal{W}(\boldsymbol{V}_1)$ and M single-mode displacements $\mathcal{D}(\gamma_i)$. Afterward, the interferometers can be decomposed as a suitable arrangement of beam-splitters, for example, following Clements decomposition [145, 146, 147].

$$\mathcal{G}^{(M)} = \mathcal{D}(\boldsymbol{\gamma})\mathcal{W}(\boldsymbol{V}_1)\mathcal{S}(\boldsymbol{\zeta})\mathcal{W}(\boldsymbol{V}_2), \tag{7.3.4}$$

where $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$ are unitary matrices.

This optimization is really clear to understand the parameter evolution inside the circuit for each element, and also easy to pass them into constructing a parameterized photonic integrated circuit. However, the redundancy of the calculation for each operator would be a big issue when it comes to a higher number of modes. That is why we come up with the idea of considering all Gaussian components as a single global object to optimize in the next section.

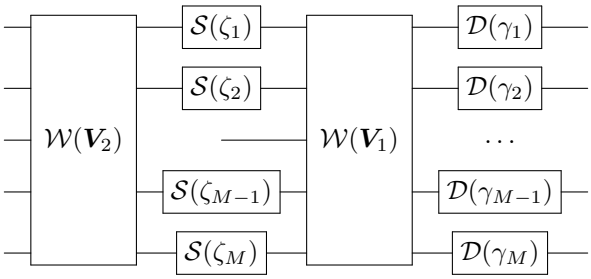


Figure 7.7: M-mode Gaussian transformation architecture (states move through the circuit left to right).

7.3.2 Riemannian optimization, theory-friendly

The theory-friendly optimization refers to the gradient update depending on the whole Gaussian operator; we can decompose the Gaussian operator afterward into the fundamental optical components. The advantage of considering only one Gaussian operator is to avoid calculating enormous gradients for each element, which is the redundant part of the optimization.

In section 3.2.2, we point out that the M-mode Gaussian unitary is characterized by a $2M \times 2M$ real symplectic matrix and a $2M$ real vector on the phase space. Therefore the update of the Gaussian operator turns into the update of the real symplectic matrix and the real vector. The update of the real vector is the same idea as the update of the real parameter in section 4.5.1:

$$d \leftarrow d - \eta \frac{\partial L}{\partial d}, \tag{7.3.5}$$

using the Euclidean gradient $\frac{\partial L}{\partial d}$.

Let us now discuss how to update the real symplectic matrix. Especially when the Gaussian unitary refers to an interferometer, the real symplectic matrix becomes the corresponding unitary matrix and it belongs to a unitary group. We also show the unitary group update for interferometers.

Update the real symplectic matrix

Riemannian manifold $\mathrm{Sp}(2n)$ We describe the manifold of real symplectic $2n \times 2n$ matrices as an embedded submanifold of $\mathbb{R}^{2n\times 2n}$:

$$\mathrm{Sp}(2n, \mathbb{R}) = \{\boldsymbol{S} \in \mathbb{R}^{2n\times 2n} | \boldsymbol{S}\boldsymbol{\Omega}\boldsymbol{S}^T = \boldsymbol{\Omega}\}, \tag{7.3.6}$$

where $\boldsymbol{\Omega} = \left(\begin{smallmatrix} 0 & \mathbb{1} \\ -\mathbb{1} & 0 \end{smallmatrix}\right)$. Given that the condition $\boldsymbol{S}\boldsymbol{\Omega}\boldsymbol{S}^T = \boldsymbol{\Omega}$ is quadratic in $\boldsymbol{S}$, the manifold of symplectic matrices is not a linear subspace of $\mathbb{R}^{2n\times 2n}$, which means that we are likely to *leave* the manifold after a straight step of gradient descent. The following sections explain how to overcome this difficulty by using the geodesic on the manifold.

Note that unless details are relevant, we abbreviate $\mathrm{Sp}(2n, \mathbb{R})$ with Sp.

Tangent and Normal spaces If we differentiate the quadratic condition $\boldsymbol{S}\boldsymbol{\Omega}\boldsymbol{S}^T = \boldsymbol{\Omega}$ we obtain the *linear* tangency condition $\boldsymbol{X}\boldsymbol{\Omega}\boldsymbol{S}^T + \boldsymbol{S}\boldsymbol{\Omega}\boldsymbol{X}^T = 0$. All the matrices $\boldsymbol{X}$ that satisfy the new condition form a linear subspace of $\mathbb{R}^{2n\times 2n}$ called the tangent space of Sp at the point $\boldsymbol{S}$:

$$T_{\boldsymbol{S}}\mathrm{Sp} = \{\boldsymbol{X} \in \mathbb{R}^{2n\times 2n} | \boldsymbol{X}\boldsymbol{\Omega}\boldsymbol{S}^T + \boldsymbol{S}\boldsymbol{\Omega}\boldsymbol{X}^T = 0_{2n}\} \tag{7.3.7}$$

$$= \{\boldsymbol{S}\boldsymbol{\Omega}\boldsymbol{A} | \boldsymbol{A} = \boldsymbol{A}^T\}. \tag{7.3.8}$$

Eq. (7.3.8) is a compact way of parametrizing the tangent space at $\boldsymbol{S}$ using symmetric matrices. The tangent space can be found by imposing $\boldsymbol{X} = \boldsymbol{S}\boldsymbol{\Omega}\boldsymbol{A}$ in the tangency condition.

As a special case, the Lie algebra of sp is the tangent space at the identity, i.e.

$$\begin{aligned}\mathrm{sp}(2n,\mathbb{R}) &= T_e\mathrm{Sp}(2n,\mathbb{R}) && (7.3.9)\\ &= \{\boldsymbol{X}\in\mathbb{R}^{2n\times 2n}|\boldsymbol{X}\boldsymbol{\Omega}+\boldsymbol{\Omega}\boldsymbol{X}^T=0_{2n}\} && (7.3.10)\\ &= \{\boldsymbol{\Omega}\boldsymbol{A}|\boldsymbol{A}=\boldsymbol{A}^T\}. && (7.3.11)\end{aligned}$$

We can then define the normal space at $\boldsymbol{S}$ as the linear space containing all the elements that are orthogonal to $T_{\boldsymbol{S}}\mathrm{Sp}$:

$$\begin{aligned}N_{\boldsymbol{S}}\mathrm{Sp} &= \{\boldsymbol{W}\in\mathbb{R}^{2n\times 2n}|\mathrm{Tr}(\boldsymbol{W}^T\boldsymbol{X})=0_{2n},\boldsymbol{X}\in T_{\boldsymbol{S}}\mathrm{Sp}\} && (7.3.12)\\ &= \{\boldsymbol{\Omega}\boldsymbol{S}\boldsymbol{B}|\boldsymbol{B}=-\boldsymbol{B}^T\}, && (7.3.13)\end{aligned}$$

with Eq. (7.3.13) showing that we can parametrize the normal space at each point in Sp using anti-symmetric matrices.

Riemannian metric on $\mathrm{Sp}(2n)$ A Riemannian manifold such as $\mathrm{Sp}(2n,\mathbb{R})$ comes equipped with an inner product $\langle\cdot,\cdot\rangle_{\boldsymbol{S}}$ on the tangent space $T_{\boldsymbol{S}}\mathrm{Sp}$ at each point $\boldsymbol{S}\in\mathrm{Sp}$. The family of inner products forms the Riemannian metric tensor. The inner product in $T_{\boldsymbol{S}}\mathrm{Sp}$ is defined as

$$\langle\boldsymbol{X},\boldsymbol{Y}\rangle_{\boldsymbol{S}} = \langle\boldsymbol{S}^{-1}\boldsymbol{X},\boldsymbol{S}^{-1}\boldsymbol{Y}\rangle = \langle\boldsymbol{R}\boldsymbol{X},\boldsymbol{Y}\rangle, \tag{7.3.14}$$

where $\boldsymbol{R}=\boldsymbol{S}^{-\boldsymbol{T}}\boldsymbol{S}^{-1}=\boldsymbol{\Omega}\boldsymbol{S}\boldsymbol{S}^T\boldsymbol{\Omega}^T$ and note that $\boldsymbol{R}^{-1}=\boldsymbol{S}\boldsymbol{S}^T$.

Consider now a cost function $L:\mathrm{Sp}\to\mathbb{R}$. The Euclidean gradient ∂L at the point $\boldsymbol{S}$ (which is computed using the embedding coordinates in $\mathbb{R}^{2n\times 2n}$) is related to the Riemannian gradient $\nabla L\in T_{\boldsymbol{S}}\mathrm{Sp}$ by the compatibility condition

$$\langle\nabla L,\boldsymbol{X}\rangle_{\boldsymbol{S}} = \langle\partial L,\boldsymbol{X}\rangle \quad \forall\boldsymbol{X}\in T_{\boldsymbol{S}}\mathrm{Sp}. \tag{7.3.15}$$

After rearranging the terms, the condition is equivalent to

$$\langle\boldsymbol{R}\nabla L-\partial L,\boldsymbol{X}\rangle = 0 \quad \forall\boldsymbol{X}\in T_{\boldsymbol{S}}\mathrm{Sp}. \tag{7.3.16}$$

This means that $\boldsymbol{R}\nabla L-\partial L\in N_{\boldsymbol{S}}\mathrm{Sp}$ and therefore, it must be possible to write

$$\boldsymbol{R}\nabla L-\partial L=\boldsymbol{\Omega}\boldsymbol{S}\boldsymbol{B}, \tag{7.3.17}$$

for some anti-symmetric matrix $\boldsymbol{B}$. At the same time we have the tangency condition $\nabla L\boldsymbol{\Omega}\boldsymbol{S}^T+\boldsymbol{S}\boldsymbol{\Omega}\nabla^T L=0$. If we replace ∇L from Eq. (7.3.17) into the tangency condition, we obtain an expression for $\boldsymbol{B}$ and we can finally write the Riemannian gradient on the symplectic group:

$$\nabla L=\frac{\boldsymbol{S}}{2}(\boldsymbol{Z}+\boldsymbol{\Omega}\boldsymbol{Z}^T\boldsymbol{\Omega}), \tag{7.3.18}$$

where $\boldsymbol{Z}=\boldsymbol{S}^T\partial L$.

The symplectic matrix that describes an interferometer (defined in Eq. (3.50)) belongs to the intersection of the orthogonal group $\mathrm{O}(2n)$ and the symplectic group $\mathrm{Sp}(2n)$, which is a Unitary group $\mathrm{U}(n)$:

$$\mathrm{U}(n)=\{\boldsymbol{M}\in\mathbb{C}^{n\times n}|\boldsymbol{M}^\dagger\boldsymbol{M}=\boldsymbol{M}\boldsymbol{M}^\dagger=\mathbb{1}_n\}. \tag{7.3.19}$$

We can go through the same arguments as with the symplectic group and obtain the Riemannian gradient in the unitary group (More calculation details are in Appendix B):

$$\nabla L=\frac{\boldsymbol{M}}{2}\left(\boldsymbol{Z}-\boldsymbol{Z}^\dagger\right), \tag{7.3.20}$$

where $\boldsymbol{Z}=\boldsymbol{M}^\dagger\partial L$.

Geodesic optimization on $\mathrm{Sp}(2n)$ and $\mathrm{U}(n)$ The shortest curve connecting two points on a Riemannian manifold $\mathcal{M}$ is called a geodesic, and it can be defined by the starting point $\gamma(0) = \boldsymbol{p}$ and its velocity on the tangent space at that point: $\boldsymbol{V} = \dot{\gamma}(0) \in T_{\boldsymbol{p}}\mathcal{M}$. For the symplectic and unitary groups, geodesics take the following form (which can be found by minimizing a variational formulation of the path length between two points [54, 59]):

$$\gamma^{\mathrm{Sp}(2n)}(t) = \boldsymbol{S} e^{t(\boldsymbol{S}^{-1}\boldsymbol{V})^T} e^{t[\boldsymbol{S}^{-1}\boldsymbol{V} - (\boldsymbol{S}^{-1}\boldsymbol{V})^T]}, \tag{7.3.21}$$

$$\gamma^{\mathrm{U}(n)}e(t) = \boldsymbol{M} e^{t(\boldsymbol{M}^\dagger \boldsymbol{V})^T}. \tag{7.3.22}$$

By using a geodesic, we guarantee that each update step remains on the manifold.

For gradient descent, we use $\boldsymbol{V} = -\nabla L$:

$$\gamma^{\mathrm{Sp}(2n)}(t) = \boldsymbol{S} e^{-t\boldsymbol{Y}} e^{-t(\boldsymbol{Y} - \boldsymbol{Y}^T)}, \tag{7.3.23}$$

with $\boldsymbol{Y} = \boldsymbol{S}^{-1}\nabla L = \frac{1}{2}(\boldsymbol{Z} + \boldsymbol{\Omega}\boldsymbol{Z}^T\boldsymbol{\Omega})$. For the unitary group, we obtain

$$\gamma^{\mathrm{U}(n)}(t) = \boldsymbol{M} e^{-t\boldsymbol{Y}}, \tag{7.3.24}$$

with $\boldsymbol{Y} = \boldsymbol{M}^\dagger \nabla L = \frac{1}{2}(\boldsymbol{Z} - \boldsymbol{Z}^\dagger) = \frac{1}{2}(\boldsymbol{M}^\dagger \partial L - (\partial L)^\dagger \boldsymbol{M})$. We now have a geodesic update formula that we can apply instead of the usual gradient descent step. The parameter t takes the role of the learning rate (which we fix depending on the application). For the symplectic group, we have

$$\boldsymbol{Z}_k \leftarrow \boldsymbol{S}_k^T \partial L, \tag{7.3.25}$$

$$\boldsymbol{Y}_k \leftarrow \frac{1}{2}(\boldsymbol{Z}_k + \boldsymbol{\Omega}\boldsymbol{Z}_k^T\boldsymbol{\Omega}), \tag{7.3.26}$$

$$\boldsymbol{S}_{k+1} \leftarrow \boldsymbol{S}_k e^{-t\boldsymbol{Y}_k} e^{-t(\boldsymbol{Y}_k - \boldsymbol{Y}_k^T)}. \tag{7.3.27}$$

For the unitary group, we have

$$\boldsymbol{Z}_k \leftarrow \boldsymbol{M}_k^\dagger \partial L, \tag{7.3.28}$$

$$\boldsymbol{Y}_k \leftarrow \frac{1}{2}(\boldsymbol{Z}_k - \boldsymbol{Z}_k^\dagger), \tag{7.3.29}$$

$$\boldsymbol{M}_{k+1} \leftarrow \boldsymbol{M}_k e^{-t\boldsymbol{Y}_k}. \tag{7.3.30}$$

Finally, we obtain the orthogonal matrix of the interferometer using Eq. (3.50).

The Riemannian update step in practice

In practice, once the relevant Euclidean gradient ∂L has been computed, we can compute the Riemannian gradient ∇L and update the real symplectic matrix with $\boldsymbol{S}$ the Eq. (7.3.27).

The backpropagation procedure is shown in Fig. 7.8.

We can first go with the right side arrows to see the forward pass in solid lines. Since we have the real symplectic matrix $\boldsymbol{S}$ and the real displacement vector $\boldsymbol{d}$ to characterize any Gaussian transformation, we can calculate its real covariance matrix $\boldsymbol{V}$ and mean vector $\bar{\boldsymbol{r}}$; then, we change the basis to the complex covariance matrix and calculate the Cayley transform of the covariance matrix, to get the triplet $(\boldsymbol{A}, \boldsymbol{b}, c)$; by using this triplet, we can write down the Fock representation of this Gaussian evolution; next step is to supply this into the cost function and get the final value. This forward pass explains how the math works in our equations to get the triplet.

Secondly, we are going to see the backward pass in dotted lines. Thanks to the Auto Differentiation technique, for any transformation, once we write down the function in their way, it automatically gives the gradient related to the input and output variables. The partial derivative goes from the cost function to the Fock representation by AD; and then the chain rule between the triplet $(\boldsymbol{A}, \boldsymbol{b}, c)$ to the Fock representation

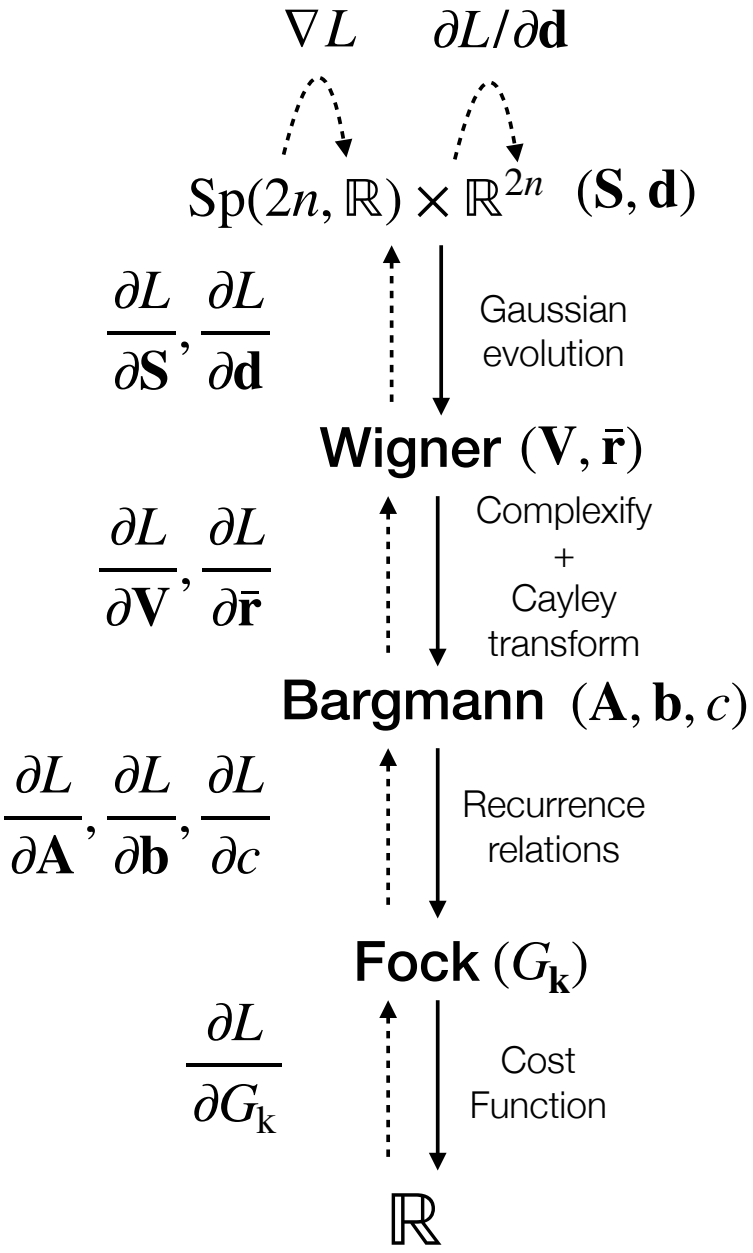


Figure 7.8: The detailed forward (solid line) and backward (dotted line) passes.

is computed by the differentiability of the recurrence relations; at the end, the partial derivative of $\boldsymbol{S}$ and $\boldsymbol{d}$ with respect to the triplet $(\boldsymbol{A}, \boldsymbol{b}, c)$ is also given by AD.

The Riemannian gradient ∇L for the geodesic update is calculated via the chain rule and Eq. (7.3.18), which backpropagates the gradient of the cost function with respect to the Fock amplitudes $\frac{\partial L}{\partial G_k}$ all the way to ∇L, while the gradient $\frac{\partial L}{\partial \boldsymbol{d}}$ is used directly to optimize $\boldsymbol{d}$ on $\mathbb{R}^{2n}$.

For example, the Euclidean gradient of the symplectic matrix can be calculated via the chain rule:

$$\frac{\partial L}{\partial \boldsymbol{S}} = 2\Re\left[\sum_{X=A,b,c}\sum_{k}\frac{\partial L}{\partial \mathcal{G}_k}\frac{\partial \mathcal{G}_k}{\partial X}\frac{\partial X}{\partial \boldsymbol{S}}\right]. \tag{7.3.31}$$

In this expression, $\frac{\partial L}{\partial \mathcal{G}_k}$ is the upstream gradient which can be obtained from an AD framework such as TensorFlow, $\frac{\partial \mathcal{G}_k}{\partial X}$ is computed by differentiating the recurrence relation in Eq. (6.2.11) and $\frac{\partial X}{\partial \boldsymbol{S}}$ is also handled by the AD framework, and it depends on the functional relation between the symplectic matrix and $X = \boldsymbol{A}, \boldsymbol{b}, c$.

7.4 Complex natural gradient descent algorithm

The natural gradient (NG) is known as the gradient descent algorithm that points the steepest direction on the manifold at the expense of needing a matrix inversion of an $n \times n$ matrix at each step, where n is the number of parameters being updated during that step [39, 40, 41]. Such a matrix represents the metric tensor of the parameter space as seen by the cost function.

We will recall how the natural gradient is introduced on Riemannian manifolds and describe the Fubini-Study (FS) metric tensor. Then, we introduce our work about extending the metric tensor to complex parameters.

7.4.1 Steepest-step on the Riemannian manifold and the Fubini-Study metric tensor

Steepest descent on Riemannian manifolds

Let us define a differentiable cost function $L : \mathbb{R}^n \to \mathbb{R}$ which maps a vector of real parameters $\boldsymbol{\theta} = [\theta_1, ..., \theta_n]^T \in \mathbb{R}^n$ to a value in $\mathbb{R}$. Our objective is to minimize $L(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$. A relatively simple optimization procedure is based on gradient descent, which in turn is based on a first-order expansion of L (throughout this section, we use the Einstein summation convention, i.e., repeated indices are summed over):

$$L(\boldsymbol{\theta} + d\boldsymbol{\theta}) \approx L(\boldsymbol{\theta}) + \frac{\partial L(\boldsymbol{\theta})}{\partial \theta_i} d\theta_i. \tag{7.4.1}$$

Therefore, we can use the following rule:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \frac{\partial L_t}{\partial \boldsymbol{\theta}}, \tag{7.4.2}$$

where η is the learning rate and may depend on step t.

In Eq. (7.4.2), each direction is weighted equally by a uniform learning rate η_t, which is fine if infinitesimal distances on the manifold are Euclidean:

$$d_E(\boldsymbol{\theta}, \boldsymbol{\theta} + d\boldsymbol{\theta}) = \sqrt{d\theta_i d\theta_i}. \tag{7.4.3}$$

However, if distances are computed using a different metric, such as the Riemannian metric, the gradient computed from the cost function as in Eq. (7.4.2) no longer points in the optimal direction. Let us then

endow the parameter space with a metric tensor $g(\boldsymbol{\theta})$ such that an infinitesimal length at the point $\boldsymbol{\theta}$ is computed as

$$d(\boldsymbol{\theta}, \boldsymbol{\theta} + d\boldsymbol{\theta}) = \sqrt{g(\boldsymbol{\theta})_{ij} d\theta_i d\theta_j}. \tag{7.4.4}$$

Eq. (7.4.2) corresponds to $g(\boldsymbol{\theta}) = \mathbb{1}$.

We now have a Riemannian structure on the parameter space, characterized by the metric tensor $g(\boldsymbol{\theta})$, which effectively describe the space. From an infinitesimal perspective, the steepest ascent or descent direction is going to be adjusted by the presence of g. From a macroscopic perspective, distances between points must be computed by integrating along geodesics, with the consequence that the shortest path between two points is generally no longer a straight line as in Euclidean space. Note that from now on, we will omit the dependence on $\boldsymbol{\theta}$ when writing g, but we stress that g is generally a local quantity.

It can then be shown using the method of Lagrange multipliers [40] that the steepest descent direction on a Riemannian manifold is given by the so-called natural gradient $\nabla_{\boldsymbol{\theta}}$, defined as

$$\nabla_{\boldsymbol{\theta}} := g^{-1} \frac{\partial}{\partial \boldsymbol{\theta}}. \tag{7.4.5}$$

This leads to a new parameter update rule:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \nabla_{\boldsymbol{\theta}} L_t \tag{7.4.6}$$

$$= \boldsymbol{\theta}_t - \eta_t g^{-1} \frac{\partial L_t}{\partial \boldsymbol{\theta}}. \tag{7.4.7}$$

Note that the metric tensor g may happen to be noninvertible for some values of $\boldsymbol{\theta}$. This corresponds to singular points in the parameter space and will be discussed briefly later on.

As an example, consider polar coordinates for the two-dimensional plane [41]. When using $(r, \phi) \in \mathbb{R}_+ \times [0, 2\pi[$ to identify points, one must remember that ϕ represents an angle and, therefore, not a distance per se. Thus, if one moves away from the point (r, ϕ) with infinitesimal shifts dr and $d\phi$, the infinitesimal distance will be computed as

$$ds = \sqrt{dr^2 + r^2 d\phi^2}. \tag{7.4.8}$$

Therefore, the metric tensor for polar coordinates is

$$g = \begin{pmatrix} 1 & 0 \\ 0 & r^2 \end{pmatrix}. \tag{7.4.9}$$

The natural gradient for a cost function defined on polar coordinates then becomes

$$\nabla_{(r,\phi)} L = \begin{pmatrix} \frac{\partial L}{\partial r} \\ \frac{1}{r^2} \frac{\partial L}{\partial \phi} \end{pmatrix}. \tag{7.4.10}$$

We can easily see that the metric is singular at the origin, where it becomes noninvertible (there is a zero eigenvalue). This does not mean that infinitesimal distances from the origin cannot be calculated or that distances somehow lose meaning; it simply means that at least one parameter becomes redundant because at the origin $ds = dr$ with no dependence on ϕ. We can mitigate this issue by using pseudoinverses and introducing regularization in practice, which is introduced in Eq. (9.2.8).

The Fubini-Study metric tensor

We now describe a metric in the space of quantum states. Let us consider a parametrized quantum state $\psi_{\boldsymbol{\theta}}$ where the parameters are real, i.e. $\boldsymbol{\theta} \in \mathbb{R}^n$:

$$\psi_{\boldsymbol{\theta}} = U(\boldsymbol{\theta}) \psi_{\mathbf{0}}. \tag{7.4.11}$$

where $U(\boldsymbol{\theta})$ denotes a parametrized unitary transformation, and $\psi_{\mathbf{0}}$ a fixed initial state. Typically, such a state can be used to describe an ansatz prepared by a parametrized quantum circuit [44].

In a VQE context, a Hamiltonian $\mathcal{H}$ is eventually evaluated. In this case, the cost function represents the energy that one aims to minimize:

$$L(\boldsymbol{\theta}) := \langle \psi_{\boldsymbol{\theta}} | \mathcal{H} | \psi_{\boldsymbol{\theta}} \rangle. \tag{7.4.12}$$

Otherwise, for the purpose of state preparation or gate synthesis, one would seek to maximize the quantum fidelity [148, 35, 149].

Note that minimization of this cost function is equivalent to a minimization of the energy, where the Hamiltonian is $\mathcal{H} = -\psi_{\text{target}} \psi_{\text{target}}^{\dagger}$. Rather conveniently, fidelity is not sensitive to the global phase of $\psi_{\boldsymbol{\theta}}$ or ψ_{target}, which is a useful feature, given that the global phase is physically unobservable. Sensitivity to the global phase (as we will see shortly) would be an issue for using Euclidean distance (such as $\|\psi_{\text{target}} - \psi_{\boldsymbol{\theta}}\|$) as a loss.

Following the approach of Provost and Vallee [150], we consider the infinitesimal Euclidean distance between two states, and then we adjust it to be insensitive to the global phase. We start from a first-order expansion of $\psi_{\boldsymbol{\theta}}$ around $\boldsymbol{\theta}$:

$$\psi_{\boldsymbol{\theta}+d\boldsymbol{\theta}} \approx \psi_{\boldsymbol{\theta}} + \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_i} d\theta_i. \tag{7.4.13}$$

The Euclidean distance between $\psi_{\boldsymbol{\theta}}$ and $\psi_{\boldsymbol{\theta}+d\boldsymbol{\theta}}$ is

$$ds^2 = \|\psi_{\boldsymbol{\theta}} - \psi_{\boldsymbol{\theta}+d\boldsymbol{\theta}}\|^2 = \left\langle \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_i} \middle| \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_j} \right\rangle d\theta_i d\theta_j. \tag{7.4.14}$$

Separating the real and imaginary parts of the Hermitian tensor, we can write

$$\left\langle \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_i} \middle| \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_j} \right\rangle = \gamma_{ij} + i\sigma_{ij}, \tag{7.4.15}$$

where $\gamma_{ij} = \gamma_{ji}$ and $\sigma_{ij} = -\sigma_{ji}$ due to the inner product having to satisfy $\langle a|b\rangle = \langle b|a\rangle^*$. This implies that effectively only the real part matters (as the full contraction of a symmetric tensor with an anti-symmetric one yields zero), i.e.,

$$\|\psi_{\boldsymbol{\theta}} - \psi_{\boldsymbol{\theta}+d\boldsymbol{\theta}}\|^2 = \gamma_{ij} d\theta_i d\theta_j. \tag{7.4.16}$$

The anti-symmetric part $\sigma_{ij} = \text{Im}[\langle \psi_{\boldsymbol{\theta}}, \partial_j \psi_{\boldsymbol{\theta}} \rangle]$ is known as the Berry curvature [151], and its significance is beyond the scope of this work.

At this stage, we cannot yet interpret γ_{ij} as a metric tensor on the space of physical quantum states because it is sensitive to global phase: had we used the physically identical state $\psi'_{\boldsymbol{\theta}} = e^{i\alpha(\boldsymbol{\theta})} \psi_{\boldsymbol{\theta}}$ for some real function $\alpha(\boldsymbol{\theta})$, the tensor γ_{ij} would have been

$$\gamma'_{ij} = \text{Re}\left[\left\langle \frac{\partial \psi'_{\boldsymbol{\theta}}}{\partial \theta_i} \middle| \frac{\partial \psi'_{\boldsymbol{\theta}}}{\partial \theta_j} \right\rangle \right] \tag{7.4.17}$$

$$= \gamma_{ij} + (\frac{\partial \alpha}{\partial \theta_i})\beta_j + (\frac{\partial \alpha}{\partial \theta_j})\beta_i + (\frac{\partial \alpha}{\partial \theta_i})(\frac{\partial \alpha}{\partial \theta_j}), \tag{7.4.18}$$

with $\beta_j = -i\langle \psi_{\boldsymbol{\theta}} | \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_j} \rangle$. Note that $\beta_j \in \mathbb{R}$ due to the norm of $\psi_{\boldsymbol{\theta}}$ being a constant:

$$0 = \frac{\partial}{\partial \theta_j} \langle \psi_{\boldsymbol{\theta}} | \psi_{\boldsymbol{\theta}} \rangle = \left\langle \psi_{\boldsymbol{\theta}} \middle| \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_j} \right\rangle + \left\langle \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_j} \middle| \psi_{\boldsymbol{\theta}} \right\rangle \tag{7.4.19}$$

$$\Rightarrow \left\langle \psi_{\boldsymbol{\theta}} \middle| \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_j} \right\rangle = -\left\langle \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_j} \middle| \psi_{\boldsymbol{\theta}} \right\rangle = -\left\langle \psi_{\boldsymbol{\theta}} \middle| \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_j} \right\rangle^*. \tag{7.4.20}$$

Under a change in global phase, we have $\beta_j \to \beta_j + \frac{\partial \alpha}{\partial \theta_j}$, which together with Eq. (7.4.18) implies that the real, symmetric, positive-definite tensor

$$g_{ij} = \gamma_{ij} - \beta_i \beta_j \tag{7.4.21}$$

$$= \mathrm{Re}\left[\left\langle \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_i} \middle| \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_j} \right\rangle\right] - \left\langle \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_i} \middle| \psi_{\boldsymbol{\theta}} \right\rangle \left\langle \psi_{\boldsymbol{\theta}} \middle| \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_j} \right\rangle \tag{7.4.22}$$

is invariant under changes in the global phase. The tensor g_{ij} is known as the FS metric.

An alternative way of obtaining the FS metric is to start from a distance measure that is already invariant under a change in global phases, such as 1 minus the fidelity:

$$ds^2 = 1 - |\langle \psi_{\boldsymbol{\theta}} | \psi_{\boldsymbol{\theta}+d\boldsymbol{\theta}} \rangle|^2 \tag{7.4.23}$$

$$= 1 - \left|1 + \left\langle \psi_{\boldsymbol{\theta}} \middle| \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_i} \right\rangle d\theta_i + \frac{1}{2} \left\langle \psi_{\boldsymbol{\theta}} \middle| \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_i \partial \theta_j} \right\rangle d\theta_i d\theta_j\right|^2 \tag{7.4.24}$$

$$= -\Bigg(\frac{1}{2}\left\langle \psi_{\boldsymbol{\theta}} \middle| \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_i \partial \theta_j} \right\rangle + \frac{1}{2}\left\langle \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_i \partial \theta_j} \middle| \psi_{\boldsymbol{\theta}} \right\rangle + \left\langle \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_i} \middle| \psi_{\boldsymbol{\theta}} \right\rangle \left\langle \psi_{\boldsymbol{\theta}} \middle| \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_j} \right\rangle \Bigg) d\theta_i d\theta_j \tag{7.4.25}$$

$$= \left(\mathrm{Re}\left[\left\langle \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_i}, \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_j} \right\rangle\right] - \left\langle \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_i} \middle| \psi_{\boldsymbol{\theta}} \right\rangle \left\langle \psi_{\boldsymbol{\theta}} \middle| \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_j} \right\rangle\right) d\theta_i d\theta_j, \tag{7.4.26}$$

where we used a second-order expansion and where the last step is due to the identities

$$\frac{\partial}{\partial \theta_i}\left\langle \psi_{\boldsymbol{\theta}} \middle| \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_j} \right\rangle = \left\langle \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_i} \middle| \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_j} \right\rangle + \left\langle \psi_{\boldsymbol{\theta}} \middle| \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_i \partial \theta_j} \right\rangle, \tag{7.4.27}$$

$$\frac{\partial}{\partial \theta_i}\left\langle \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_j} \middle| \psi_{\boldsymbol{\theta}} \right\rangle = \left\langle \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_i \partial \theta_j} \middle| \psi_{\boldsymbol{\theta}} \right\rangle + \left\langle \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_j} \middle| \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_i} \right\rangle, \tag{7.4.28}$$

and therefore, given that $\frac{\partial}{\partial \theta_i}\left\langle \psi_{\boldsymbol{\theta}} | \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_j} \right\rangle = -\frac{\partial}{\partial \theta_i}\left\langle \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_j} | \psi_{\boldsymbol{\theta}} \right\rangle$, if we add Eqs. (7.4.27) and (7.4.28), we obtain

$$\left\langle \psi_{\boldsymbol{\theta}} \middle| \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_i \partial \theta_j} \right\rangle + \left\langle \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_i \partial \theta_j} \middle| \psi_{\boldsymbol{\theta}} \right\rangle = -\left\langle \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_i} \middle| \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_j} \right\rangle - \left\langle \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_j} \middle| \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_i} \right\rangle$$

$$= -2\mathrm{Re}\left[\left\langle \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_i} \middle| \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_j} \right\rangle\right]. \tag{7.4.29}$$

7.4.2 Complex natural gradient descent algorithm on quantum optical circuits

We now generalize the FS metric to the case where the parametrization of ψ can involve a mix of real and complex parameters, and we apply it to the optimization of quantum optical circuits.

We will achieve our goal in two steps: first, we will convert complex parameters into their real and imaginary parts; second, we will reassemble the two parts by applying the rules of the Wirtinger calculus [152, 82, 153] to rewrite the Riemannian metric tensor in terms of derivatives with respect to complex parameters and their conjugate as *independent* variables. Using a complex version of the FS metric makes it straightforward to deal with non-holomorphic cost functions, as it removes the need to treat complex parameters as pairs of real parameters, leading to simpler and faster computations of the complex gradients.

Real and complex parameters

A complex parameter z and its conjugate z^* are related to the real and imaginary parts of z by a linear transformation $\boldsymbol{W}$:

$$\begin{pmatrix} z \\ z^* \end{pmatrix} = \begin{pmatrix} 1 & i \\ 1 & -i \end{pmatrix} \begin{pmatrix} z_R \\ z_I \end{pmatrix} = \boldsymbol{W} \begin{pmatrix} z_R \\ z_I \end{pmatrix} \tag{7.4.30}$$

$$\begin{pmatrix} z_R \\ z_I \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 & 1 \\ -i & i \end{pmatrix} \begin{pmatrix} z \\ z^* \end{pmatrix} = \boldsymbol{W}^{-1} \begin{pmatrix} z \\ z^* \end{pmatrix}, \tag{7.4.31}$$

where $z_R = \mathrm{Re}(z)$ and $z_I = \mathrm{Im}(z)$. Similarly, the gradients with respect to z and z^* and the gradients with respect to z_R and z_I are related by a linear transformation V, which we can find by applying the chain rule:

$$\frac{\partial \psi}{\partial z_R} = \frac{\partial \psi}{\partial z}\frac{\partial z}{\partial z_R} + \frac{\partial \psi}{\partial z^*}\frac{\partial z^*}{\partial z_R} = \frac{\partial \psi}{\partial z} + \frac{\partial \psi}{\partial z^*} \tag{7.4.32}$$

$$\frac{\partial \psi}{\partial z_I} = \frac{\partial \psi}{\partial z}\frac{\partial z}{\partial z_I} + \frac{\partial \psi}{\partial z^*}\frac{\partial z^*}{\partial z_I} = i\left(\frac{\partial \psi}{\partial z} - \frac{\partial \psi}{\partial z^*}\right), \tag{7.4.33}$$

i.e., $\partial_{z_R} = \partial_z + \partial_{z^*}$ and $\partial_{z_I} = i(\partial_z - \partial_{z^*})$. Conversely, $\partial_z = \frac{1}{2}(\partial_{z_R} - i\partial_{z_I})$ and $\partial_{z^*} = \frac{1}{2}(\partial_{z_R} + i\partial_{z_I})$, which means

$$\begin{pmatrix} \partial_z \\ \partial_{z^*} \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 & -i \\ 1 & i \end{pmatrix} \begin{pmatrix} \partial_{z_R} \\ \partial_{z_I} \end{pmatrix} = \boldsymbol{V} \begin{pmatrix} \partial_{z_R} \\ \partial_{z_I} \end{pmatrix} \tag{7.4.34}$$

$$\begin{pmatrix} \partial_{z_R} \\ \partial_{z_I} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ i & -i \end{pmatrix} \begin{pmatrix} \partial_z \\ \partial_{z^*} \end{pmatrix} = \boldsymbol{V}^{-1} \begin{pmatrix} \partial_z \\ \partial_{z^*} \end{pmatrix}. \tag{7.4.35}$$

In the next section, we will write $\boldsymbol{V}$ and $\boldsymbol{W}$ even for larger collections of parameters, with the understanding that $\boldsymbol{V}$ and $\boldsymbol{W}$ will be block-diagonal with 2×2 blocks for complex parameters and, if a parameter is real, its block will be 1×1 with a value of 1.

The FS metric for complex parameters

In this section, we will derive an expression for the *Hermitian* FS metric f for complex parameters (and by extension for a mix of parameters of any type).

Instead of writing the FS metric tensor using the real-part function (which is not holomorphic), we write it using the fact that the FS metric is the symmetric part of the Riemannian metric tensor G_{ij}, defined in Eq. (7.4.37):

$$g = \frac{G + G^T}{2}, \tag{7.4.36}$$

where

$$G_{ij} = \left\langle \frac{\partial \boldsymbol{\psi}}{\partial \theta_i} \middle| \frac{\partial \boldsymbol{\psi}}{\partial \theta_j} \right\rangle - \left\langle \frac{\partial \boldsymbol{\psi}}{\partial \theta_i} \middle| \boldsymbol{\psi} \right\rangle \left\langle \boldsymbol{\psi} \middle| \frac{\partial \boldsymbol{\psi}}{\partial \theta_j} \right\rangle. \tag{7.4.37}$$

Using such a linear relation between g and G, we can write the FS metric with respect to a mixture of real and complex parameters. We define $\boldsymbol{\xi}$ as our parameter vector, containing real parameters and complex parameters with the provision that for each complex parameter, we also include its complex conjugate. In this way, the all-real natural gradient update rule

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta g^{-1} \frac{\partial L}{\partial \boldsymbol{\theta}}, \tag{7.4.38}$$

turns into the more general update rule

$$\boldsymbol{\xi} \leftarrow \boldsymbol{\xi} - \eta f^{-1} \frac{\partial L}{\partial \boldsymbol{\xi}^*}, \tag{7.4.39}$$

which works for any type of parameter, real or complex.

Note that the natural gradient update rule differs from the general ones defined in section 4.5 by the inverse FS-metric factor f^{-1}.

We then show how to derive the general FS metric f in two steps. First, we write

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = \boldsymbol{V}^{-1^*} \frac{\partial L}{\partial \boldsymbol{\xi}^*} = \boldsymbol{W}^\dagger \frac{\partial L}{\partial \boldsymbol{\xi}^*}, \tag{7.4.40}$$

where we used the functional relation $\boldsymbol{V}^{-1} = \boldsymbol{W}^T$. Second, we transform the basis from $\boldsymbol{\theta}$ to $\boldsymbol{\xi}$ in Eq. (7.4.38) using Eq. (7.4.30) to obtain the gradient updates for $\boldsymbol{\xi}$:

$$\boldsymbol{\xi} \leftarrow \boldsymbol{\xi} - \epsilon \boldsymbol{W} g^{-1} \boldsymbol{W}^\dagger \frac{\partial L}{\partial \boldsymbol{\xi}^*}. \tag{7.4.41}$$

So we deduce by comparing (7.4.39) and (7.4.41) that the metric tensor with respect to the complex parameters is given by

$$f = (\boldsymbol{W} g^{-1} \boldsymbol{W}^\dagger)^{-1} = (\boldsymbol{W}^\dagger)^{-1} g \boldsymbol{W}^{-1} \tag{7.4.42}$$

$$= \frac{(\boldsymbol{W}^\dagger)^{-1} G \boldsymbol{W}^{-1} + (\boldsymbol{W}^\dagger)^{-1} G^T \boldsymbol{W}^{-1}}{2} \tag{7.4.43}$$

$$= \frac{\boldsymbol{V}^* G \boldsymbol{V}^T + \boldsymbol{V}^* G^T \boldsymbol{V}^T}{2}. \tag{7.4.44}$$

In the last step, we used the functional relation $\boldsymbol{W}^{-1} = \boldsymbol{V}^T$. Finally, we find the expression for f_{mn} by inserting Eq. (7.4.37) into Eq. (7.4.44):

$$\begin{aligned} f_{mn} = &\frac{1}{2} \left\langle V_{mi} \frac{\partial \boldsymbol{\psi}}{\partial \theta_i} \middle| V_{nj} \frac{\partial \boldsymbol{\psi}}{\partial \theta_j} \right\rangle - \frac{1}{2} \left\langle V_{mi} \frac{\partial \boldsymbol{\psi}}{\partial \theta_i} \middle| \boldsymbol{\psi} \right\rangle \left\langle \boldsymbol{\psi} \middle| V_{nj} \frac{\partial \boldsymbol{\psi}}{\partial \theta_j} \right\rangle \\ &+ \frac{1}{2} \left\langle V^*_{nj} \frac{\partial \boldsymbol{\psi}}{\partial \theta_j} \middle| W^*_{mi} \frac{\partial \boldsymbol{\psi}}{\partial \theta_i} \right\rangle - \frac{1}{2} \left\langle V^*_{nj} \frac{\partial \boldsymbol{\psi}}{\partial \theta_j} \middle| \boldsymbol{\psi} \right\rangle \left\langle \boldsymbol{\psi} \middle| W^*_{mi} \frac{\partial \boldsymbol{\psi}}{\partial \theta_i} \right\rangle \end{aligned} \tag{7.4.45}$$

$$\begin{aligned} = &\frac{1}{2} \left\langle \frac{\partial \boldsymbol{\psi}}{\partial \xi_m} \middle| \frac{\partial \boldsymbol{\psi}}{\partial \xi_n} \right\rangle - \frac{1}{2} \left\langle \frac{\partial \boldsymbol{\psi}}{\partial \xi_m} \middle| \boldsymbol{\psi} \right\rangle \left\langle \boldsymbol{\psi} \middle| \frac{\partial \boldsymbol{\psi}}{\partial \xi_n} \right\rangle \\ &+ \frac{1}{2} \left\langle \frac{\partial \boldsymbol{\psi}}{\partial \xi^*_n} \middle| \frac{\partial \boldsymbol{\psi}}{\partial \xi^*_m} \right\rangle - \frac{1}{2} \left\langle \frac{\partial \boldsymbol{\psi}}{\partial \xi^*_n} \middle| \boldsymbol{\psi} \right\rangle \left\langle \boldsymbol{\psi} \middle| \frac{\partial \boldsymbol{\psi}}{\partial \xi^*_m} \right\rangle. \end{aligned} \tag{7.4.46}$$

Note that for real parameters, the expression for f in Eq. (7.4.46) falls back to the usual FS metric. Also note that the tensor f is Hermitian rather than symmetric (i.e. $f^T = f^*$), which is to be expected, as the metric tensor of a complex manifold is Hermitian.

7.5 Conclusion

Thanks to the differentiability of our recurrence relation, this chapter showed us how to calculate the gradient in optical quantum circuits. Then we illustrated Euclidean and Riemannian optimizations of Gaussian operators. We also generalized the natural gradient algorithm to its complex version.

Our recurrence relation is differentiable, and we provide two ways to calculate its derivatives in order to finally derive the gradient in the optical quantum circuits: the chain rule and the generating function.

The generating function is a more direct method, while the chain rule allows us to get gradients in both optimizations.

Based on the circuits with the decomposed fundamental optical components, we developed Euclidean optimization, which optimizes each component in a Euclidean way. If we combine all the Gaussian operators as a global object, we can then optimize it on its Riemannian manifold and then decompose it back into fundamental optical components if we want to realize it in practice. This idea reduces the optimization time with fewer parameters and smaller parameter space; even with a circuit with hundreds of modes, the optimization works well. We found a geodesic-based Riemannian optimization method to update the real symplectic matrix on the Riemannian manifold, together with the unitary optimization for interferometers as a special case.

In addition, because our optical quantum circuit also has complex variables, we will generalize the Fubini-Study metric for complex parameters and derive the complex version of the natural gradient algorithm.

The next chapter provides some extra strategies for better using our recurrence relation for some particular tasks.

Chapter 8

Optimal task-based strategies for utilizing recurrence relation

8.1 Introduction

The recurrence relation of the Gaussian objects in chapter 6 can construct the elements of a tensor $\mathcal{G}_{\boldsymbol{k}}$ in Fock representation by linearly combining their neighbors. This chapter introduces various algorithms to use the recurrence relation in different situations.

We call default method to use the recurrence relation *vanilla version* and we explain it in section 8.2.

Here are the four new algorithms that we propose:

- Global and local cutoff algorithm in section 8.3: by defining the global cutoff G and local cutoffs $\boldsymbol{C} = [C_1, C_2 \dots]$ of the tensor, one can change the way of constructing the tensor. With the new algorithm, we can make the computation parallel and faster.

- Diagonal algorithm in section 8.4: to obtain the Fock amplitudes of a mixed state, we propose the diagonal algorithm, which requires calculating $\mathcal{O}(C^M)$ amplitudes compared with the vanilla version that computes $\mathcal{O}(C^{2M})$ amplitudes (assuming all cutoffs are C).

- Interferometer algorithm in section 8.5, we propose two algorithms in different cases to obtain the amplitudes $\boldsymbol{J}_k$ (in Eq. (3.49)) of an interferometer that conserves the total photon number. The first algorithm is under the assumptions that we can have at most one photon per mode, we already know the input and output pattern, and we want to calculate the corresponding probability: suppose that there are N photons injected in the interferometer and $N << M$, we need to compute $\mathcal{O}(2^N)$ amplitudes compared with the vanilla version with $\mathcal{O}(2^{2N})$. As for the second algorithm, we do not limit the number of photons on each mode, and we propose two versions to calculate the probability of multiple input and output patterns.

- Gaussian evolution algorithm in section 8.6: this algorithm gives a new recurrence relation for transformed states directly. Compared with the old matrix-vector multiplication method (usually used in evolution simulation), this new algorithm benefits both the forward pass and backward pass. The complexity scales $\mathcal{O}(C^M)$ rather than $\mathcal{O}(C^{2M})$.

8.2 Vanilla version of recurrence relation

We recall the recurrence relation defined in chapter 6:

$$\mathcal{G}_{\boldsymbol{k}+1_i} = \frac{1}{\sqrt{k_i+1}} \left(b_i \mathcal{G}_{\boldsymbol{k}} + \sum_j \sqrt{k_j} A_{ij} \mathcal{G}_{\boldsymbol{k}-1_j} \right), \tag{8.2.1}$$

with the vacuum amplitude initialized as $\mathcal{G}_{\boldsymbol{0}} = c$.

In Fig. 8.1, we show a tensor with two indices. The element $\mathcal{G}_{m,n+1}$ can be obtained by linearly combing the neighboring elements $\mathcal{G}_{m,n}$, $\mathcal{G}_{m-1,n}$ and $\mathcal{G}_{m,n-1}$. We can also find out that the elements $\mathcal{G}_{m+1,n}$ can be obtained with the same elements as the element $\mathcal{G}_{m,n+1}$ only with different coefficients $\boldsymbol{A}$ and $\boldsymbol{b}$.

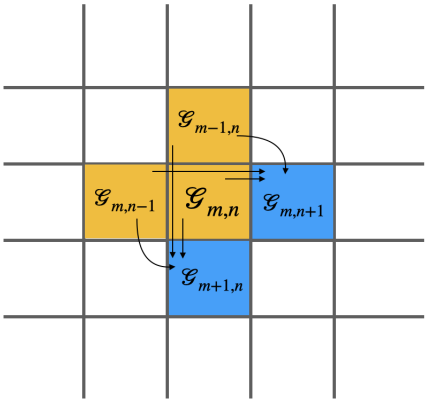


Figure 8.1: The vanilla recurrence relation. The two blue amplitudes can be computed by linearly combining the three orange ones.

This section first gives the graphical representation of the vanilla version of the recurrence relation and introduces some important definitions to understand our algorithms (local and global cutoffs, pivot read and write, and partition).

8.2.1 Cutoffs

For a Gaussian tensor $\mathcal{G}_{\boldsymbol{k}}$, the *local cutoffs* $\boldsymbol{C}$ are defined as

$$\boldsymbol{C} = [C_1, C_2, \ldots, C_\ell], \tag{8.2.2}$$

where ℓ is the length of index vector $\boldsymbol{k}$ and it is also the dimension of the tensor $\mathcal{G}$. In case we do not mention the explicit local cutoffs, we talk about the *equal local cutoffs* $\boldsymbol{C} = [C, C, \ldots, C]$.

In section 8.3, we will introduce a *global cutoff* G, which means that we will calculate the tensor with an index vector $\boldsymbol{k}$ where

$$\sum_i k_i < G. \tag{8.2.3}$$

8.2.2 Vanilla version

To fill the Gaussian tensor in Fock representation with the vanilla version of recurrence relation, we compute **one new item at each iteration**: we start from the first element with an index of all zeros and continue to add one more element on the last index until we reach the cutoff and move to the next index value.

Given an example of an index vector $\boldsymbol{k}$ with 3 indices and a cutoffs $\boldsymbol{C} = [2, 2, 2]$ of each index:

$$\begin{aligned}\boldsymbol{k} = (000) \to (001) \to (002) \to (010) \to (011) \to (012) \to (020) \to (021) \to (022) \to (100) \to (101) \to \\ (102) \to (110) \to (111) \to (112) \to (200) \to (201) \to (202) \to (210) \to (211) \to (212) \to \\ (220) \to (221) \to (222).\end{aligned} \tag{8.2.4}$$

Let us see another example of a matrix (a 2-dimensional tensor) and the equal local cutoffs $\boldsymbol{C} = [6, 6]$. The vanilla version of the recurrence relation is applied as follows: we fill the first row and then fill each row. Fig. 8.2 shows the filling process of the matrix.

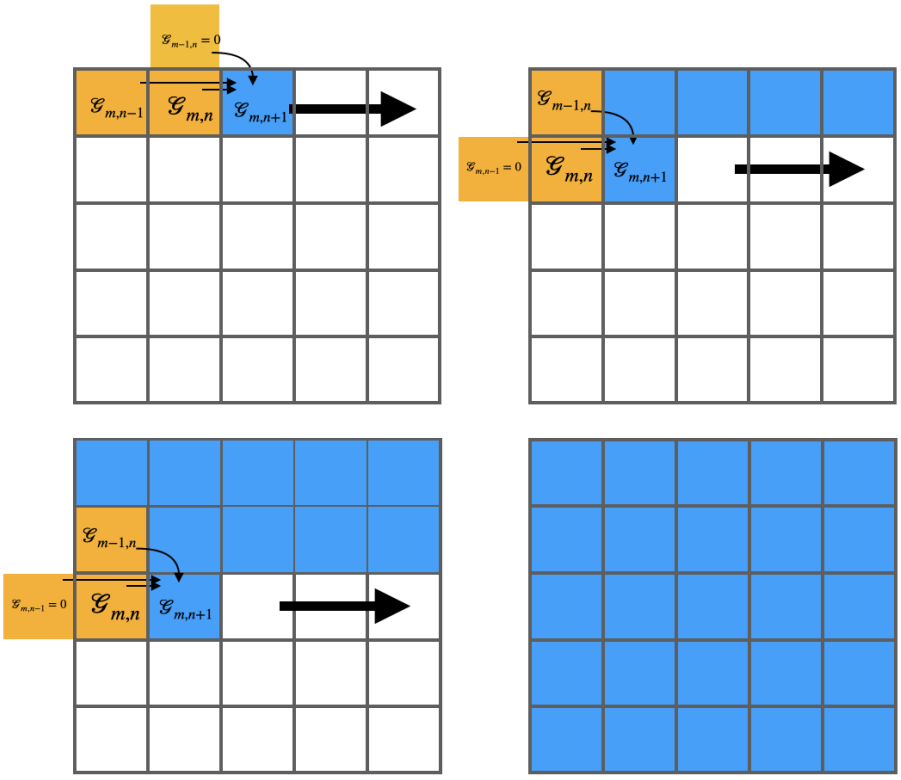


Figure 8.2: Vanilla version of recurrence relation with a 2-dimensional tensor. The elements are computed in the first row and fill each row in the vertical direction. The blue elements can be obtained by linearly combining the orange elements.

Given a ℓ-length index vector $\boldsymbol{k}$ and equal local cutoffs $\boldsymbol{C} = [C, C, \ldots, C]$, we need to compute C^k elements for the full tensor, which costs more computational resources and memory for the computation process with the increase of the length ℓ of index vector $\boldsymbol{k}$.

8.2.3 Pivot, read and write

In the vanilla version, we describe the combination of recurrence relation in a graphical version as in [35].

We introduce the new notations with the example of a 2-dimensional tensor: we call the elements (like $\mathcal{G}_{m,n}$) "pivot", the elements (like $\mathcal{G}_{m,n-1}$ and $\mathcal{G}_{m-1,n}$) "read" and the new elements ($\mathcal{G}_{m,n+1}$ and $\mathcal{G}_{m+1,n}$) "write", as shown in Fig. 8.3. So we say that when we have the orange elements "pivot", we can "write" the two green elements by "reading" the values of blue elements, which means that we can calculate the green elements from the orange and blue ones.

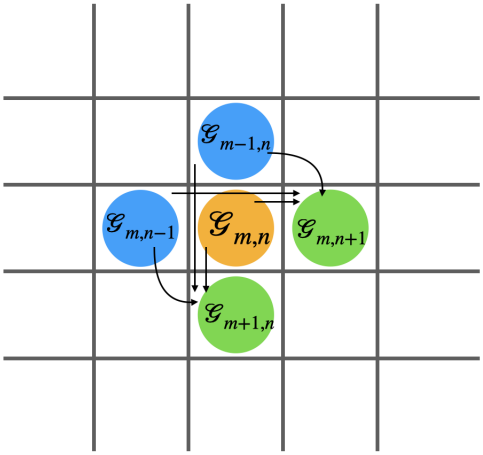


Figure 8.3: The pivot read and write in the vanilla version of the recurrence relation.

8.2.4 Partition

We define the *partition* of weight w as the collection of all strings $\boldsymbol{s}$ with length ℓ such that the sum of each element is the same w:

$$P_\ell(w) = \{\boldsymbol{s} | \sum_{i=0}^{\ell-1} s_i = w\}. \tag{8.2.5}$$

For example the partition of weight 2 in 4-mode is $P_4(2) = \{0002, 0020, 0200, 2000, 0011, 0101, 0110, 1100\}$.

8.3 Global and local cutoff algorithm

In the vanilla version, the recurrence relation is used to calculate one element at a time. Then we try to express the relation in the way of "pivot, read and write". This new pattern gives us a new point of view to look into the recurrence relation, and we came up with the idea of grouping elements with the same combination pattern. We notice that in Fig. 8.4, all pivots, reads, and writes have the same pattern: the sum of their two indices is equal ($m + n + 1$ for both), which is defined as *weight* in the following.

Suppose that we have a Gaussian tensor with an index vector $\boldsymbol{k}$ of length ℓ. We define the weight of the index vector $\boldsymbol{k}$ as the sum of its coordinates:

$$w(\boldsymbol{k}) = \sum_{i=0}^{\ell-1} k_i. \tag{8.3.1}$$

$w(\boldsymbol{k})$ corresponds to the total photon number of a basis vector if the index vector refers to a ket, such as $|\boldsymbol{k}\rangle$. The set of index vectors of weights w and length ℓ is the partition $P_\ell(w)$.

The global and local cutoff algorithm is shown in Algo.1. We define a global cutoff G and local cutoffs $\boldsymbol{C} = [C_1, C_2, \ldots, C_\ell]$ and start from the first element $\mathcal{G}_{\boldsymbol{0}} = c$. We progressively calculate all the tensor elements with a constant weight w. Each time we pick a pivot from the partition $P_\ell(w)$, which connects "read" elements in partition $P_\ell(w-1)$ and produces "write" elements $P_\ell(w+1)$. This process can also be viewed as we pick the pivots with weight w, we already have computed all the elements with a lower weight $w-1$, and we are calculating the elements with higher weights $w+1$. During the calculation, the index vector can also satisfy the local cutoffs $\boldsymbol{C}$.

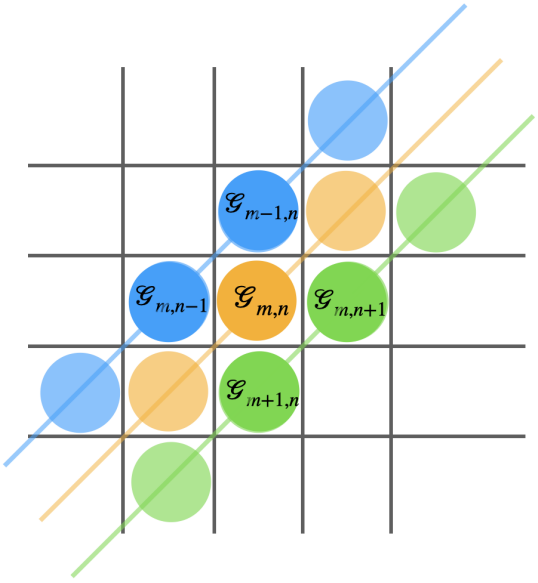


Figure 8.4: Grouping tensor elements with index vector of the same weight.

Algorithm 1 Global and local cutoff algorithm

initialize the global cutoff G and local cutoffs $\boldsymbol{C} = [C_1, C_2, \ldots, C_\ell]$.
initialize the pivot as $\mathbf{0}_{\boldsymbol{k}} = c$.
for $w \leftarrow 1$ to G **do**
    calculate the partition $P_\ell(w)$ of all length-ℓ indices.
    for *ind* in $P_\ell(w)$ **do**
        if *ind* satisfies local cutoffs $\boldsymbol{C}$ **then**
            take *ind* as pivot and calculate all "write" elements.
        end if
    end for
end for

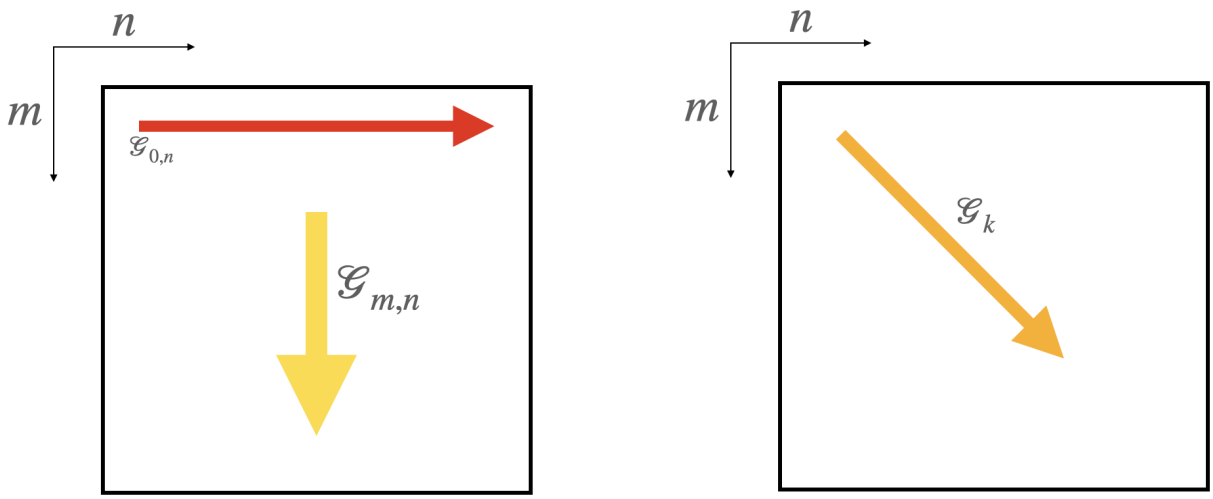


Figure 8.5: Visualization of two ways of using the recurrence relation. On the left we fill the tensor along the m and n indices. On the right we fill it by computing all the indices with equal weight.

The difference between this global and local cutoff algorithm and our vanilla version is shown in Fig. 8.5 with an example of 2-dimensional tensor. With the definition of recurrence relation, we have:

$$\mathcal{G}_{0,n} = \frac{b_2}{\sqrt{n}}\mathcal{G}_{0,n-1} + A_{22}\sqrt{\frac{n-1}{n}}\mathcal{G}_{0,n-2}, \tag{8.3.2}$$

$$\mathcal{G}_{m,n} = \frac{b_1}{\sqrt{m}}\mathcal{G}_{m-1,n} + A_{11}\sqrt{\frac{m-1}{m}}\mathcal{G}_{m-2,n} + A_{12}\frac{1}{\sqrt{m}}\mathcal{G}_{m-1,n-1}. \tag{8.3.3}$$

We need to calculate the first row of the tensor elements and then we can derive all the others. However, if we derive the tensor elements with the increase in weight, each time we calculate all elements in one off-diagonal. Hence, the tensor elements would be computed in the diagonal direction.

When $w = 0$, the first element $\mathcal{G}_{\mathbf{0}}$ is obtained. When $w = 1$, we would obtain $2M$ elements having only a "1": $\mathcal{G}_{1000\ldots}, \mathcal{G}_{0100\ldots}, \mathcal{G}_{0010\ldots}, \ldots$. The number of elements sharing the same weight increases binomially. This scheme also looks like a tree structure, where each weight value corresponds to a layer. A three-mode tree is shown in Fig. 8.6. We show the partition $P_3(w)$ corresponding to increasing weight w.

8.3.1 Parallelization of the calculation

With the new global and local cutoff algorithm, we increase the weight each round; all elements sharing the same weight can be calculated in parallel. For example, in Fig. 8.6, the weight 3 has ten independent elements that can be obtained simultaneously. The higher weight and more elements we have, the more beneficial this parallelization is.

8.3.2 Norm bound

The idea here is to use bounds related to the physical meaning of the elements we are calculating. For instance, if we are calculating a normalized state, the tensor elements represent the Fock amplitudes, and we can get the norm of the state by adding them up.

Before the calculation, we set a norm bound of the state we want to achieve, for example, 0.99. Then, while computing the elements by increasing the index vector weight w, we could sum up all the elements in the tensor and evaluate it. If it is at least 0.99, we will stop the calculations of the elements in the tensor.

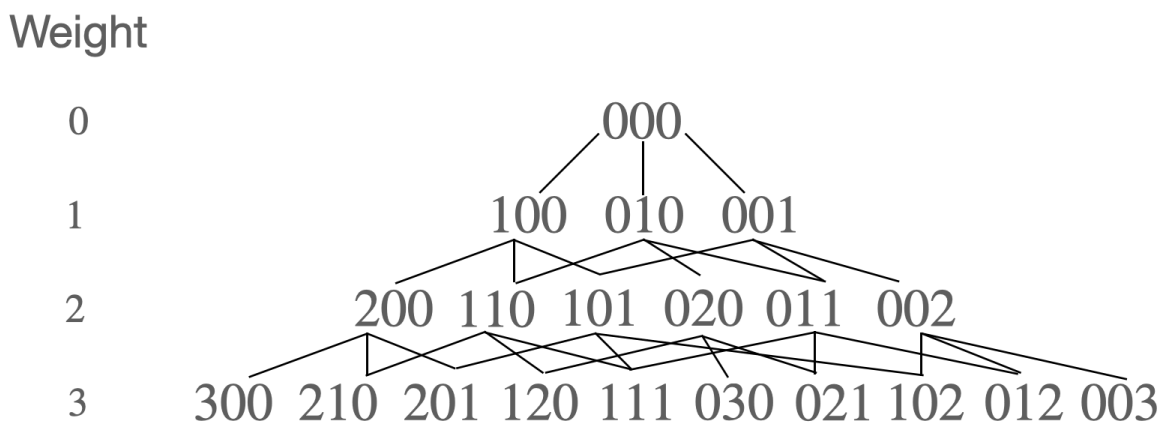


Figure 8.6: Three-mode tree of the recurrence relation elements.

Considering the single-mode case, the tensor would be a matrix (shown in Fig. 8.7). The left upper part of the matrix corresponds to low energy (or weight w), while the right bottom part corresponds to high energy. The matrix elements are computed in the direction of the diagonal, top to down. After filling the matrix for each weight, the norm is calculated. When the norm is higher than the bound, such as 0.99, the other parts in the matrix remain 0.

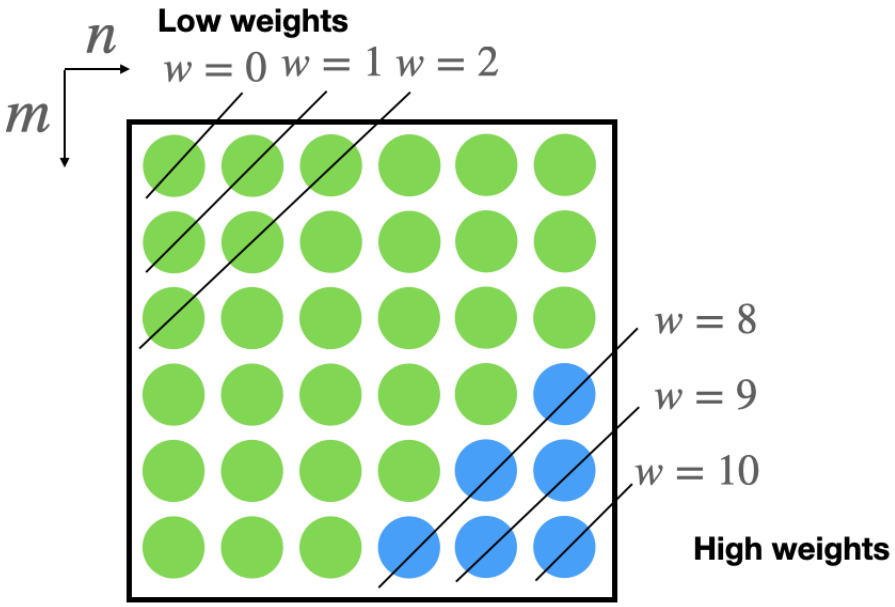


Figure 8.7: Example of using the norm bound to stop the calculation. The sum of elements corresponding to green points is enough to reach a high norm, and there is no need to compute the remaining elements corresponding to blue points.

8.3.3 No displacement

If the Gaussian tensor is a Gaussian unitary, the Gaussian object parameters have special meaning: $\boldsymbol{b}$ is related to the displacement operator, and $\boldsymbol{A}$ is related to the interferometer and squeezing operator.

In case that there is no displacement, $\boldsymbol{b} = 0$, our recurrence relation becomes:

$$\mathcal{G}_{\mathbf{0}} = C, \tag{8.3.4}$$

$$\mathcal{G}_{\boldsymbol{k}+1_i} = \frac{1}{\sqrt{k_i+1}} \sum_{l=1}^{2\ell} \sqrt{k_l}\mathcal{G}_{\boldsymbol{k}-1_l}\boldsymbol{A}_{il}. \tag{8.3.5}$$

We find that we skip the odd parity indices and the filling of the tensor requires half the steps.

8.4 Diagonal algorithm for mixed states

If the Gaussian tensor refers to a Gaussian state, especially if it is a mixed state, the Fock probability (amplitude) of the state depends on the main diagonal of the tensor.

To better explain our algorithm, we define a diagonal number k of a matrix (m, n) as

- $k = 0$, 0-diagonal represents the main diagonal;
- $k > 0$, k-diagonal represents the elements $(0, k), (1, k+1), (2, k+2), \ldots$ above the main diagonal;
- $k < 0$, k-diagonal represents the elements $(k, 0), (k+1, 1), (k+2, 2), \ldots$ below the main diagonal.

Fig. 8.8 explains k-diagonal.

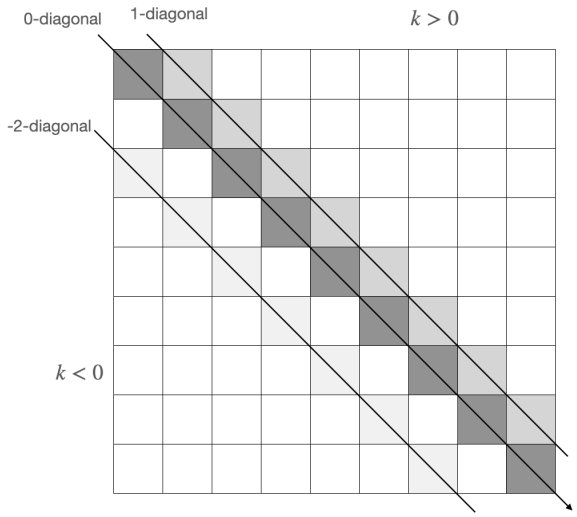


Figure 8.8: The definition of k-diagonal.

Since we are only interested in the elements on the diagonals of the tensor, it is not necessary to fill the full tensor. If we consider the main diagonal as the pivot, we can get ± 1-diagonals and the main diagonal. However, to get ± 1-diagonals, one needs one of ± 2-diagonals as "reads". Hence, to get all the elements in the main diagonal, we have the main diagonal and 1-diagonals as "pivots" as shown in Fig. 8.9.

The corresponding algorithm to calculate all diagonals is written in Alg.2.

A special case to consider is when there is no displacement in the operator ($\boldsymbol{b} = \mathbf{0}$). The computation pattern simplifies to require only one pivot for each run, which is shown in Fig. 8.10.

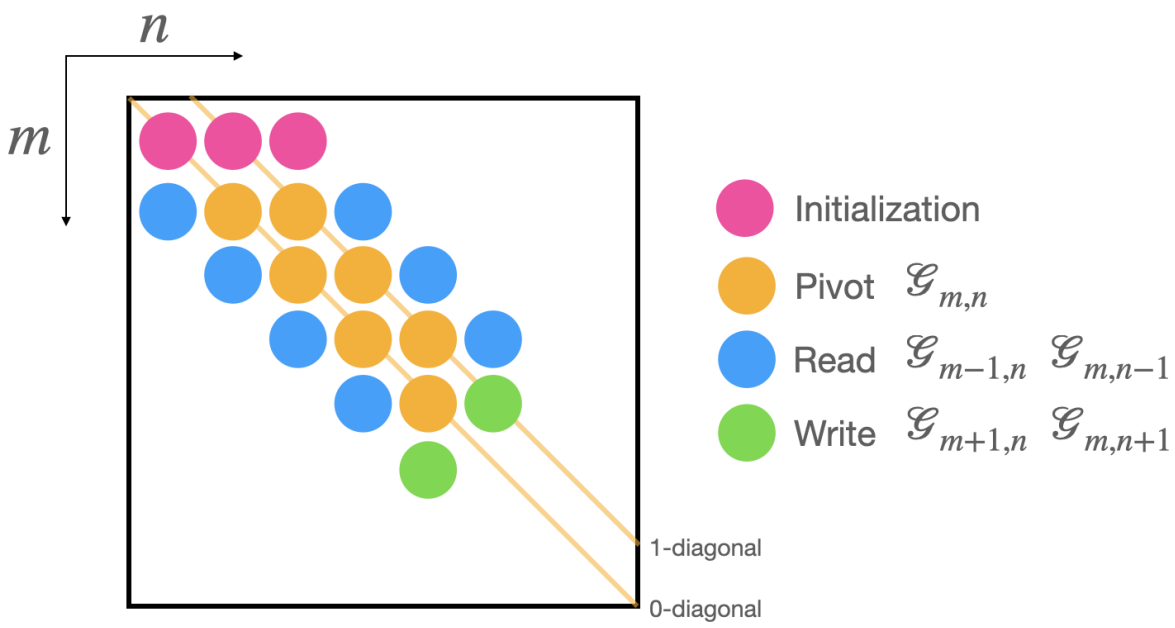


Figure 8.9: Computation pattern for single-mode mixed states. We aim for the elements on the main diagonal. We need two diagonals as pivots to get all of them: the main diagonal and 1-diagonal.

Algorithm 2 Diagonals algorithm

initialize the pivot as $\mathbf{0}_{2M}$ with $w(pivot) = 0$.
initialize the vector $\boldsymbol{C}$ as *cutoffs* on each mode, and hence the max weight that needs to be computed as $W = \sum_i C_i$.
for $w \leftarrow 0$ to W **do**
    calculate the set *diag_set* of all length-$2M$ indices $(a, a, b, b, c, c, ...)$ that satisfy $a + b + c + ... = sum$ and $(a, b, c, ...) < cutoffs$
    for *diag* in *diag_set* **do**
        apply *diag* as pivot
        for $d \leftarrow 1$ to M **do**
            if the first $2(d-1)$ elements of *diag* are 0 **then**
                apply $diag + 1_{2d-1}$ as pivot
            end if
        end for
    end for
end for

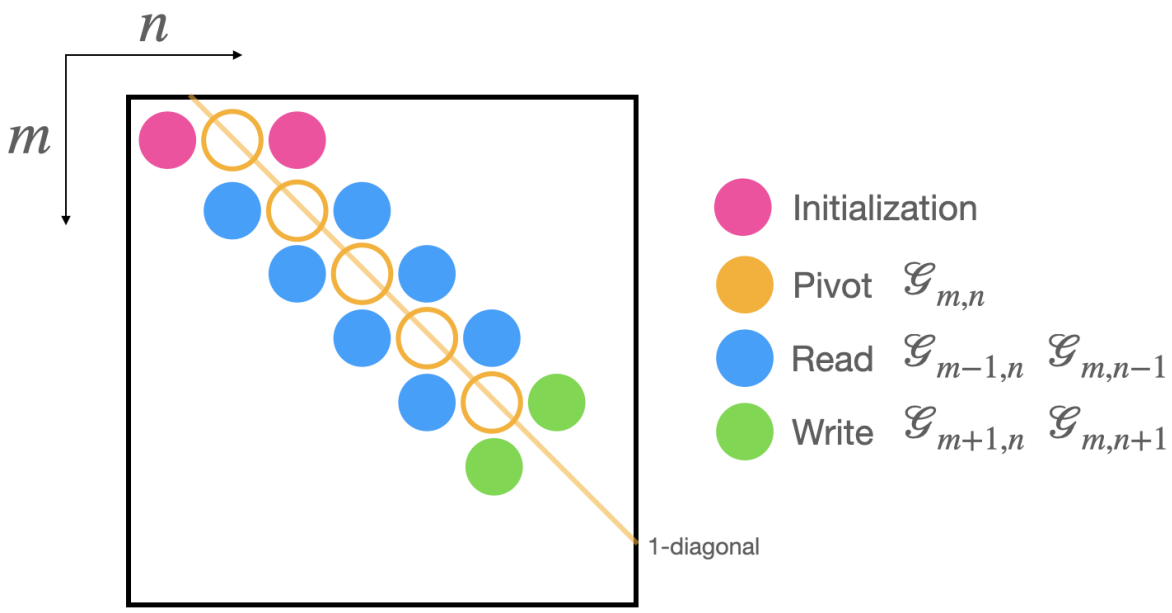


Figure 8.10: Computation pattern for single-mode mixed states without displacement. We aim for the elements on the main diagonal. Without displacement, only pivots along the 1-diagonal are picked (and their value is zero, so they don't even contribute).

Discussion for the complexity. We can say that we need two diagonals to calculate the next one which defines the depth of our recurrence relation to be equal to 2. With the increasing number of modes, we need more k-diagonals to be filled to get all the elements in the main diagonal. The increasing complexity is linear with the number of elements in the main diagonal $\mathcal{O}((2M+1)C^M) \approx \mathcal{O}(C^M)$. This is a quadratic improvement compared with the vanilla version $\mathcal{O}(C^{2M})$.

8.5 Interferometer algorithms

If the Gaussian unitary is an interferometer, we have the following:

$$c = 1, \tag{8.5.1}$$

$$\boldsymbol{b} = \boldsymbol{0}, \tag{8.5.2}$$

$$\boldsymbol{A} = \left(\begin{bmatrix} 0 & \boldsymbol{V} \\ \boldsymbol{V}^T & 0 \end{bmatrix} \right). \tag{8.5.3}$$

Accordingly, the recurrence relation becomes:

$$U_{\boldsymbol{m}+1_i,\boldsymbol{n}} = \frac{1}{\sqrt{m_i+1}} \sum_{p=1}^{M} \sqrt{n_p} V_{ip} U_{\boldsymbol{m},\boldsymbol{n}-1_p}, \tag{8.5.4}$$

$$U_{\boldsymbol{m},\boldsymbol{n}+1_i} = \frac{1}{\sqrt{n_i+1}} \sum_{p=1}^{M} \sqrt{m_p} V_{pi} U_{\boldsymbol{m}-1_p,\boldsymbol{n}}. \tag{8.5.5}$$

The main idea is that the interferometer conserves photon numbers; hence we are calculating the amplitudes with the index vector $\boldsymbol{k} = [\boldsymbol{m}, \boldsymbol{n}]$ and $w(\boldsymbol{m}) = w(\boldsymbol{n})$.

At most 1 photon on each mode

We adopt a helpful notation where we specify which indices have a 1, e.g., $\boldsymbol{k} = 010011100 \to \boldsymbol{s} = [2, 5, 6, 7]$. We will use this notation for the indices of U as well.

In our algorithm, we begin from the vacuum to vacuum amplitude $U_{[],[]} = 1$, and we build our way toward $U_{\boldsymbol{s}^{(\text{out})},\boldsymbol{s}^{(\text{in})}}$. At each step, we add one new photon to the output where we expect one (according to the target specification), and we apply the recurrence formula. In the formula, we consider 1_i to be the newly added output photon and $\boldsymbol{k}^{(\text{in})}$ to be all the possible inputs compatible with the target input state that generate it.

We exemplify this procedure by computing the amplitude for transforming $|111000\rangle = [1, 2, 3]$ to $|101010\rangle = [1, 3, 5]$. Notice that at each step, we only need amplitudes that have already been computed at the previous step.

step 0 (vacuum to vacuum) : $U_{[],[]} = 1$.

step 1 ($\boldsymbol{k}^{(\text{out})} = []$, $i = 1$) :

$$U_{[1],[1]} = V_{11}, \tag{8.5.6}$$
$$U_{[1],[2]} = V_{12}, \tag{8.5.7}$$
$$U_{[1],[3]} = V_{13}. \tag{8.5.8}$$

step 2 ($\boldsymbol{k}^{(\text{out})} = [1]$, $i = 3$) :

$$U_{[13],[12]} = V_{32}U_{[1],[1]} + V_{31}U_{[1],[2]}, \tag{8.5.9}$$
$$U_{[13],[13]} = V_{33}U_{[1],[1]} + V_{31}U_{[1],[3]}, \tag{8.5.10}$$
$$U_{[13],[23]} = V_{33}U_{[1],[2]} + V_{32}U_{[1],[3]}. \tag{8.5.11}$$

step 3 ($\boldsymbol{k}^{(\text{out})} = [13]$, $i = 5$) :

$$U_{[135],[123]} = V_{53}U_{[13],[12]} + V_{52}U_{[13],[13]} + V_{51}U_{[13],[23]}. \tag{8.5.12}$$

This algorithm can be represented in the tree scheme as well. In Fig. 8.11, we have a 6-mode interferometer.

Normally, we need many more elements to be computed for each weight level; however, as for the interferometer (which holds the same photon number at the input and the output), the tree has degenerated into a much easier one.

For an m-photons input-output target on M modes, we need to compute $\binom{m}{k}$ amplitudes at step k by summing k elements each and therefore $\sum_{k=0}^{m}\binom{m}{k} = 2^m$ amplitudes. Interestingly, this number does not depend on M.

Compared with the vanilla version with a complexity $\mathcal{O}(2^{2m})$, we only need $\mathcal{O}(2^m)$.

Multiple inputs and outputs

Sometimes we will want to optimize for multiple input and output states. This is the case, for example, when we have an input/output state that is not an eigenstate of the number operator in each mode. In order to minimize the runtime and memory use, we adopt a tree strategy.

Ultimately, for a superposition of m inputs and n outputs, we will have to compute mn amplitudes, but we can be more clever than simply re-running the algorithm mn times.

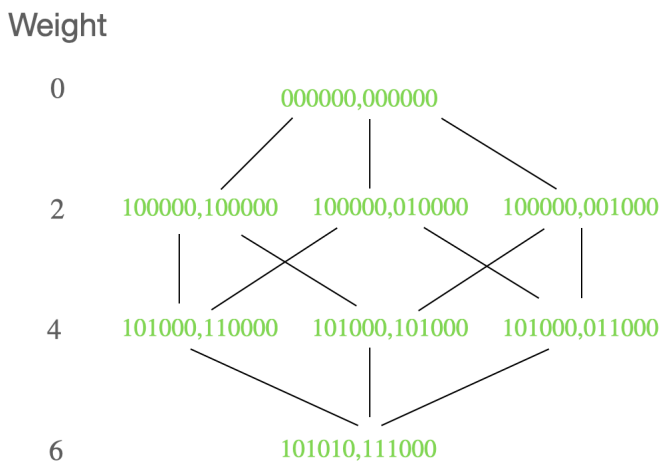


Figure 8.11: Interferometer weight tree scheme.

Multiple outputs If we have the same input pattern $\boldsymbol{s}^{(\text{in})}$ going into multiple outputs $\boldsymbol{s}_i^{(\text{out})}$, we could run the same algorithm from top to bottom several times, generating all the necessary amplitudes, but this could be very wasteful. A better solution is to reuse what we have already computed.

Following the example above, if we wanted to compute the output $[1, 3, 6]$ after the previous computation is over, we would only need to redo the last step and obtain $U_{[136],[123]} = V_{63}U_{[13],[12]} + V_{62}U_{[13],[13]} + V_{61}U_{[13],[23]}$, i.e., the amplitudes computed in step 2 would be reused. This is because the only difference is one output photon. If two photons were to differ (e.g. $[1, 2, 6]$), then we would need to go back two steps, etc... The scheme is shown in Fig. 8.12. So, in essence, for each output pattern $\boldsymbol{s}_i^{(\text{out})}$, we only need to go back until we have found the most similar pattern that was already computed and restart from there instead of restarting from scratch.

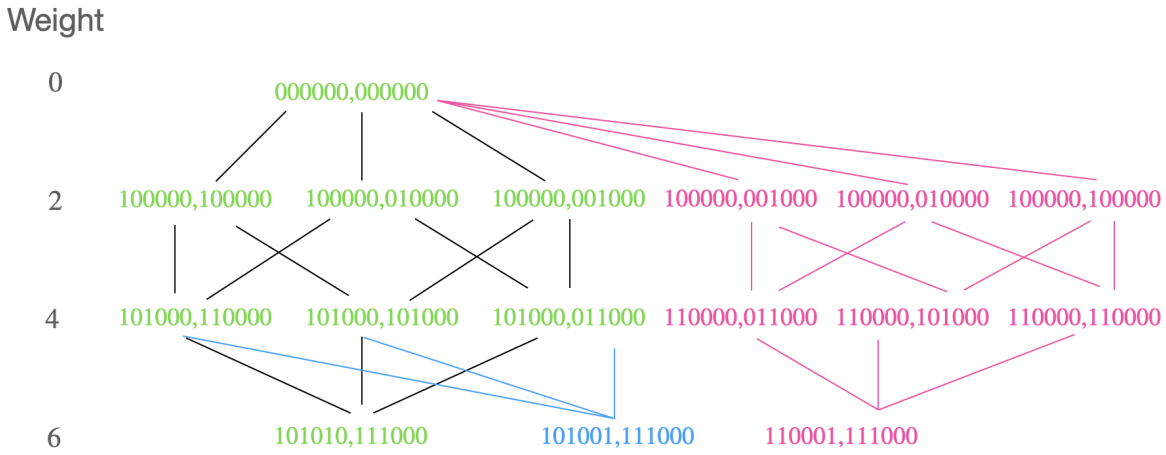


Figure 8.12: Multiple outputs tree scheme.

Algorithm 3 Multiple outputs algorithm

the input pattern is $\boldsymbol{s}^{(\text{in})}$ and the multiple output patterns are $\{\boldsymbol{s}^{(\text{out})}\}$, combining the input and output pattern, they all have the same weight w.
initialize the sets of weights $W(0), W(2), \ldots, W(w-2)$ as vide, and $W(w)$ contains the multiple output patterns $\{\boldsymbol{s}^{(\text{out})}\}$ with the fixed input pattern.
for $\boldsymbol{s}_{\text{now}} \in \{W(w)\}$ **do**
    CalculatePattern($\boldsymbol{s}_{\text{now}}$).
end for
procedure CalculatePattern($\boldsymbol{s}$)
    $w_{\text{now}} = \sum_i s_i$
    if $w_{\text{now}} == 0$ **then**
        return 1.
    end if
    calculate patterns $\boldsymbol{s}'$ with weight $w_{\text{now}} - 2$ and corresponds to $\boldsymbol{s}$.
    for $\boldsymbol{s}'_i \in \boldsymbol{s}'$ **do**
        if $\boldsymbol{s}'_i$ not in $W(w-2)$ **then** ▷ here is the reuse of strings that have been calculated!
            return $c_{\boldsymbol{s}'_i,\boldsymbol{s}}$*CalculatePattern($\boldsymbol{s}'_i$). ▷ $c_{\boldsymbol{s}'_i,\boldsymbol{s}}$ is the coefficient between the two strings.
        else
            return the value of $\boldsymbol{s}'_i$.
        end if
    end for
end procedure

Multiple inputs Suppose we have multiple inputs $\boldsymbol{s}_i^{(\text{in})}$ and a fixed output $\boldsymbol{s}^{(\text{out})}$, it holds the same strategy if we change the label of *in* and *out*.

In conclusion, with the tree strategy, for a superposition of m inputs and n outputs, we will have to compute mn times, however, the recursive algorithm stops when the result is already in the sets. The complexity of this algorithm depends largely on the similarities between the set of inputs and the set of outputs.

8.6 Gaussian evolution algorithm

As for the previous way to evolve a quantum state under Gaussian operators $\mathcal{G}$, we need to fill the Gaussian transformation matrix recursively (the recursive relation in chapter 6 for generating the matrix elements of a Gaussian transformation in the Fock space representation) and then take the matrix-vector product $\mathcal{G}|\psi\rangle$ with the input vector $|\psi\rangle$. Note that if the Gaussian operator is multimode, the matrix-vector product turns to the contraction of tensors.

Here we present a Gaussian evolution algorithm that directly computes the transformed state under Gaussian operators without generating the full transformation matrix, which works by "fusing" the matrix generation step and inner product step.

Based on our recurrence relation in Chapter 6, this new algorithm is differentiable, which means we can use it in conjunction with gradient-based optimizers for circuit optimization tasks.

Algorithm 4 Multiple inputs algorithm

the output pattern is $\boldsymbol{s}^{(\text{out})}$ and the multiple input patterns are $\{\boldsymbol{s}^{(\text{in})}\}$, combining the input and output pattern, they all have the same weight w.
initialize the sets of weights $W(0), W(2), \ldots, W(w-2)$ as vide, and $W(w)$ contains the multiple output patterns $\{\boldsymbol{s}^{(\text{out})}\}$ with the fixed input pattern.
for $\boldsymbol{s}_{\text{now}} \in \{W(w)\}$ **do**
    CalculatePattern($\boldsymbol{s}_{\text{now}}$).
end for
procedure CalculatePattern($\boldsymbol{s}$)
    $w_{\text{now}} = \sum_i s_i$
    if $w_{\text{now}} == 0$ **then**
        return 1.
    end if
    calculate patterns $\boldsymbol{s}'$ with weight $w_{\text{now}} - 2$ and corresponds to $\boldsymbol{s}$.
    for $\boldsymbol{s}'_i \in \boldsymbol{s}'$ **do**
        if $\boldsymbol{s}'_i$ not in $W(w-2)$ **then** ▷ here is the reuse of strings that have been calculated
            return CalculatePattern($\boldsymbol{s}'_i$).
        else
            return the value of $\boldsymbol{s}'_i$.
        end if
    end for
end procedure

8.6.1 Forward pass

General Gaussian transformed state

Recall the recursive relation in Eq. (6.2.11) :

$$\mathcal{G}_{\boldsymbol{k}+1_i} = \frac{1}{\sqrt{k_i+1}} \left(\mathcal{G}_{\boldsymbol{k}} \mu_i - \sum_{l=1}^{2M} \sqrt{k_l} \mathcal{G}_{\boldsymbol{k}-1_l} \Sigma_{il} \right).$$

When the tensor refers to Gaussian operators, the index $\boldsymbol{k}$ can be split into two parts: output indices $\boldsymbol{m}$ and input indices $\boldsymbol{n}$. We also need to recall our definition of $\boldsymbol{k} + 1_i$, which means to increase 1 on the i-th index of vector $\boldsymbol{k}$.

The following identities hold:

$$\mathcal{G}_{\boldsymbol{m},\boldsymbol{n}-1_i} \sqrt{n_i} = \mathcal{G}_{\boldsymbol{m},\boldsymbol{n}} a_i, \tag{8.6.1}$$

where a_i is the annihilation operator acting on i-th input indices $\boldsymbol{n}$. We can then write the annihilation operator vector as

$$\boldsymbol{a}_{\boldsymbol{k}} = (a_1, \ldots, a_M), \tag{8.6.2}$$

which indicates whether there is an annihilation operator on each index of $\boldsymbol{n}$.

We write down Eq. (6.2.11) together with Eq. (8.6.1):

$$\mathcal{G}_{\boldsymbol{m},\boldsymbol{n}} = \frac{\mu_i}{\sqrt{m_i}} \mathcal{G}_{\boldsymbol{m}-1_i,n} - \sum_{l=1}^{M} \frac{\Sigma_{il}}{\sqrt{m_i}} \sqrt{(\boldsymbol{m}-1_i)_l} \mathcal{G}_{\boldsymbol{m}-\boldsymbol{1_i}-\boldsymbol{1_l},\boldsymbol{n}} - \sum_{l=M}^{2M} \frac{\Sigma_{il}}{\sqrt{m_i}} \mathcal{G}_{\boldsymbol{m}-1_i,\boldsymbol{n}} a_{l-M}. \tag{8.6.3}$$

The input state in our algorithm can be any multi-mode pure state, and we use the notation $\psi_{\boldsymbol{n}}$ in this section to take it as a n-dimensional tensor.

Now we can define a new tensor $R^{\boldsymbol{k}}_{\boldsymbol{m}}$ by sandwiching the annihilation operator vector $\boldsymbol{a_k}$ with the input state tensor $\psi_{\boldsymbol{n}}$ and Gaussian operator tensor $\mathcal{G}_{\boldsymbol{m},\boldsymbol{n}}$:

$$R^{\boldsymbol{k}}_{\boldsymbol{m}} = \mathcal{G}_{\boldsymbol{m},\boldsymbol{n}} \boldsymbol{a_k} \psi_{\boldsymbol{n}}, \tag{8.6.4}$$

where we adopt Einstein's summation convention.

We can then try to express this new tensor with the help of our recurrence relation:

$$\underbrace{\mathcal{G}_{\boldsymbol{m},\boldsymbol{n}}\psi_{\boldsymbol{n}}}_{R^{\boldsymbol{0}}_{\boldsymbol{m}}} = \frac{\mu_i}{\sqrt{m_i}} \underbrace{\mathcal{G}_{\boldsymbol{m}-\boldsymbol{1}_i,n}\psi_{\boldsymbol{n}}}_{R^{\boldsymbol{0}}_{\boldsymbol{m}-\boldsymbol{1}_i}} - \sum_{l=1}^{M} \frac{\Sigma_{il}}{\sqrt{m_i}} \sqrt{(\boldsymbol{m}-\boldsymbol{1}_i)_l} \underbrace{\mathcal{G}_{\boldsymbol{m}-\boldsymbol{1}_i-\boldsymbol{1}_l,n}\psi_{\boldsymbol{n}}}_{R^{\boldsymbol{0}}_{\boldsymbol{m}-\boldsymbol{1}_i-\boldsymbol{1}_l}} - \sum_{l=M}^{2M} \frac{\Sigma_{il}}{\sqrt{m_i}} \underbrace{\mathcal{G}_{\boldsymbol{m}-\boldsymbol{1}_i,n}\boldsymbol{a}_{l-M}\psi_{\boldsymbol{n}}}_{R^{\boldsymbol{1}_{l-M}}_{\boldsymbol{m}-\boldsymbol{1}_i}}. \tag{8.6.5}$$

The relation holds for any $\boldsymbol{k}$, so we get:

$$R^{\boldsymbol{k}}_{\boldsymbol{m}} = \frac{\mu_i}{\sqrt{m_i}} R^{\boldsymbol{k}}_{\boldsymbol{m}-\boldsymbol{1}_i} - \sum_{l=1}^{M} \frac{\Sigma_{il}}{\sqrt{m_i}} \sqrt{(\boldsymbol{m}-\boldsymbol{1}_i)_l} R^{\boldsymbol{k}}_{\boldsymbol{m}-\boldsymbol{1}_i-\boldsymbol{1}_l} - \sum_{l=M}^{2M} \frac{\Sigma_{il}}{\sqrt{m_i}} R^{\boldsymbol{k}+\boldsymbol{1}_{l-M}}_{\boldsymbol{m}-\boldsymbol{1}_i}. \tag{8.6.6}$$

Here Eq. (8.6.6) is our main result and the transformed state is $R^{\boldsymbol{0}}_{\boldsymbol{m}}$. This equation explains that to get the elements in the tensor $R^{\boldsymbol{k}}_{\boldsymbol{m}}$, one needs to combine its neighboring elements from the previous level of $\boldsymbol{m}-\boldsymbol{1_i}$ and $\boldsymbol{m}-\boldsymbol{1_i}-\boldsymbol{1_j}$ with the same $\boldsymbol{k}$ or $\boldsymbol{k}+1$. Because of this kind of dependence, it is not necessary to calculate the entire tensor to get the transformed state finally.

In the next section, we'll elaborate on how this R tensor could be used in single-mode and two-mode cases. We will also discuss the complexity reduction for each case.

Single-mode transformed state

A single-mode Gaussian transformation can be expressed as a 2-dimensional tensor $\mathcal{G}_{m,n}$. As explained in the previous section, to fill in the full tensor, we need to first use Eq. (8.6.5) and then in Eq. (8.6.25). For the single-mode case, according to Eq. (6.2.11), the first row is given by:

$$\mathcal{G}_{0,n} = \frac{\mu_2}{\sqrt{n}} \mathcal{G}_{0,n-1} - \Sigma_{22} \sqrt{\frac{n-1}{n}} \mathcal{G}_{0,n-2}, \tag{8.6.7}$$

where the first element is $\mathcal{G}_{0,0} = C$.

By using Eq. (6.2.11) again, each successive row of the Gaussian transformation can be computed recursively:

$$\mathcal{G}_{m,n} = \frac{\mu_1}{\sqrt{m}} \mathcal{G}_{m-1,n} - \Sigma_{11} \sqrt{\frac{m-1}{m}} \mathcal{G}_{m-2,n} - \frac{\Sigma_{12}}{\sqrt{m}} \mathcal{G}_{m-1,n} a, \tag{8.6.8}$$

where a is the annihilation operator acting on the index n (in the single-mode case, there is only one index of $\boldsymbol{n}$).

Given a vector of the input state ψ_n, the amplitude of the output state is given by $\mathcal{G}_{m,n}\psi_n$. Therefore, using the recurrence relation for the rows in Eq. (8.6.8), we can write it for the single-mode case by using the definition of R^k_m in Eq. (8.6.4):

$$\underbrace{\mathcal{G}_{m,n}\psi_n}_{R^0_m} = \frac{\mu_1}{\sqrt{m}} \underbrace{\mathcal{G}_{m-1,n}\psi_n}_{R^0_{m-1}} - \Sigma_{11} \sqrt{\frac{m-1}{m}} \underbrace{\mathcal{G}_{m-2,n}\psi_n}_{R^0_{m-2}} - \frac{\Sigma_{12}}{\sqrt{m}} \underbrace{\mathcal{G}_{m-1,n} a \psi_n}_{R^1_{m-1}}. \tag{8.6.9}$$

This relation must hold for any $k \geq 0$:

$$R^k_m = \frac{\mu_1}{\sqrt{m}} R^k_{m-1} - \Sigma_{11} \sqrt{\frac{m-1}{m}} R^k_{m-2} - \frac{\Sigma_{12}}{\sqrt{m}} R^{k+1}_{m-1}, \tag{8.6.10}$$

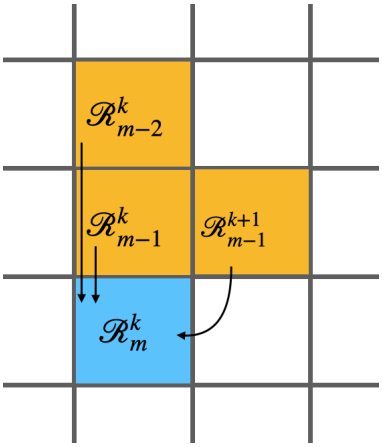


Figure 8.13: Any element in the R matrix (e.g. the blue one at the bottom) can be calculated as a linear combination of the three neighboring matrix elements shown in orange.

which means that we can write all the final amplitudes R^0_m as linear combinations of elements in R^k_0.

Imagine now R^k_m as a matrix whose rows are indexed by m and whose columns are indexed by k. From Eq. (8.6.10) we know that each element in R only depends on its three neighboring elements (see Fig. 8.13).

As we are now interested in the transformed state, which is the first column of R^0_m, we start by filling the first row R^k_0 and then apply the rule iteratively, stopping at one fewer element per row, effectively filling half of the matrix.

In our Python code, we proceed as follows:

1. We build a vector of square roots of integers up to N to avoid having to recompute them often.
2. We build the first row of the transformation matrix, $\mathcal{G}_{0,n}$.
3. We build the first row R^k_0 by computing $\mathcal{G}_{0,n}a^k\psi_n$ for $k \leq N$. To do so we take inner products only between the relevant part of the vector $\mathcal{G}_{0,n}$ and we update the state as $a^k\psi_n$ for the next value of k.
4. Starting from $m = 1$, we compute each row R^k_m using Eq. (8.6.10), but we only need $N - m$ elements.
5. After we have built the last row R^0_N (consisting of a single element), we read out the final result from the first column R^0_m.

The code can then be sped-up by using the Numba [154] decorator `numba.njit`, and it will be compiled at the first call.

Special case: large squeezing If the squeezing parameter is large, we don't need to generate the whole R matrix to find the transformed state.

Notice that in Eq. (6.2.53), we can approximate $\text{sech}(r) \sim 0$ and $\tanh(r) \sim 1$ as $\mathbf{r} \to \infty$. We keep $\sqrt{\text{sech}(r)}$ in C, as it is quadratically larger than $\text{sech}\, r$.

For a single-mode Gaussian transformation, we have:

$$C \sim \sqrt{\text{sech}\, r}\exp\left(-\frac{1}{2}|\gamma|^2 - \frac{1}{2}\gamma^{*2}e^{i(\delta+2\phi)}\right), \tag{8.6.11}$$

$$\boldsymbol{\mu}^T \sim [\gamma^* e^{i(\delta+2\phi)} + \gamma, 0], \tag{8.6.12}$$

$$\boldsymbol{\Sigma} \sim \left[\begin{array}{c|c} e^{i(\delta+2\phi)} & 0 \\ \hline 0 & -e^{-i\delta} \end{array}\right], \tag{8.6.13}$$

We can hence rewrite the recurrence relations of Eqs. (8.6.7) and (8.6.10) for a single-mode Gaussian transformation with large squeezing:

$$ß\mathcal{G}_{0,n} \sim e^{-i\delta}\sqrt{\frac{n-1}{n}}\mathcal{G}_{0,n-2}, \tag{8.6.14}$$

$$R^0_m \sim \frac{\gamma^* e^{i(\delta+2\phi)}+\gamma}{\sqrt{m}}R^0_{m-1} - e^{i(\delta+2\phi)}\sqrt{\frac{m-1}{m}}R^0_{m-2}. \tag{8.6.15}$$

Notice that in the second relation, R^0_m does not depend on R^k_m for $k > 0$. We then use the first relation to compute R^0_0 and the second to compute the output state directly. We remark that under these conditions, one can (differentiably) approximate the transformed state up to a cutoff in the order of seconds.

To assess the quality of the approximation within the cutoff, in Fig. 8.14, we show the deviation from the normalized overlap with respect to the exact state as a function of the parameter r of the squeezing for a few random states. The error in the normalized overlap between the approximated state and the exact state (here, a set of 10 random states) goes to zero as the magnitude of the squeezing parameter $r = |z|$ increases. Note that the overlap only includes photon number states up to $n = 50$. Also, note that such large values of r are unreachable in practice.

Note that we need the renormalization because we compute the exact output state only up to a fixed cutoff dimension, but large squeezing also populates Fock states with very large photon numbers.

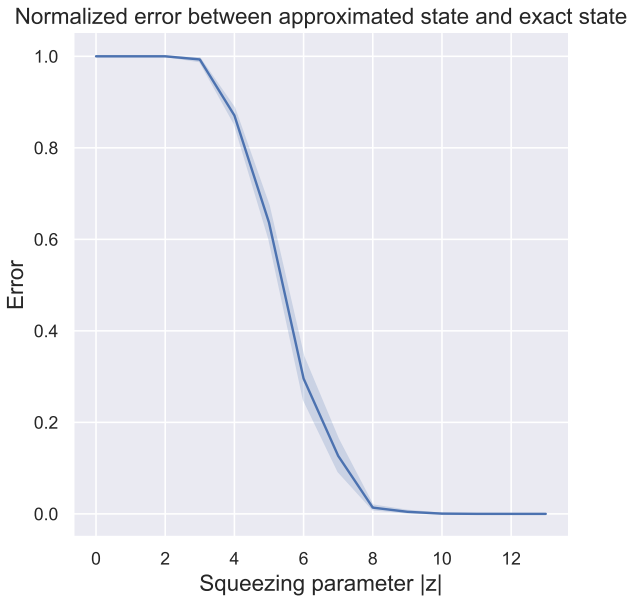


Figure 8.14: Normalized error between approximated and exact states for large squeezing parameter.

Two-mode transformed state

The transformation operator of a two-mode Gaussian transformation is a 4-dimensional tensor $\mathcal{G}_{m,n,p,q}$, so we need to compute its elements by using our recursive equations four times. We start from the input index, which is the last two indices p, q, and continue to calculate the R tensor combining the output index m, n with the input pure state $\psi_{m,n}$.

The first two indices are filled by applying the recurrence relation in Eq. (6.2.11):

$$\mathcal{G}_{0,0,0,q} = \frac{1}{\sqrt{q}}\left(\mathcal{G}_{0,0,0,q-1}\mu_4 - \sqrt{q-1}\mathcal{G}_{0,0,0,q-2}\Sigma_{44}\right), \tag{8.6.16}$$

$$\mathcal{G}_{0,0,p,q} = \frac{1}{\sqrt{p}}(\mathcal{G}_{0,0,p-1,q}\mu_3 - \sqrt{p-1}\mathcal{G}_{0,0,p-2,q}\Sigma_{33} - \sqrt{q}\mathcal{G}_{0,0,p-1,q-1}\Sigma_{34}). \tag{8.6.17}$$

In this case, we now have two different annihilation operators a_1, a_2 on each mode:

$$\mathcal{G}_{m,n,p-1,q}\sqrt{p} = \mathcal{G}_{m,n,p,q}a_1, \tag{8.6.18}$$

$$\mathcal{G}_{m,n,p,q-1}\sqrt{q} = \mathcal{G}_{m,n,p,q}a_2. \tag{8.6.19}$$

We then apply the recurrence relation again to fill the third and fourth indices m, n:

$$\mathcal{G}_{0,n,p,q} = \frac{1}{\sqrt{n}}(\mu_2\mathcal{G}_{0,n-1,p,q} - \Sigma_{22}\sqrt{n-1}\mathcal{G}_{0,n-2,p,q} - \Sigma_{23}\mathcal{G}_{0,n-1,p,q}a_1 - \Sigma_{24}\mathcal{G}_{0,n-1,p,q}a_2), \tag{8.6.20}$$

$$\begin{aligned}\mathcal{G}_{m,n,p,q} = \frac{1}{\sqrt{m}}(&\mu_1\mathcal{G}_{m-1,n,p,q} - \Sigma_{11}\sqrt{m-1}\mathcal{G}_{m-2,n,p,q} - \Sigma_{12}\sqrt{n}\mathcal{G}_{m-1,n-1,p,q} - \Sigma_{13}\mathcal{G}_{m-1,n,p,q}a_1 \\ &- \Sigma_{14}\mathcal{G}_{m-1,n,p,q}a_2).\end{aligned} \tag{8.6.21}$$

To compute the transformed output state $\mathcal{G}_{m,n,p,q}\psi_{p,q}$, we need to rewrite the recurrence relation in Eqs. (8.6.20) and (8.6.21) with R matrix. We get:

$$\begin{aligned}\underbrace{\mathcal{G}_{0,n,p,q}\psi_{p,q}}_{R^{0,0}_{0,n}} = \frac{1}{\sqrt{n}}(&\mu_2\underbrace{\mathcal{G}_{0,n-1,p,q}\psi_{p,q}}_{R^{0,0}_{0,n-1}} - \Sigma_{22}\sqrt{n-1}\underbrace{\mathcal{G}_{0,n-2,p,q}\psi_{p,q}}_{R^{0,0}_{0,n-1}} \\ &- \Sigma_{23}\underbrace{\mathcal{G}_{0,n-1,p,q}a_1\psi_{p,q}}_{R^{1,0}_{0,n-1}} - \Sigma_{24}\underbrace{\mathcal{G}_{0,n-1,p,q}a_1\psi_{p,q}}_{R^{0,1}_{0,n-1}}),\end{aligned} \tag{8.6.22}$$

$$\begin{aligned}\underbrace{\mathcal{G}_{m,n,p,q}\psi_{p,q}}_{R^{0,0}_{m,n}} = \frac{1}{\sqrt{m}}(&\mu_1\underbrace{\mathcal{G}_{m-1,n,p,q}\psi_{p,q}}_{R^{0,0}_{m-1,n}} - \Sigma_{11}\sqrt{m-1}\underbrace{\mathcal{G}_{m-2,n,p,q}\psi_{p,q}}_{R^{0,0}_{m-2,n}} \\ &- \Sigma_{12}\sqrt{n}\underbrace{\mathcal{G}_{m-1,n-1,p,q}\psi_{p,q}}_{R^{0,0}_{m-1,n-1}} - \Sigma_{13}\underbrace{\mathcal{G}_{m-1,n,p,q}a_1\psi_{p,q}}_{R^{1,0}_{m-1,n}} - \Sigma_{14}\underbrace{\mathcal{G}_{m-1,n,p,q}a_2\psi_{p,q}}_{R^{0,1}_{m-1,n}}).\end{aligned} \tag{8.6.23}$$

The relation holds for any j and k, so we finally obtain the recurrence relation for R:

$$R^{j,k}_{0,n} = \frac{1}{\sqrt{n}}(\mu_2 R^{j,k}_{0,n-1} - \Sigma_{22}\sqrt{n-1}R^{j,k}_{0,n-2} - \Sigma_{23}R^{j+1,k}_{0,n-1} - \Sigma_{24}R^{j,k+1}_{0,n-1}), \tag{8.6.24}$$

$$R^{j,k}_{m,n} = \frac{1}{\sqrt{m}}(\mu_1 R^{j,k}_{m-1,n} - \Sigma_{11}\sqrt{m-1}R^{j,k}_{m-2,n} - \Sigma_{12}\sqrt{n}R^{j,k}_{m-1,n-1} - \Sigma_{13}R^{j+1,k}_{m-1,n} - \Sigma_{14}R^{j,k+1}_{m-1,n}). \tag{8.6.25}$$

Similar to the single-mode case, because of the dependencies between neighboring elements, we compute the output state $R^{0,0}_{m,n}$ and the number of elements that need to be computed has an order of $\mathcal{O}(G^2)$ rather than the whole tensor $\mathcal{O}(G^4)$.

In our python code for the two-mode case, we proceed as follows :

1. We build a vector of square roots of integers up to N to avoid having to recompute them often.
2. We compute the tensor $\mathcal{G}_{0,0,p,q}$ using Eq. (8.6.16) and Eq. (8.6.17).
3. We compute $R^{j,k}_{00}$ by multiplying $\mathcal{G}_{0,0,p,q}a_1^j a_2^k$ and $\psi_{p,q}$. This step is analogous to the single-mode case, except that we loop over two indices instead of one.

4. We compute index by index from $R^{j,k}_{0,n}$ using Eq. (8.6.24) to $R^{j,k}_{m,n}$ using Eq. (8.6.25). We do the first equation under the conditions: $n < N,\ j < N - n,\ k < N - n - j$, and the second equation under the conditions: $m < N,\ n < N,\ j < N - m,\ k < N - m - j$, which allows us to compute only one-quarter of the possible N^4 elements of $R^{j,k}_{m,n}$.

5. We read out the final result from $R^{0,0}_{m,n}$.

8.6.2 Gradient calculation of new recurrence relation

The recurrence relations that we have presented are all linear, and therefore they can be easily differentiated. This gives us a direct way of computing the gradient of the transformed state with respect to the parameters of the transformation.

To compute the desired gradients (here for a single-mode), we differentiate Eqs. (8.6.8) and (8.6.10) with respect to a parameter ξ (or its conjugate ξ^*):

$$\partial_\xi \mathcal{G}_{0,n} = \frac{1}{\sqrt{n}}\left[\mu_2(\partial_\xi \mathcal{G}_{0,n-1}) + (\partial_\xi \mu_2)\mathcal{G}_{0,n-1} - \sqrt{\frac{n-1}{n}}(\partial_\xi \Sigma_{22})\mathcal{G}_{0,n-2} + \Sigma_{22}(\partial_\xi \mathcal{G}_{0,n-2})\right], \tag{8.6.26}$$

$$\begin{aligned}\partial_\xi R^k_m =& \frac{1}{\sqrt{m}}\left[(\partial_\xi \mu_1)R^k_{m-1} + \mu_1(\partial_\xi R^k_{m-1}) - \sqrt{m-1}(\partial_\xi \Sigma_{11})R^k_{m-2} + \Sigma_{11}(\partial_\xi R^k_{m-2})\right] \\ & - \frac{1}{\sqrt{m}}\left[(\partial_\xi \Sigma_{12})R^{k+1}_{m-1} + \Sigma_{12}(\partial_\xi R^{k+1}_{m-1})\right].\end{aligned} \tag{8.6.27}$$

And we obtain the gradient from the first column:

$$\frac{\partial \psi_m}{\partial \xi} = \partial_\xi R^0_m. \tag{8.6.28}$$

These recurrence relations can be computed once the R matrix is known. Therefore, after generating the R matrix, we keep it in memory for the computation of the gradient.

From a practical point of view, the backpropagated gradients have the same shape as the state tensor rather than the shape of the transformation tensor, which is a quadratic advantage over previous approaches. We still have to compute the transformation itself to backpropagate the gradients, but instead of the transformation gradient, we can just compute the gradient of R.

8.6.3 Comparison with state of the art

This section presents benchmarks of comparison between experiments based on our new method and state of the art.

We benchmark the runtime to compute the transformed state (i.e., the forward pass) against the current state of the art. We obtain the transformed state by three methods: constructing the Gaussian transformation matrix by using a sequence of gates in StrawberryFields [63] and taking the matrix-vector product by getting the Gaussian transformation matrix from previous work in the Ggate branch of The Walrus [155] and taking the matrix-vector product, and with the method in this work to get the transformed state directly.

The results are summarized in Fig. 8.15 and Fig. 8.16:

1. Fig. 8.15 is realized in 2019, state of the art was StrawberryFields (version 0.17.0), the Ggate we use is only a branch Ggate in TheWalrus (version 0.14.0).

2. Fig. 8.16 is realized recently in 2022, state of the art is StrawberryFields (version 0.22.0-dev) and TheWalrus (version 0.18.0).

In both figures, our implementation (lowest line, green line) is up to an order of magnitude faster at generating the output of a Gaussian transformation than the other methods. For the two-mode case, Strawberry Fields has the advantage of exploiting the photon-number conservation of the beamsplitter to spare

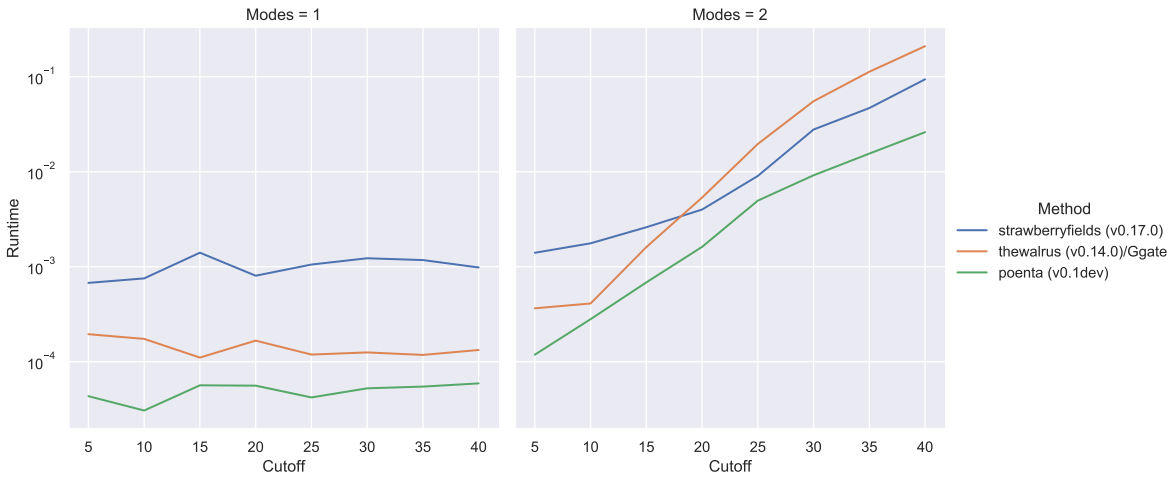


Figure 8.15: Comparison of the runtime to generate the transformed Gaussian states with SOTA in 2019.

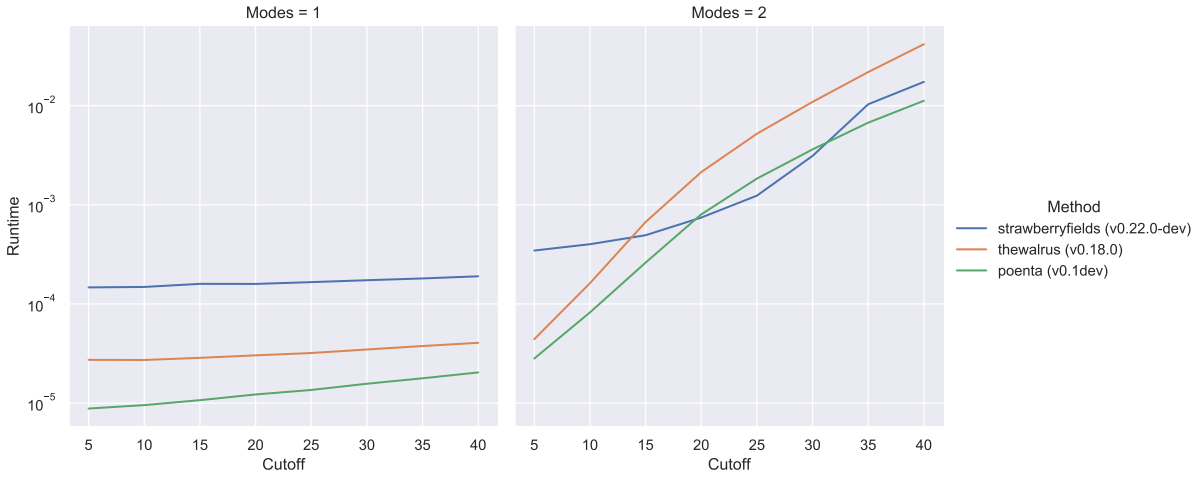


Figure 8.16: Comparison of the runtime to generate the transformed Gaussian states with SOTA.

one nested loop. In the other two implementations, we build the matrix all at once; therefore, we don't have access to the same savings. Despite this, our implementation is still faster and even more so when the backward pass is computed, as the backpropagated gradients have the shape of the state rather than the shape of the transformation.

8.7 Conclusion

The recurrence relation can be considered as a math tool that can solve lots of different problems. One can largely simplify the calculation for some special cases with specific constraints. This chapter gives some significant algorithms.

1. The first global and local cutoffs algorithm gives a new view to evaluate the linear combinations of the recurrence relation. Unlike the vanilla version, where one can only compute one element at a time, the algorithm combines the elements with the same weight and can calculate them all at a time.

2. The diagonal algorithm is proposed specifically for calculating the Fock amplitudes of mixed states. The complexity improves from $\mathcal{O}(C^{2M})$ to $\mathcal{O}(C^{M})$.

3. As for the interferometer, the first algorithm is proposed in case that we have at most one photon on each mode, and the total photon number n is less than the number of modes M, we improve the complexity quadratically from $\mathcal{O}(2^{2n})$ to $\mathcal{O}(2^{n})$ and the complexity is independent of the number of modes M. The second and third algorithms are proposed based on the tree strategy to cope with the multi-inputs and multi-outputs. Their performances are changeable depending on how similar the inputs and the outputs are.

4. We also propose a faster evolution algorithm under Gaussian operations, which not only gives the transformed output directly, avoiding the matrix multiplication errors but also is written as a recurrence equation preserving the differentiability.

The next chapter will give some examples and applications for optimizing photonic quantum circuits.

Chapter 9

Applications

9.1 Introduction

This chapter introduces applications based on optimizing photonic quantum circuits to show the excellent performance obtained with Poenta [60] and MrMustard [61] libraries that I have contributed to.

As a preliminary consideration, let us compare two libraries. The comparison between the two libraries is shown in Tab. 9.1. They are all based on the TF backend to take advantage of AD. The recurrence relation in chapter 6 is the core for both of them to build the differentiable photonic quantum circuits. If one uses the measurement in MrMustard to introduce the non-Gaussian effects, then the output state depends on the value of the measurement, and it is a conditional state preparation. Two optimization methods for Gaussian operators are explained in section 7.3, where Poenta uses the hardware-friendly one while MrMustard has them both. Moreover, Poenta implements the complex natural gradient we developed in section 7.4 and the fast evolution algorithm of Gaussian operators in section 8.6 as well. In addition, Poenta can build the circuit up to 2 modes for now because with the increasing number of modes, the hardware-friendly optimization would need to write down the custom gradient with respect to each parameter of the circuit, which would become very fastidious. This problem is solved in MrMustard because we use theory-friendly optimization instead and gradient update focuses only on the symplectic matrix and the mean vector of Gaussian operators. That is why we say we can process infinite modes, and in practice, we will optimize a 216-mode circuit as an example.

	Poenta	MrMustard
Backend	TF	TF
Elements	Gaussian + non-Gaussian	Gaussian + measurement
Gaussian Opt.	Euclidean	Euclidean + Riemannian
Technical part	Recurrence relation + CNG + Fast evolution Algo.	Recurrence relation
Scaling	Up to 2-modes	∞-mode

Table 9.1: Comparaisong between library Poenta and MrMustard

As shown in our optimization structures, we can build photonic PQCs and then optimize them with a given suitable cost function L. This chapter also shows examples of cost functions L corresponding to specific applications.

The first example is the state preparation task, which is widely considered as an important application, such as the preparation of Gottesman-Kitaev-Preskill (GKP) states, which paves the way to fault-tolerant quantum computing [10]. The suitable cost function is the quantum state fidelity, representing how close the

output and desired state are. We also show the state preparation task using the complex natural gradient algorithm we developed in section 7.4. The libraries Poenta, and MrMustard are used.

We can broaden the optimization task from the global optimization of the full circuit (first task) to the local optimization of one component (second task). The second task is optimizing the matrix $\boldsymbol{A}$ (defined in chapter 6) so that it has some properties to speed up the classical simulation of Gaussian Boson Sampling (GBS). The suitable cost function is related to the properties we expect with respect to $\boldsymbol{A}$. MrMustard [61] is the library we use in this part.

After training the circuit and obtaining the corresponding parameters' values in the circuit, implementing them in practice is also a big challenge. There are several essential points: in the layered structure, how to deal with the non-Gaussian operators (Kerr gate) we are widely using in the simulation; how to manipulate the imperfections of the components in practice and introduce them into the simulation; and other effects, such as decoherence.

This chapter is structured as follows: firstly, we produce GKP, single-photon, NOON, and cat states by using different libraries, such as Poenta [60] and MrMustard [61]. Then we show an example to optimize the matrix $\boldsymbol{A}$ with a 216-mode circuit of GBS. Last but not least, we discuss challenges and opportunities from our theoretical optimization to the experimental aspect and propose the lossy model of components of the circuits.

9.2 Quantum state preparation task

Quantum state preparation uses the PQCs and optimizes them to prepare some given states. In this case, the cost function is one minus the quantum state fidelity. The circuit's output, hence, becomes closer to the given state with the decrease of the cost function.

Depending on which library we use, there are two types of preparations:

- deterministic preparation, where non-Gaussian effects come from the non-Gaussian operators.
- conditional non-Gaussian state preparation. a conditioning measurement in a non-Gaussian basis, often the Fock basis, collapsing the wave function over the remaining mode(s), the signal, into a non-Gaussian state [134].

In this section, we first define the cost function for the quantum state preparation task. Then we define the states to prepare: the single-photon state, the cat state, the NOON state, the GKP1 state, and the GKP0 state. Last but not least, we show the ability to prepare these states with high fidelity in both libraries (Poenta and MrMustard) and discuss the runtime issue compared with state-of-the-art (SOTA).

9.2.1 The cost function for quantum state preparation task

Cost function in state preparation

To prepare a single pure state, it is simply to use one minus the fidelity (defined in Eq. (2.13)) of the output state to the target state as the cost function:

$$L(\boldsymbol{\theta}) = 1 - |\langle \Psi^{\text{target}} | U(\boldsymbol{\theta}) | \Psi^{\text{in}} \rangle|^2, \tag{9.2.1}$$

where $U(\boldsymbol{\theta})$ is our parametrized circuit and its parameters $\boldsymbol{\theta}$, $U(\boldsymbol{\theta})|\Psi_s^{\text{in}}\rangle$ denotes the output state of the circuit.

To prepare the superposition of a set of pure states, one would need to sum up the fidelity (defined in Eq. (2.13)) across the state set and get the average of them. With a set of S required input and target pairs $\{(\Psi_s^{\text{in}}, \Psi_s^{\text{target}})\}$, the cost function would be:

$$L(\boldsymbol{\theta}) = 1 - \frac{1}{S} \sum_{s=1}^{S} |\langle \Psi_s^{\text{target}} | U(\boldsymbol{\theta}) | \Psi_s^{\text{in}} \rangle|^2, \tag{9.2.2}$$

where $U(\boldsymbol{\theta})$ is our parametrized circuit and its parameters $\boldsymbol{\theta}$, $U(\boldsymbol{\theta})|\Psi_s^{\text{in}}\rangle$ denotes the s-th output state of the circuit. Therefore, the inner product here is the average fidelity of the output states to their respective target states.

9.2.2 States

GKP state

Gottesman-Kitaev-Preskill (GKP) states are a way to encode a qubit in a CV system (such as an optical oscillator) in order to achieve error correction capabilities [156]. GKP state is one of the most famous and important bosonic codes in quantum communication [157] because of its error resistance. However, the ideal GKP states have infinite energy and cannot be realized in practice.

We consider finite-energy versions of these states obtained by applying the energy damping operator $E(\epsilon) = e^{-\epsilon a^\dagger a}$ (with $\epsilon = 0.2$) to the ideal GKP states [156, 140]:

$$E(\epsilon)|\psi\rangle_{\text{gkp}} = \cos\frac{\theta}{2}E(\epsilon)|0\rangle_{\text{gkp}} + e^{-i\phi}\sin\frac{\theta}{2}E(\epsilon)|1\rangle_{\text{gkp}}, \tag{9.2.3}$$

where

$$|\mu\rangle_{\text{gkp}} = \sum_n |(2n+\mu)\sqrt{\pi\hbar}\rangle_q, \tag{9.2.4}$$

and $|x\rangle_q$ is an eigenstate of the quadrature $q = \sqrt{\frac{\hbar}{2}}(a + a^\dagger)$ with eigenvalue x.

In our experiment, we generate two ideal GKP states from the library StrawberryFields as the target state. The GKP state in Fig. 9.2a is defined as GKP1, which has $\epsilon = 0.1$, $\theta = \pi$, and $\phi = 0$. The GKP state in Fig. 9.5a is defined as GKP0, which has $\epsilon = 0.2$, $\theta = 0$, and $\phi = 0$.

single-photon state

Single-photon sources play an essential role in quantum key distribution [158] and quantum computing.

The single-photon state (shown in Fig.9.1a) in the Fock representation is written as $|1\rangle$.

NOON state

NOON states are important as they are known to be an optimal resource for quantum metrology and quantum sensing for their ability to allow precise phase measurements, which saturates the Heisenberg limit for the error. They were first introduced in the paper [159] and got named in paper [160].

A 2-mode NOON state is defined as:

$$|\Psi_{\text{NOON}}\rangle = \frac{|N,0\rangle + |0,N\rangle}{\sqrt{2}}, \tag{9.2.5}$$

where the first term denotes N particles in the first mode and vacuum in the second mode and vice versa in the second term.

Cat state

Cat states are superpositions of coherent states, which we write as

$$|\text{cat}_\pm\rangle = \mathcal{N}\left(|\alpha\rangle \pm |-\alpha\rangle\right), \tag{9.2.6}$$

where $\mathcal{N}$ is a normalization constant and $|\alpha\rangle = \mathcal{D}(\alpha)|0\rangle$ is a coherent state. In the last equation, the plus sign corresponds to even cat states and the minus corresponds to odd cat states. The normalization constant is never $1/\sqrt{2}$ as the two coherent states are never fully orthogonal, but they approach orthogonality exponentially fast in $|\alpha|$.

For this example, we will target the generation of an odd cat state with $\alpha = 2$. The ideal cat state is shown in Fig. 9.6a.

9.2.3 State preparation in both libraries

We produce the GKP, single-photon, and NOON states in Poenta, which matches the targets of Refs. [148, 35, 149] and allows for a comparison. And with MrMustard, we produce two candidate states for fault-tolerant quantum computing: GKP and Cat states.

	Poenta	MM
GKP state	×	×
single-photon	×	
NOON state	×	
Cat state		×

Table 9.2: State preparation list

Poenta

We benchmark a circuit optimization task for various one-and two-mode states.

Our architecture is made of a sequence of L layers, each formed by a Gaussian gate followed by a single-mode Kerr gate in each mode:

$$U(\boldsymbol{\beta}) = \prod_{\ell=1}^{L} \mathcal{G}^{\ell}(\boldsymbol{\alpha}_{\ell}) K(\boldsymbol{\kappa}_{\ell}), \tag{9.2.7}$$

where $\boldsymbol{\alpha}_{\ell}$ denotes the parameters of the ℓ-th Gaussian layer and $\boldsymbol{\beta}$ denotes the parameters of the whole circuit. One can also find the scheme in Fig. 5.2.

We need to point out that because our benchmark is based on the numerical simulation, we will not discuss if the value of κ here of the Kerr operator is realistic for the presently-available technologies or not. But in Appendix D, as an example in Poenta, we give optimized circuit parameters in the table to prepare a single-mode state. In addition, MrMustard adds the choice of bounds when defining operators, allowing us to have a more realistic optimization.

We ran benchmarks on generating a single-photon ($|\Psi^{\text{target}}\rangle = |1\rangle$), NOON state with $N = 5$ ($|\Psi^{\text{target}}\rangle = (|5,0\rangle + |0,5\rangle)/\sqrt{2}$) and GKP1 state (defined in section 9.2.2). And all initial states are the vacuum state $|\Psi^{\text{in}}\rangle = |0\rangle$. The optimized single-photon state and the comparison with the target single-photon state are shown in Fig.9.1. Also, the optimized GKP1 state and the comparison are shown in Fig.9.2. The fidelity for the single photon is 99.996%, while the GKP1 has 99.721%.

The details about the hyperparameters of training and the results are shown in Tab.9.3. One can compare them with the previous implementation in table I of [35] and the original implementation in table I of [148] while taking into account differences in computational hardware, and we achieved about twice the performance in the runtime with high fidelity.

MrMustard

We elaborate on how we can accomplish the state preparation task inside the library MrMustard (version 0.3.0) [61] with different kinds of optimization methods: such as euclidean, unitary, and symplectic. GKP0 state and cat state are generated as examples.

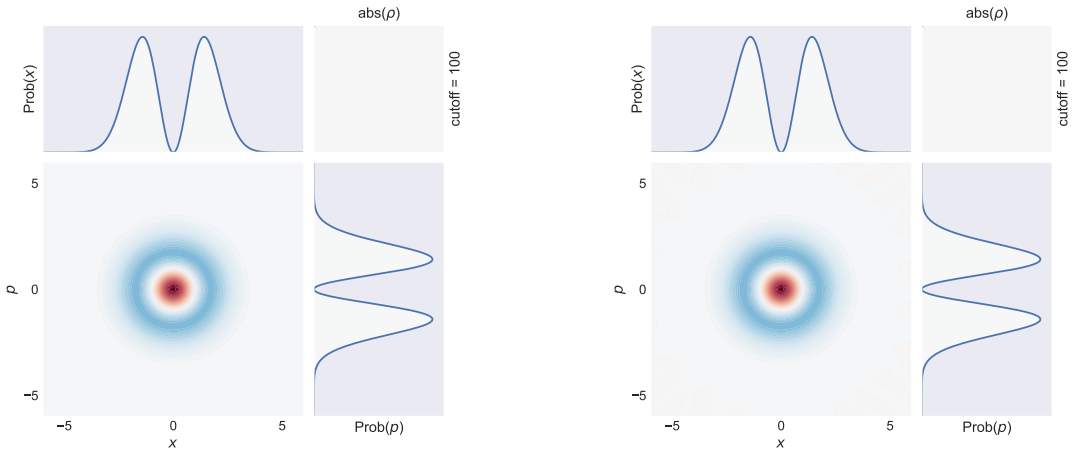
As for the circuit, we propose three kinds of simple circuits:

1. Gaussian gates with symplectic optimization;

2. Squeezing gate and multi-mode interferometers with unitary optimization;

		Single-photon			GKP1			NOON	
Work		[148]	[35]	This	[148]	[35]	This	[148]	This
Hyperparameter	Cutoff	6	100	100	50	100	50	10	10
	Layers	8	8	8	25	35	25	20	20
	Steps	1500	1500	1500	10000	5000	10000	5000	**3000**
Results	Fidelity (%)	99.998	99.998	99.996	99.83	99.60	99.721	99.89	**99.913**
	Runtime (s)	65	50	**15**	6668	720	**286**	1270	**145**

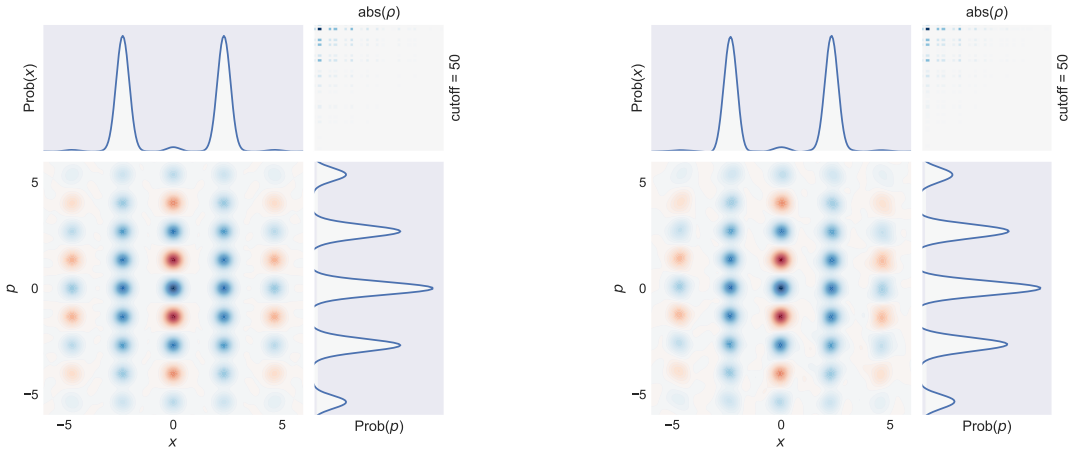
Table 9.3: Runtime for three circuit optimization tasks (single-core on an Apple M1 chip) compares with previous implementations.



(a) Target single-photon state.

(b) Optimized single-photon state with fidelity 99.996%.

Figure 9.1: The target single-photon state and the optimized one. (With Poenta)



(a) Target GKP1 state.

(b) Optimized GKP1 state with fidelity 99.721%.

Figure 9.2: The target GKP1 state and the optimized one. (With Poenta)

3. Squeezing gate and beam-splitter with Euclidean optimization.

We will show three optimization methods in the experiments, including the Euclidean method for the basic optical components (single-mode squeezing, rotation operator, beam-splitter, etc.), the Unitary method for the multi-mode interferometer, and the Symplectic method for Gaussian objects.

We generate the GKP0 state (defined in section 9.2.2)) from the library StrawberryFields.

We have three state preparation circuits for the GKP0 state (shown in Fig. 9.3), which use different optimizers: symplectic optimizer, orthogonal optimizer, and euclidean optimizer.

The first one consists of a three-mode Gaussian gate with the measurements on the first and second modes, and then the third mode is the output of the GKP0 state. The circuit is shown in Fig. 9.3a. Its optimization is based on the symplectic optimizer for the Gaussian gates, and the result is shown in Fig. 9.5b with a fidelity of 93.04%.

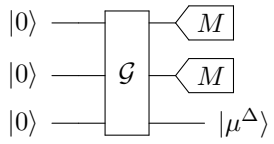
The second one consists of three single-mode Squeezing and a three-mode interferometer and the measurements on the first and second modes. We can get the GKP0 state on the third mode. The circuit is shown in Fig. 9.3b. Its optimization is based on the orthogonal optimizer for the multi-mode interferometer and the result is shown in Fig. 9.5c with a fidelity of 93.00%.

The last experiment consists of three single-mode Squeezing, a beamsplitter on the first and the second mode, and a beamsplitter on the second and the third mode. After measuring the first and second modes, we can get the GKP0 state on the third mode. The circuit is shown in Fig. 9.3c. Its optimization is based on the euclidean optimizer for all the basic optical gates (squeezing gate and beamsplitter), and the result is shown in Fig. 9.5d with a fidelity of 93.01%.

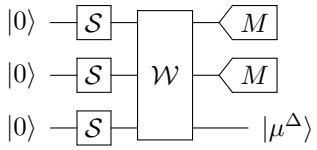
To obtain the cat state, we also tried two different optimizers in our library: the symplectic one and the euclidean one.

The first circuit (shown in Fig. 9.4a) consists of a Gaussian gate and photon-number measurement on the first mode and generates the (approximate) cat state in the second mode. We use the symplectic optimizer to train the Gaussian gate. The result is shown in Fig. 9.6b with a fidelity of 99.42% and 5.40% success probability.

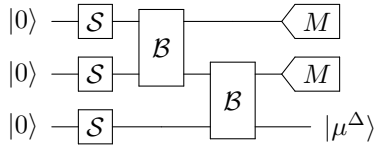
In the second circuit (shown in Fig. 9.4b), we use a simple two-mode circuit, starting with two single-mode squeezers followed by a beamsplitter and ending with a Fock measurement of 3 photons on the first mode. We train the circuit from random initial parameters for two squeezers and a beamsplitter. We obtain Fig. 9.6c as a result with a fidelity of 99.38% and 7.39% success probability.



(a) The trainable circuit consists of only Gaussian gates $\mathcal{G}$ and measurements M, using symplectic optimization.

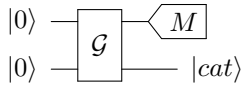


(b) The trainable circuit consists of squeezing gates $\mathcal{S}$ and multi-mode interferometer $\mathcal{W}$ and measurements M, using orthogonal optimization and euclidean optimization.

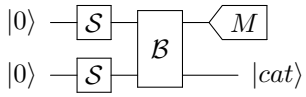


(c) The trainable circuit consists of squeezing gates $\mathcal{S}$ and beam-splitters $\mathcal{B}$ and measurements M, using only euclidean optimization.

Figure 9.3: GKP0 state preparation circuits. In the example optimization, we choose both $M = 6$.

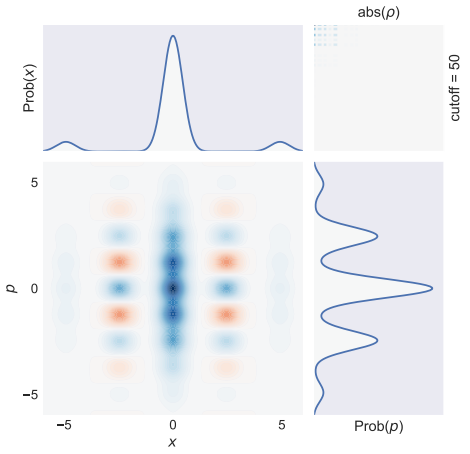


(a) The trainable circuit consists of a Gaussian gate $\mathcal{G}$ and a measurement M, using symplectic optimization.

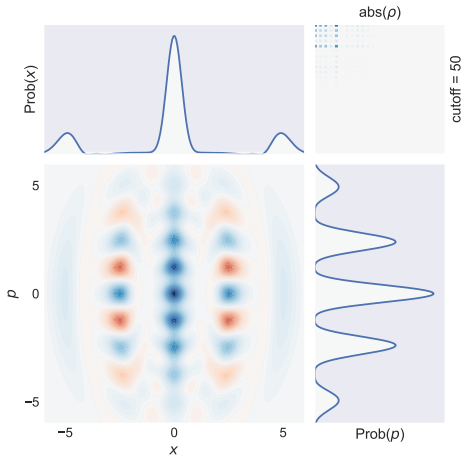


(b) The trainable circuit consists of squeezing gates $\mathcal{S}$ and a beam-splitter $\mathcal{B}$ and measurements M, using euclidean optimization.

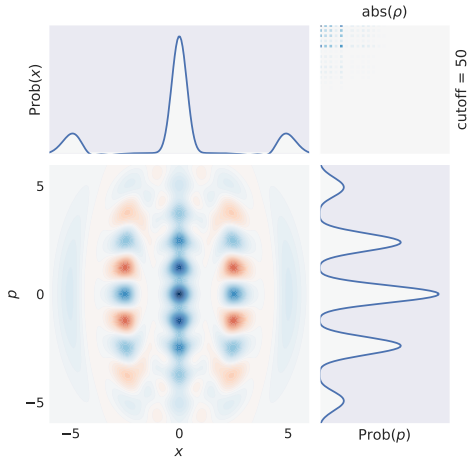
Figure 9.4: Cat state preparation circuits. In the example optimization, we choose $M = 3$.



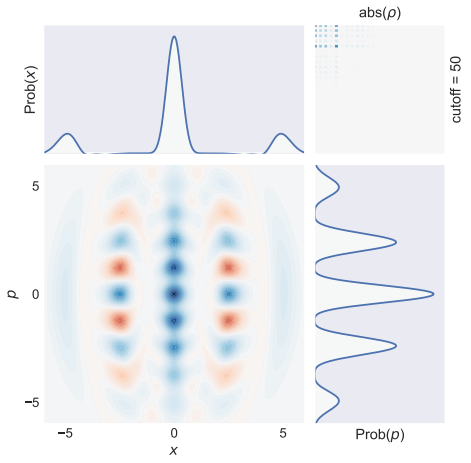
(a) Target state of the GKP0 state from the library StrawberryFields.



(b) The optimized output state with the circuit in Fig. 9.3a.

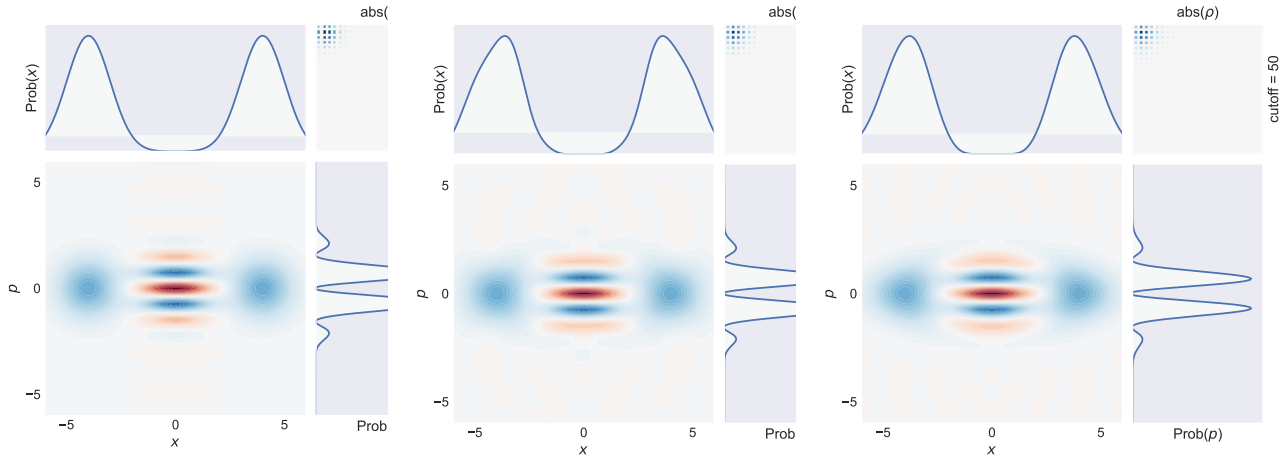


(c) The optimized output state with the circuit in Fig. 9.3b.



(d) The optimized output state with the circuit in Fig. 9.3c.

Figure 9.5: The target GKP0 state and the optimized state. (With MrMustard)



(a) Target cat state defined in Eq. (9.2.6) with $\alpha = 2$.

(b) Optimization of the cat state with the circuit in Fig. 9.4a.

(c) Optimization of the cat state with the circuit in Fig. 9.4b.

Figure 9.6: The target cat state and the optimized cat state. (With MrMustard)

These results are interesting in that we can generate cat and GKP states with such simple circuits and achieve high fidelity. Compared with the simulation of layered structure circuits in Poenta, MrMustard gives a great solution with much fewer elements and without the non-Gaussian gates. This inspires us to think about what is the minimum resources for state preparation tasks and how to find them (one of the perspectives in section 10.2). Since we use PNR detectors at the end of circuits in MrMustard to supply the non-Gaussian effects, the state preparation task is conditional, and the success probability is important in this case and depends on the measurements. As for the cat state example in Fig. 9.4b with a fidelity of 99.38%, the success probability is 7.39%. However, the success probability of the GKP state preparation is really low because the two measurements have a high number of 6 photons, such as the circuit in Fig. 9.3c with a fidelity of 93.00%, its success probability is 5.93e-06%. In the future, it is possible to take the measurement value as one of the optimization's hyperparameters to have a higher success probability.

9.2.4 Complex Natural Gradient in Poenta

We have implemented the complex natural gradient algorithm in our library Poenta.

There are two ways to compute the natural gradient: The first is with respect to all the parameters in the entire circuit at once; the second is with respect to the parameters of the layer being updated. In the first case, we compute the gradient of the cost function with respect to the entire collection of parameters of the circuit; then we compute the full Hermitian metric tensor, we invert it, and finally, we apply it to the full gradient vector before performing a single gradient-descent step. This can be very costly if the number of layers is large, as the effective number of parameters (i.e., counting twice as many for complex ones) in each N-mode layer is $n = 2N^2 + 4N$, and the metric tensor is $n \times n$. This gives an estimate of the cost of matrix inversion as $O(n^3) = O(N^6)$. In the second case, we compute the Hermitian metric in block-diagonal form, where we include only the parameters that belong to the same layer in each block. This technique was studied in [51], where it was shown that using the block-diagonal metric is, in practice, just as good as using the full metric while sparing a significant amount of computation. Finally, we note that before the (pseudo)inversion step, we add a constant regularization term, as we observed that the metric is often singular [42]:

$$\tilde{f}^{+} = (f + \lambda \mathbb{1})^{+}, \tag{9.2.8}$$

where we set an empirical value of $\lambda \approx 0.1$ after assessing the results of numerical experiments. As mentioned before, the singularities in the metric correspond to points in parameter space where at least one of the parameters cannot influence the quantum state [as an intuitive analogy, compare a qubit in the state $|0\rangle$ (on the north pole) with a qubit in the state $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ (on the equator): the $|+\rangle$ state is going to be very sensitive to a rotation around the z axis, but the $|0\rangle$ state is completely insensitive to it].

In the following experiments, we compare vanilla gradient descent (vanilla GD), Adam, and natural gradient descent (NGD) using the optimal learning rate for each one. In particular, Adam [89] is a ubiquitous gradient-descent optimization method that can dynamically select a step size and performs well in many deep-learning applications.

In order to compare the gradient-descent algorithms fairly, we cannot use the same learning rate, as it means different things for different algorithms. Moreover, in Adam, the learning rate evolves over time.

To determine the best learning rate for each algorithm, we search for it by trial and error as the one that allows the optimizer to reach the lowest cost function value quickly without incurring excessive oscillations if we let it run.

We run tests separately with the three different algorithms (vanilla GD, NGD, and Adam) to train the single-mode quantum circuit for generating a single-photon and GKP1 state.

For each algorithm, we use 20 random seeds to generate the initial parameters in the circuit and test for a range of learning rates from 0.0001 to 0.5. The curves are made by averaging the 20 different tests. We present in Figs. 9.7 and 9.8 a few typical curves around the *optimal* learning rate to show how we choose it for each of them.

In Fig. 9.7, we look for the optimal learning rate in the single-photon preparation task that allows the optimizer to reach the lowest cost function value quickly without incurring excessive oscillations if we let it run. The optimal learning rate to generate a single photon is 0.02 for vanilla GD, 0.02 for NGD, and 0.01 for Adam.

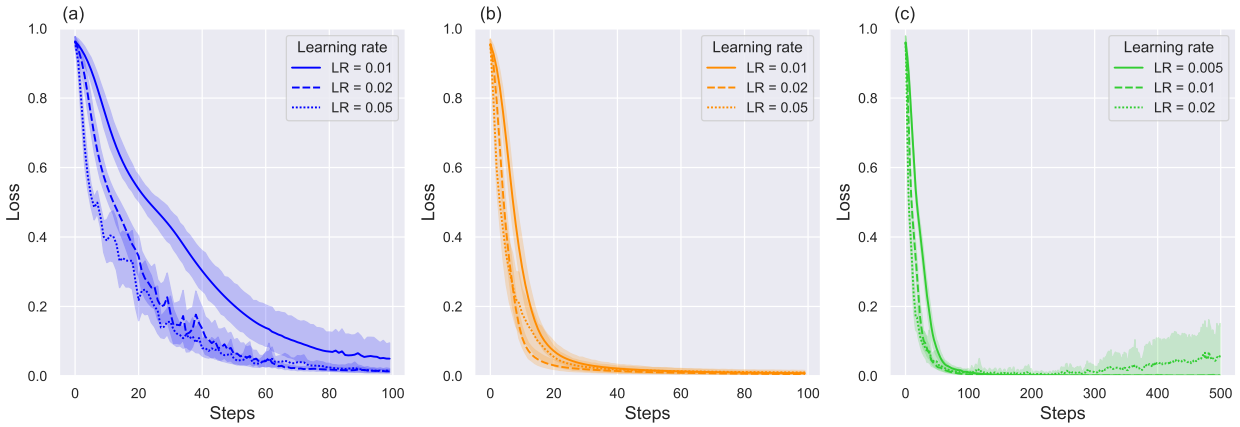


Figure 9.7: Find the optimal learning rate of single-photon preparation task. (a) Vanilla GD. (b) NGD. (c) Adam.

In Fig. 9.8, we look for the optimal learning rate in the GKP1 preparation task. The optimal learning rate to generate a single photon is 0.001 for vanilla GD, 0.02 for NGD, and 0.001 for Adam.

After choosing the best learning rate for each optimizer, we compare them in each state preparation task.

The first target that we choose is a single-photon state $|\psi_{\text{target}}\rangle = |1\rangle$ and the initial state is a vacuum state. We employ an eight-layer circuit with a total of $8 \times 6 = 48$ parameters and a Fock-space cutoff of 100.

We report the results of the optimization in Fig. 9.9. The NGD converges significantly faster than Adam and vanilla GD, all the while following a smooth decay of the cost function.

The second target is the GKP1 state, and the initial state is the vacuum state. We use a 25-layer circuit with a total of 150 parameters and a Fock-space cutoff of 50.

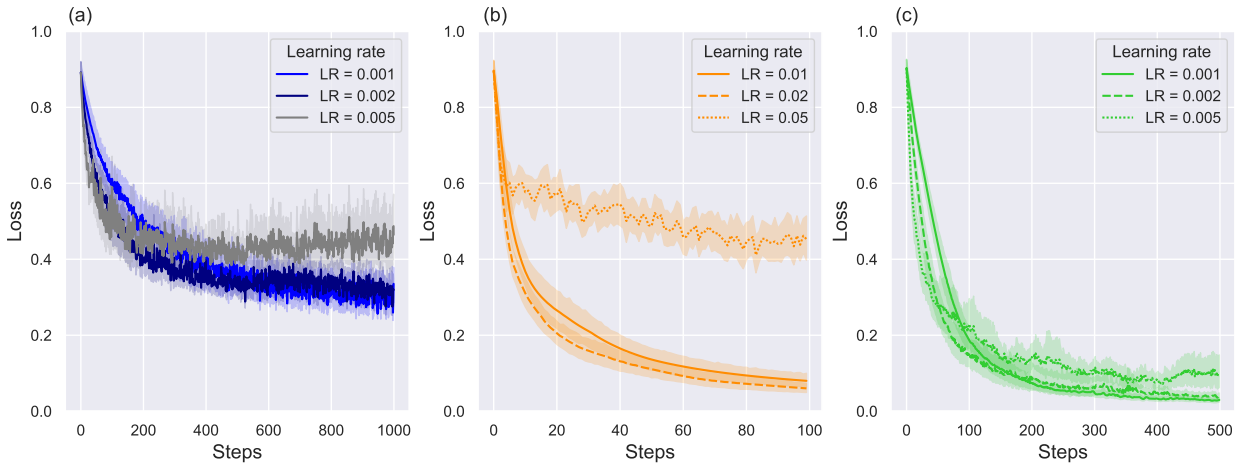


Figure 9.8: Find the optimal learning rate of GKP1 preparation task. (a) Vanilla GD. (b) NGD. (c) Adam.

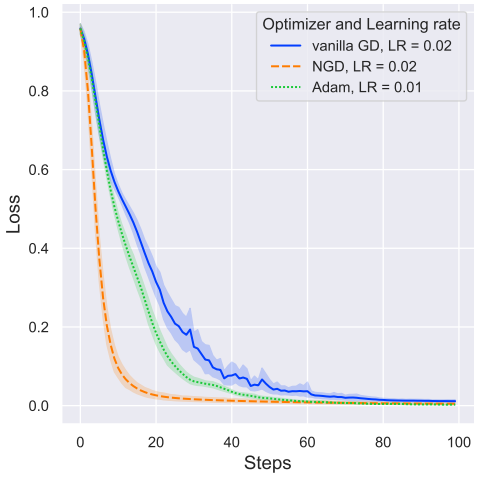


Figure 9.9: Optimization of the single photon generation circuit for vanilla GD, NGD, and Adam.

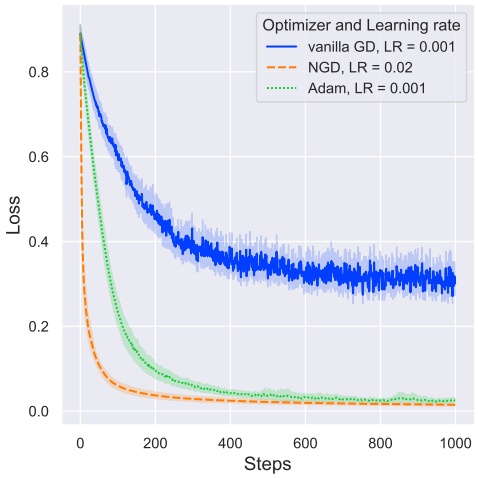


Figure 9.10: Optimization of the GKP1 state-generation generation circuit for vanilla GD, NGD, and Adam.

This is a much more challenging optimization task than the single-photon target, and the cost landscape is much more rugged, as can be seen from the behavior of the vanilla GD curve. Adam also seems to suffer from the rugged landscape, albeit much smaller. In contrast, the NGD curve is smoother even at a learning rate of 0.02, allowing it to converge quickly to a high-fidelity output.

9.3 Maximizing the entanglement in Gaussian Boson Sampling

We first analyze high-dimensional Gaussian Boson Sampling (GBS) instances similar to the 216-mode circuit of the photonic processor Borealis [7]. In a high $D-$dimensional GBS instance with $M = d^D$ modes, a set of $K \leq M$ squeezed modes are sent into an interferometer composed of layers of beamsplitter gates (with a local rotation gate in the first mode) between modes i and $i + \tau$ with $\tau \in \{1, d, d^2, \ldots, d^{D-1}\}$ as shown schematically for $d = 6$ and $D = 3$ in Fig. 9.11.

One desirable property of any GBS instance is that its adjacency matrix, which corresponds to $\boldsymbol{A}_\psi$ in our notation, should not have any special property like being banded, sparse, or low-rank. This is because these types of properties can be exploited to speed-up the classical simulation of GBS.

For high-dimensional GBS instances like the one implemented in Borealis, it is known that the $\boldsymbol{A}_\psi$ is full-rank (since every input is squeezed) and not banded (due to the long-ranged gates). However, one needs to judiciously choose the parameters of the beamsplitter so that the distribution of its entries is not heavily dominated by just a few of them. For example, if one chooses the rotation gates and the transmission angles of the beamsplitters to be uniformly random in $[-\frac{\pi}{2}, \frac{\pi}{2}]$, one obtains the distribution shown in blue bars in Fig. 9.12c and the $\boldsymbol{A}_\psi$ matrix show in Fig. 9.12a. For these results and following Ref. [7], we fix the phase angle of the beamsplitter to be $\pi/2$, we set the input squeezing parameter in all the modes to be $r = \operatorname{arcsinh} 1 \approx 0.8813736$ and take $D = 3$, $d = 6$ and thus a total of $M = 6^3 = 216$ modes. Note that the values of the matrix are heavily concentrated, i.e., for each row and column, a few values are overwhelmingly larger than the rest.

We can now use the methods we developed to try to spread out the entries of the matrix $\boldsymbol{A}_\psi$. We can then optimize the cost function

$$\min \sum_{ij} (|\boldsymbol{A}_\psi|^2_{ij} - \operatorname{mean}|\boldsymbol{A}_\psi|^2)^2. \tag{9.3.1}$$

We perform this optimization by obtaining the distribution shown with the orange bars in Fig. 9.12c and the matrix shown in Fig. 9.12b. Notice that now the values are more evenly distributed.

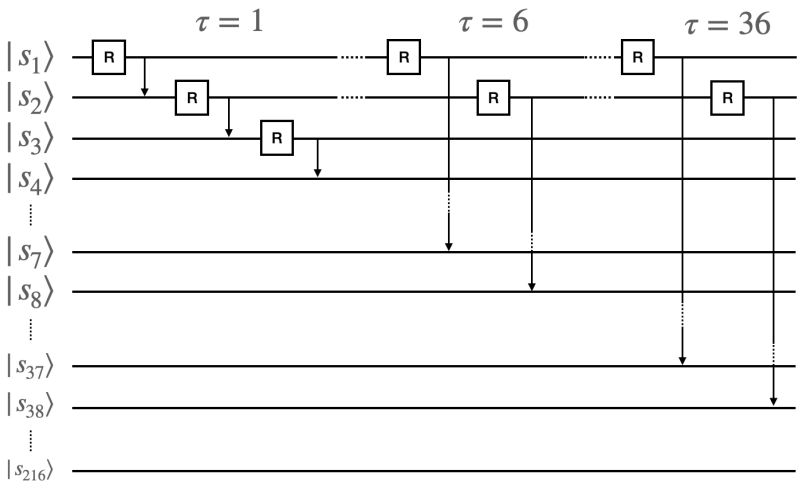


Figure 9.11: Schematic of the connectivity of the 216-mode circuit of the photonic processor Borealis, where all the nearest neighbor modes are connected by beam splitters, then all pairs at a distance 6 and finally all pairs at a distance 36. The arrows represent beam splitters.

9.4 From theoretical optimization to experimental aspect

This section discusses how to transfer the circuits we use in the optimization task into a real optical circuit. Introducing the numerical simulation with the optimization method can help us to prove the feasibility of various experiments before launching, arrange new quantum optical experiments, and even design new quantum devices.

Recall in chapter 5, we give two optimization schemes: one using Gaussian and non-Gaussian gates (Fig. 5.3), another one using Gaussian gates with measurements (Fig. 5.4). Both schemes are used to complete the state preparation task in section 9.2.

At the starting stage of my thesis, we developed the library Poenta to simulate the scheme with Gaussian and non-Gaussian gates. As shown in Tab. 9.3, we can produce really high-fidelity quantum states with a short runtime to run the simulation. However, the difficulty bringing those circuits into realistic things is the non-Gaussian gates.

The Kerr gate we use in this layered structure is defined in Eq. (3.63), whose parameter κ is the one that we try to find in the optimization. The parameter κ is proportional to the third-order nonlinear susceptibility $\chi^{(3)}$. We derive the explicit expression between the third-order nonlinear susceptibility $\chi^{(3)}$ and the Kerr parameter κ.

The energy of the Kerr operator can be expressed as

$$\mathcal{H} = P^{(3)}E = \epsilon_0 \chi^{(3)} |E|^4. \tag{9.4.1}$$

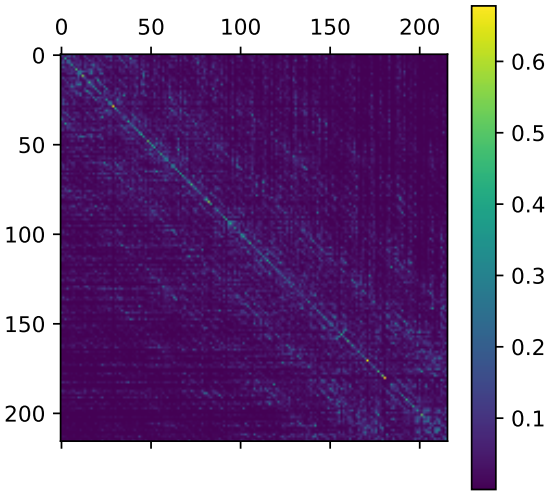
The energy density of N photons per unit volume is $N\hbar\omega$, and the energy density of the optical field is $W = 2\epsilon n|E|^2$, so we have

$$N\hbar\omega = 2\epsilon n|E|^2, \tag{9.4.2}$$

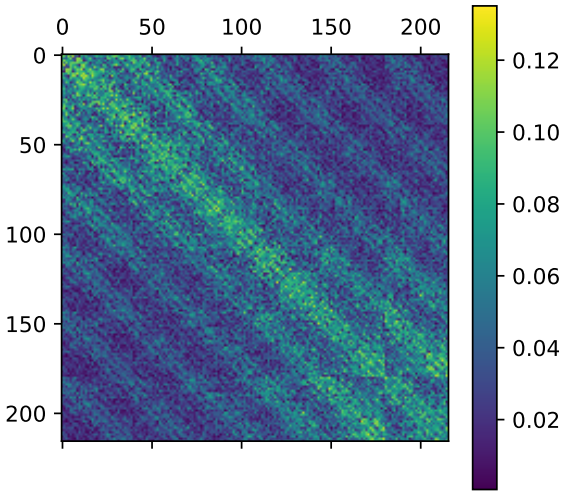
where ω is the frequency of the light, $\epsilon = \epsilon_r\epsilon_0 = \epsilon_r 8.854 \times 10^{-12} F/m = J.V^{-2}.m^{-1}$ is the permittivity of free space and ϵ_r is the relative permittivity between the Kerr medium and free space, n is the refractive index of the Kerr medium, and E is the amplitude of the optical field.

We then replace E in Eq. (9.4.1) by using Eq. (9.4.2):

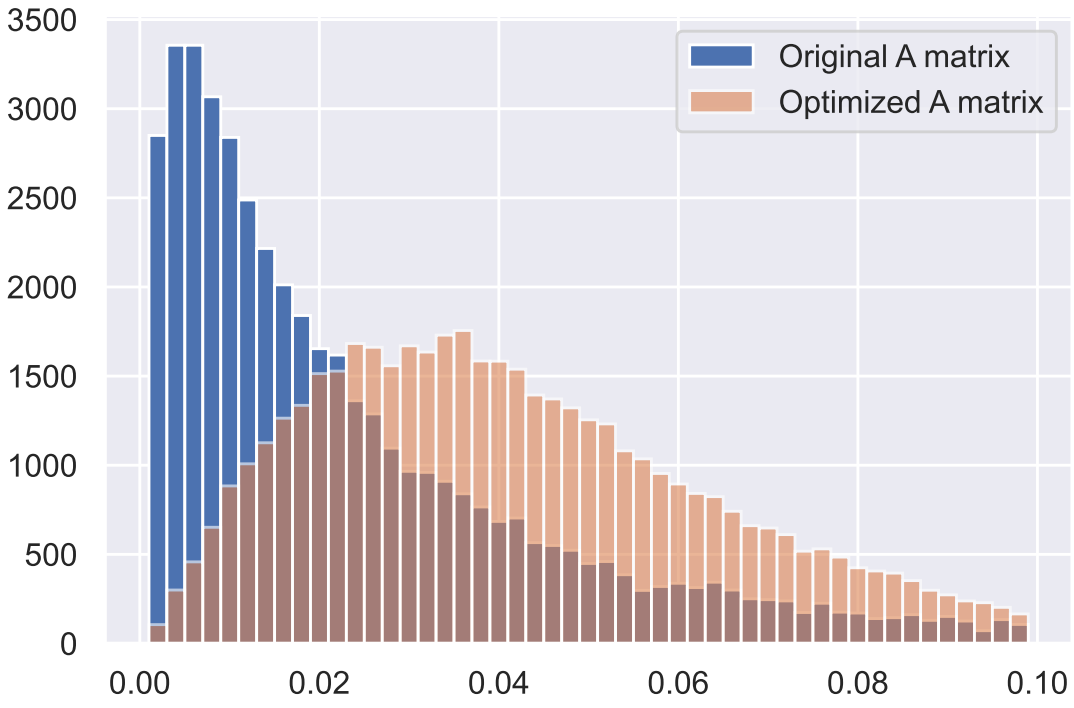
$$\mathcal{H} = \frac{\chi^{(3)}\hbar^2\omega^2}{4\epsilon n^2} N^2. \tag{9.4.3}$$



(a) The absolute value of the original matrix $\boldsymbol{A}$.



(b) The absolute value of the matrix $\boldsymbol{A}$ after optimization.



(c) Histogram of the absolute values of $\boldsymbol{A}$ matrix elements before and after optimization.

Figure 9.12: Maximizing the entanglement in Gaussian Boson sampling

Compared with the definition of Kerr Hamiltonian in Eq. (3.63), we can get the expression of κ:

$$\kappa = \frac{\chi^{(3)}\hbar^2\omega^2}{4\epsilon n^2}. \tag{9.4.4}$$

The magnitude order of $\chi^{(3)}$ is always small for us to simulate, for example $3.55 \times 10^{-12} m^2/V^2$ for Nitrobenzene [161]. If we take the visible light $\omega = 600THz$, we have

$$\kappa \approx \frac{3.55 \times 10^{-12} m^2/V^2 (6.63 \times 10^{-34} J.s)^2 (6 \times 10^{14} s^{-1})^2}{4\epsilon_r 8.854 \times 10^{-12} J.V^{-2}.m^{-1} n^2} \approx 10^{-40} J.m^3. \tag{9.4.5}$$

The magnitude order of κ in this example is 10^{-40}. There is one example in Appendix D.1, the optimized value of κ varies in $[-0.115, 0.310]$, which is too large compared with the magnitude of κ. Moreover, if we start our simulation from $\kappa = 0$, the learning rate is always 0.001; after one step of updates, the optimized value of κ is 0.001 or -0.001, which are all impossible to realize in practice.

Since it is impossible to realize the non-Gaussian gates with a parameter that we want in the laboratory, we changed the scheme to the second one with Gaussian gates and measurements and developed the library MrMustard. This section talks about how to transform the results of our simulations into realistic experiments of the optimization scheme with measurements.

To transform the numerical simulation results into realistic photonic circuits, our optimized circuits can be considered as the first step. There are still several effects that should be taken into account:

- imperfection of the components in the circuits;
- decoherence.

In the circuit, photon loss is the first imperfection. Photon loss can happen during light transmission and traversing different elements. We propose a lossy model to include the imperfections of components in our simulation scheme. We give examples of imperfect gates and detectors in section 9.4.1.

The decoherence problem comes from environmental noise, such as light or other electronic devices.

Moreover, the rise of integrated photonics chips [162, 163, 164] would make it possible to implement our parametrized circuits into a single small chip and thus minimize some of these problems.

9.4.1 Include the imperfections of components in the simulation

The idea would be to add some more gates in the optimization scheme to mimic the imperfections that could happen in practice.

Imperfections of gates

Fig. 9.13 gives an example of the rotation gate $\mathcal{R}(\phi)$.

One could optimize the full circuit with the ideal gate $\mathcal{R}(\phi)$ shown in Fig. 9.13 (a) and then replace the ideal gate with our lossy model shown in Fig. 9.13 (b), which uses beam-splitters $\mathcal{B}(\theta, \varphi)$ to introduce photon loss. We use an ancilla mode with the vacuum input, let the state passes a beam-splitter and throw out the ancilla mode, an ideal gate, and a second ancilla mode with the vacuum input with another beam-splitter, and then throw out the second ancilla mode. This lossy model can describe the photon loss between two elements and inside the operator together. The parameters θ, φ of the beam-splitter come from prior experiments in the laboratory. Rerun this lossy model, we can obtain a new parameter ϕ' for the rotation gate. By analyzing the nuances between ϕ and ϕ', one can recognize how the loss influences the parameters.

Imperfections of detectors

Photon number resolving (PNR) detectors are used in various applications, such as quantum key distribution [165], quantum imaging [166], and quantum computation [101]. In our optimization scheme, we use the PNR detector to do the herald state preparation.

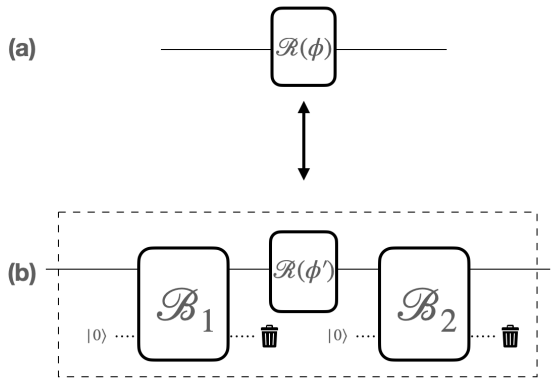


Figure 9.13: Simulate imperfections of gates. (a) Ideal rotation gate. (b) Lossy model of rotation gate with two beam-splitters.

An ideal PNR detector gives the number of photons of the signal independently of its other parameters. However, imperfections exist in practice. For example, the detector can not detect any number of photons, so there is a largest photon number. PNR quality is defined in [167] to evaluate the detector's accuracy. There are some other characteristics of a PNR detector: quantum efficiency is 1, instant response time, no noise counts, no limit on the flow rate or on time between two detections, dark count rate, etc. Moreover, the wavelength of the PNR detector is also an essential characteristic in some quantum communication applications [168].

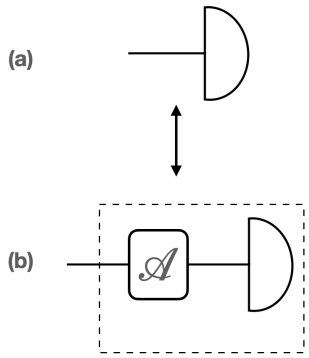


Figure 9.14: Simulate imperfections of detectors. (a) Ideal PNR detector. (b) Lossy model of PNR detector with attenuator.

Our lossy model of the PNR detector is in Fig. 9.14. Compared with the ideal PNR detector shown in Fig. 9.14 (a), we add one attenuator channel before the PNR detector in Fig. 9.14 (b) to describe the imperfections, such as the photon loss or dark count, in the PNR detector. The attenuator channel is defined in Eq. (3.57). The parameter of the attenuator also comes from prior knowledge.

9.5 Conclusion

This chapter showed examples of various optimization tasks by defining a suitable cost function and optimizing the parameterized circuits.

Two libraries based on our work are used: Poenta and MrMustard. Poenta is better for designing a layered neural network consisting of Gaussian and non-Gaussian gates in each layer. Thanks to the CNG algorithm and the fast evolution algorithm under Gaussian operators, Poenta greatly performs Euclidean optimization. On the other hand, MrMustard outperforms in its Riemannian optimization to contain the evolution of the quantum states in phase space as long as possible, which is powerful in optimizing large multi-mode photonic PQCs.

The state preparation task was the first example. First, we showed the single-photon, hex-GKP, and NOON state preparation in Poenta with high fidelity and a significantly reduced run-time. Then, simple circuits are presented with the optimization of the cat state and GKP state in MrMustard.

We proposed some first ideas for a lossy model for gates and PNR detectors to include their defects and the loss in the simulation. Ultimately, with this lossy model, we can better transfer the optimized circuit in numerical simulation into the photonic circuits in practice.

Moreover, our work of optimization PQCs paves the way for quantum device design in future work.

Part III

Conclusion

Chapter 10

Conclusion and perspectives

10.1 Conclusion

We first build a toolbox to bridge the phase space and Fock space representations for Gaussian objects. It is a rich theory that includes pure and mixed states, unitary transformations, and channels. It gives us the possibility to simulate the quantum circuits in phase space representation and then transfer them into Fock space representation. The numerical simulation of Gaussian objects in phase space is much simpler. However, in order to simulate interesting and realistic photonic circuits that include devices such as PNR detectors, it is necessary to work in Fock space representation.

The recurrence relation for generating the Fock amplitudes for Gaussian objects plays an important role in realizing the numerical simulation of photonic PQCs. Especially the generating function of Gaussian unitaries is also related to other mathematical concepts, such as the Hermite polynomials and the Hafnian, which gives more opportunities to find out more properties through their relations. Our approach is also linked to the Fock-Bargmann representation, which has recently been used to describe holomorphic quantum computing [169].

We also find the composition rule of Gaussian operators in Fock space, which allows us to obtain the correct global phase when composing Gaussian operators (which normally is absent from the description of Gaussian objects), and therefore to extend our model to states that can be written as linear combinations of Gaussians.

Because of the recurrence relation's differentiability, we can calculate the gradient of the Gaussian objects. We implement numerically a simulator that can leverage AD and allow us to optimize photonic circuits.

We can execute the optimization task with our quantum circuits. We have implemented two methods of optimizing the Gaussian operators: the Euclidean method optimizes the individual gate parameters for each optical component; the Riemannian method optimizes the entire Gaussian operator as a whole, which we can then decompose into fundamental optical components. To make the optimization fast, we developed and implemented the complex natural gradient algorithm by using the geometry of the underlying parameter space to accelerate gradient descent. We observed that the Riemannian optimization is better than the Euclidean one for solving the problem with a large number of modes.

We presented some state preparation tasks as examples of optimization. We showed

- 8-layer Gaussian and non-Gaussian circuits for the preparation of single-photon state with fidelity up to 99.996% in 15s on a laptop.
- 20-layer Gaussian and non-Gaussian circuits for the preparation of NOON state with fidelity up to 99.913% in 145s on a laptop.
- 25-layer Gaussian and non-Gaussian circuits for the preparation of Hex GKP state with fidelity up to 99.721% in 286s on a laptop.
- simple Gaussian circuits for the heralded preparation of GKP states with fidelity up to 93.01%.

- simple Gaussian circuits for the heralded preparation of cat states with mean photon number 4, fidelity 99.38%, and success probability 7.39%.

We also show the optimization to analyze high-dimensional GBS instances similar to the 216-mode circuit of the photonic processor Borealis [7].

I developed or contributed to several libraries based on this thesis:

- Poenta [60], which optimizes layered-structure circuits with Gaussian and non-Gaussian operators. I contributed to most of the development work.
- MrMustard [61], which optimizes circuits with Gaussian operators and measurements. I contributed to part of the development work.
- TheWalrus [62], where I implemented the calculations of multidimensional Hermite functions.
- StrawberryFields [68], where I implemented the general Gaussian gate in the TF backend.

The library based on this work is powerful and valuable. It can do many tasks: simulate the optical experiments as in the laboratory and find out the best setting with optimization, do the quantum computing work simulation, etc. Moreover, the way to use them is easy. Some code snippets are shown in Appendix D.

Optical quantum computing is a promising platform for fault-tolerant quantum computing, and our work is one important step in that way. For example, the GKP state can be used for quantum error correction, and we look for circuits to produce GKP states.

10.2 Perspectives and ideas for future work

10.2.1 Quantum repeater based on bosonic error correction code

In quantum communication, quantum information becomes degraded as it is transmitted, and the rules of quantum mechanics do not allow signals to be amplified by conventional repeater nodes. A quantum repeater (QR) is needed in this case.

There are three generations of quantum repeaters [170], which are proposed to defend against photon loss and imperfect operations:

- The first-generation QRs generate entanglement signals between neighboring nodes and use entanglement swapping to link them. It asks for also entanglement purification to increase the fidelity of corrupted entanglement resulting from noisy quantum channels.
- The second-generation QRs employ the same entanglement generation as the first one, while the imperfect operations are overcome by quantum error corrections.
- The third-generation QRs aim to use quantum error corrections to overcome both photon loss and imperfect operations.

Our idea is related to the third-generation QRs: quantum information is encoded into logical qubits and transmitted over noisy quantum channels, the quantum repeater will recover the logical qubits, and one can continue to transmit the quantum information in quantum channels and repeaters until the last node. This is the one-way quantum repeater, and it is still in the theoretical stage. The unitary one-way quantum repeater [171] is proved to be workable; theoretically, it has been proved to beat the fundamental repeater-less key rate bound even in the presence of an additional coupling loss. Some specific bosonic error correction codes are also designed to realize the one-way quantum repeater [172, 173].

We, therefore, want to find a way to design the one-way quantum repeater without committing to a specific bosonic code in quantum communication and with no need for quantum memory.

Since PQCs can solve the optimization problem with a suitable loss function, we want to try to design the quantum repeater with it. The simple quantum one-way communication structure, as shown in Fig. 10.1 (a),

includes a quantum encoder, a quantum lossy channel, and a quantum decoder (e.g., the quantum repeater within the transmission and the quantum error corrector at the end). Usually, we encode our quantum information in some specific quantum error correction code (e.g., to the GKP code and cat states).

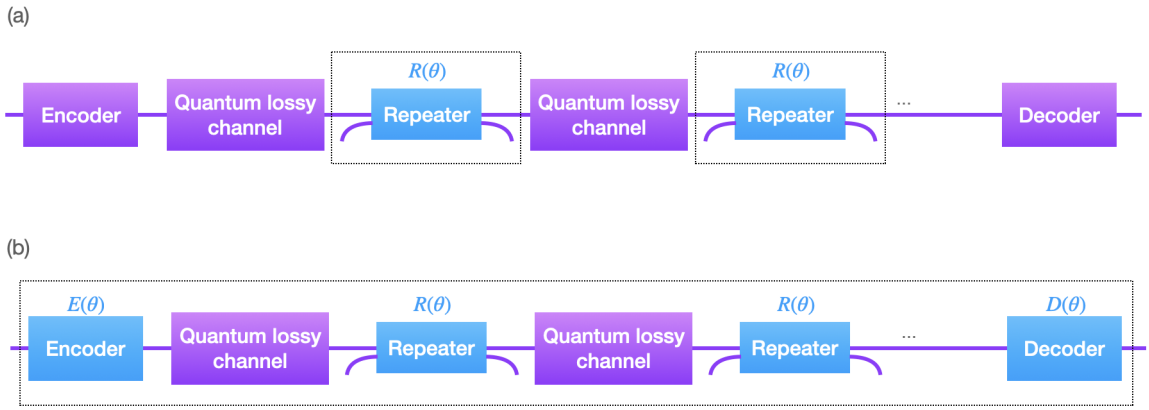


Figure 10.1: Quantum repeater scheme. For scheme (a), we fix an error-correcting code and only train the parametrized quantum repeater during transmission. For scheme (b), we also include the encoder and decoder in training. Hence we do not need to choose a specific code.

However, to achieve the one-way quantum repeater without committing to a specific bosonic code, we propose scheme (b) in Fig. 10.1. The new structure with parameterized quantum circuits combines all elements (encoder, decoder, and quantum repeater) and trains them together. We could conduct numerical simulations to design the exact structure for the one-way quantum repeater and find the bosonic code with reasonable assumptions. Then, it could be implemented as a real component of quantum communication.

10.2.2 Parameterized quantum circuits for quantum devices

Our simulation results showed the ability of parametrized circuits to prepare different quantum states by training the circuit with respect to all parameters for state generation. Hence, this can be considered the quantum state generator, such as the example of the single-photon generator where we use the layered quantum circuits to generate single-photon states in section 9. With the circuit structure we have given, one can design and optimize quantum circuits even though one is unfamiliar with quantum optics.

Moreover, integrated optical devices' development paves the way to transfer our optimized quantum optical circuit into small chips to be used in practice, especially as the quantum devices in quantum communication (such as the quantum emitter, quantum repeater, and quantum receiver). Ideally, the hardware can be designed like Field Programmable Gate Arrays (FPGAs), like quantum FPGA [174]. One can first get the optimized parameters of the circuit in simulation with our libraries, then program it on the quantum FPGA.

10.2.3 Recurrence relation formula for non-Gaussians

This thesis introduces the non-Gaussian effects in our numerical simulation: using the non-Gaussian operator directly, such as the cubic gate or Kerr gate, or adding the measurement at the end of the circuit by using a PNR detector. Although the Kerr gate is very easy to use in quantum simulations because its Fock representation is a diagonal matrix, it is challenging to implement in practice. We, therefore, want to find out the recurrence relation formula for some other non-Gaussian transformations. For instance, in the appendix of the paper [35], the formula of the cubic gate is given.

10.2.4 How to define the minimum resources for quantum state preparation

One interesting question in the machine learning or deep learning community is how to configure the number of layers and nodes in the neural network. These are two essential hyperparameters in the training process. Unfortunately, too many layers and nodes sometimes lead to a complicated and heavy network to train and learn, while one single layer seems insufficient for the task.

For instance, to prepare the GKP state in section 9.2.3, we used two different circuits: one single-mode circuit with 25 layers (Gaussian transformation and non-Gaussian transformation in each layer), while the other three-mode one only has one single Gaussian transformation and two PNR detector measuring two modes. This leads to the question of how to get the minimum resources for one task for quantum state preparation.

There are several ways are recommended in the machine learning community:

- By experimenting: one can change the hyperparameters each time and find out which is the least resources with the highest fidelity (or the lowest loss function). This way takes time and needs some chances.
- By intuition: the prior knowledge can help to estimate the difficulty of the task and pick up the right settings;
- Just go for deeper: most machine learning or deep learning textbooks would suggest that the deeper networks perform better, so it is always better to choose a deeper network.
- Smartly go for deeper: one can also develop the efficient neural architecture search algorithm, which is the hot direction now in *Automated Machine Learning* as neural architecture search.

Even though the paper [97] shows that the classical network can be embedded into the quantum CV neural network. This is a good start for quantum machine learning. However, we are eager to find out what the quantum neural network is capable of and have not started that much-automated architecture searching in quantum formalism.

10.2.5 Riemannian manifold optimization

In our present work, Gaussian objects can be parametrized by elements of the symplectic group, and one can optimize them directly on the corresponding manifold. However, our implemented technique uses the Riemannian gradient to follow geodesics using a fixed learning rate. One step further could be to learn more about the Riemannian manifold optimization problem: such as the generalization of ADAM to Riemannian Manifolds [175] and to add momentum [176] to the optimization, which will improve the speed, stability, and reliability of the training procedure.

Acknowledgement

Several months after I started my Ph.D., Covid19 started. No one would expect it to last for long, but it lasted during almost all of my Ph.D. Doing research during Covid19 is challenging, but finally, here is the end.

Someone told me that the thesis is like your "baby"; you give it birth and raise it. Even though it is not perfect, it is your lovely baby. It was painful to go through it repeatedly, but I've tried my best to improve it. When I get older, thinking back to these three years, I would not know how to describe the experience, but I will never forget that I could go through all of these thanks to everyone I've met. I will always be thankful to see you all.

I would like first to express my gratitude to my primary supervisor, Filippo Miatto, who guided me throughout my Ph.D. Over these three years, his serious attitude toward research and optimistic life wisdom influenced me immensely. As his first Ph.D. student, it is my honor to have such a great professor leading me in my research area.

I want to show my appreciation to my director of the thesis, professor Isabelle Zaquine. As an experienced professor, her guidance and advice helped me a lot. Significantly her handwriting comments contributed to the final birth of this thesis.

My sincere thanks also go to professor Nicolás Quesada, who greatly influenced me with his rigorous academic attitude and impressive work. I appreciate the excellent collaboration with Pierre Cussenot, Richard A. Wolf, and Ebrahim Karimi. I want to thank Xanadu for allowing me to work with them during my residency program and contribute to their open-source library.

I am incredibly grateful to our IQA group members at TP for helping me and discussing with me over the three years, professor Romain Alléaume and professor Peter Brown, and the best colleagues in the world, Dr. Nilesh Vyas, Dr. Raphaël Aymeric, Francesco Mazzoncini, and Guillaume Richard.

I want to thank all my TP friends for staying around me and supporting me, Di Cui, Yi Liu, Yibo Wang, Jingtian Liu, Shihao Ding, Shiyuan Zhao, Zhu Liao, Yinghao Wang, Yibo Quan, Qiong Liu, Lanfan Kong, professor Weiqiang Wen, and professor Heming Huang.

I want to give special thanks to my father, Zhenfang YAO, and my mother, Xueyan ZHAO, who supported me firmly whatever my choices.

Last but not least, I want to thank my lover, Deliang Chang, who is the most beautiful "accident" in my life. I would never regret meeting you before Covid19, loving you during Covid19, and hoping to stay with you 4ever after Covid19.

Appendix A

TensorFlow

A.1 Calculate the partial derivative of complex functions with respect to complex variables

This section introduces how to get the gradient of complex-valued functions with respect to complex-valued variables.

In TF, we have a record operation for automatic differentiation: tf.GradientTape. One could "record" the computation steps inside this tape with TensorFlow's trainable variables, then the gradient with respect to the trainable variables can be obtained with AD. However, TF works only for real-valued variables and functions.

Suppose a complex function $f(z) = U + iV$ with respect to a complex variable $z = x + iy$, where U, V, x, y are real, the partial derivative of function f with respect to z is:

$$\frac{\partial f}{\partial z} = \frac{\partial U}{\partial z} + i\frac{\partial V}{\partial z}.$$

With the Wirtinger derivatives [152], it can be extended to

$$\begin{aligned}\frac{\partial f}{\partial z} &= \frac{\partial U}{\partial z} + i\frac{\partial V}{\partial z} \\ &= \frac{1}{2}\left(\frac{\partial U}{\partial x} - i\frac{\partial U}{\partial y}\right) + \frac{i}{2}\left(\frac{\partial V}{\partial x} - i\frac{\partial V}{\partial y}\right).\end{aligned}$$

And, the partial derivative of f with respect to z^* is

$$\begin{aligned}\frac{\partial f}{\partial z^*} &= \frac{\partial U}{\partial z^*} + i\frac{\partial V}{\partial z^*} \\ &= \frac{1}{2}\left(\frac{\partial U}{\partial x} + i\frac{\partial U}{\partial y}\right) + \frac{i}{2}\left(\frac{\partial V}{\partial x} + i\frac{\partial V}{\partial y}\right).\end{aligned}$$

However, with TF, the gradient function is calculated by combing the two partial derivatives of the real part U of the complex function $f(z)$

$$\text{tf.gradient}(f, z) = \frac{\partial U}{\partial x} + i\frac{\partial U}{\partial y}. \tag{A.1.1}$$

This is not the gradient we expect, which lacks the derivative of the imaginary part V. So we need to get this part and add it to the calculation of the gradient.

We can compute the gradient of function $if = -V + iU$ to turn the imaginary part into the real part of the function, then we have the missing imaginary part:

$$\text{tf.gradient}(if, z) = \frac{\partial(-V)}{\partial x} + i\frac{\partial(-V)}{\partial y}. \tag{A.1.2}$$

Finally, one can calculate the gradients of a complex function with respect to a complex variable:

$$\frac{\partial f}{\partial z} = \frac{1}{2}\left(\text{tf.gradient}(f, z) + i\text{tf.gradient}(if, z)\right)^*,$$

and

$$\frac{\partial f}{\partial z^*} = \frac{1}{2}\left(\text{tf.gradient}(f, z) - i\text{tf.gradient}(if, z)\right).$$

Appendix B

Some facts about the groups

B.1 Definition

The *general linear group* is defined as:

$$\mathrm{GL}(n,k) = \{\boldsymbol{T} \in \boldsymbol{M}(n,k) : \det(T) \neq 0\}, \tag{B.1.1}$$

where $k = \mathbb{R}$ or $\mathbb{C}$, and $\boldsymbol{M}(n,k)$ denotes the space of all $n \times n$ real or complex matrices.

The *real symplectic group* is defined as

$$\mathrm{Sp}(2n,\mathbb{R}) = \{\boldsymbol{S} \in \mathbb{R}^{2n\times 2n} | \boldsymbol{S}\boldsymbol{\Omega}\boldsymbol{S}^T = \boldsymbol{\Omega}\}, \tag{B.1.2}$$

where $\boldsymbol{\Omega}$ is defined in Eq. (3.8). Some properties of this group:

$$\boldsymbol{\Omega} \in \mathrm{Sp}(2n,\mathbb{R}), \tag{B.1.3}$$

$$\boldsymbol{\Omega}^{-1} = \boldsymbol{\Omega}^T = -\boldsymbol{\Omega} \in \mathrm{Sp}(2n,\mathbb{R}), \tag{B.1.4}$$

$$\boldsymbol{S}^{-1} = -\boldsymbol{\Omega}\boldsymbol{S}^T\boldsymbol{\Omega} \in \mathrm{Sp}(2n,\mathbb{R}). \tag{B.1.5}$$

The *orthogonal group* is defined as

$$\mathrm{O}(2n) = \{\boldsymbol{M} \in \mathbb{R}^{2n\times 2n} | \boldsymbol{M}^T\boldsymbol{M} = \boldsymbol{M}\boldsymbol{M}^T = \mathbb{1}_{2n}\}. \tag{B.1.6}$$

B.2 2-out-of-3 property

The *unitary group* is the 3-fold intersection of the orthogonal, complex, and symplectic groups:

$$\mathrm{U}(n) = \mathrm{O}(2n) \cap \mathrm{GL}(n,\mathbb{C}) \cap \mathrm{Sp}(2n,\mathbb{R}). \tag{B.2.1}$$

The definition of the unitary group is

$$\mathrm{U}(n) = \{\boldsymbol{M} \in \mathbb{C}^{n\times n} | \boldsymbol{M}^\dagger\boldsymbol{M} = \boldsymbol{M}\boldsymbol{M}^\dagger = \mathbb{1}_n\}. \tag{B.2.2}$$

B.3 The singular value decomposition

A real symplectic matrix $\boldsymbol{S}$ can be decomposed as

$$\boldsymbol{S} = \boldsymbol{O}_1\boldsymbol{\Lambda}\boldsymbol{O}_2, \tag{B.3.1}$$

with $\boldsymbol{O}_1, \boldsymbol{O}_2 \in C(n)$ and

$$\boldsymbol{\Lambda} = \Lambda_x \otimes \Lambda_x^{-1}, \tag{B.3.2}$$

with $\Lambda_x = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ and $\lambda_j > 0 \forall j \in [1, \ldots, n]$. $C(n)$ denotes the intersection between the real symplectic group and the orthogonal group: $C(n) = Sp_{2n,\mathbb{R}} \cap O(2n)$. Any symplectic matrix can be decomposed into a diagonal and positive semi-definite matrix Z with two orthogonal matrices O_1 and O_2, which stands for the passive transformation (interferometer).

Appendix C

Some facts about the Riemannian manifold

In differential geometry, a Riemannian manifold is a smooth manifold $\mathcal{M}$ equipped with a positive-definite inner product g_p on the tangent space $T_p\mathcal{M}$ at each point p.

C.1 Riemannian metric

The inner product, also called the Riemannian metric, allows for defining distances and angles on the manifold.

The Riemannian metric of the symplectic group at point $\boldsymbol{A}$ is:

$$\langle \boldsymbol{X}, \boldsymbol{Y} \rangle_{\boldsymbol{A}} = \langle \boldsymbol{A}^{-1}\boldsymbol{X}, \boldsymbol{A}^{-1}\boldsymbol{Y} \rangle_{\mathbb{1}_{2n}} = \mathrm{Tr}\left((\boldsymbol{A}^{-1}\boldsymbol{X})^T \boldsymbol{A}^{-1}\boldsymbol{Y} \right), \quad \boldsymbol{X}, \boldsymbol{Y} \in T_{\boldsymbol{A}}\mathrm{Sp}(2n, \mathbb{R}). \tag{C.1.1}$$

The same for the orthogonal group.

The Riemannian metric of the unitary group at point $\boldsymbol{A}$ is:

$$\langle \boldsymbol{X}, \boldsymbol{Y} \rangle_{\boldsymbol{A}} = \langle \boldsymbol{A}^{-1}\boldsymbol{X}, \boldsymbol{A}^{-1}\boldsymbol{Y} \rangle_{\mathbb{1}_{2n}} = \mathrm{Tr}\left((\boldsymbol{A}^{-1}\boldsymbol{X})^\dagger \boldsymbol{A}^{-1}\boldsymbol{Y} \right), \quad \boldsymbol{X}, \boldsymbol{Y} \in T_{\boldsymbol{A}}\mathrm{U}(n, \mathbb{C}). \tag{C.1.2}$$

C.2 Tangent space

The tangent bundle (or Tangent space) of a smooth manifold $\mathcal{M}$ assigns to each point $\boldsymbol{A}$ of $\mathcal{M}$ a vector space: the tangent space $T_{\boldsymbol{A}}\mathcal{M}$ of $\mathcal{M}$ at $\boldsymbol{A}$.

For the symplectic group, we differentiate the quadratic condition $\boldsymbol{A\Omega A}^T = \boldsymbol{\Omega}$ in Eq. (B.1.2), we obtain the *linear* tangency condition $\boldsymbol{X\Omega A}^T + \boldsymbol{A\Omega X}^T = 0$:

$$T_{\boldsymbol{A}}\mathrm{Sp} = \{\boldsymbol{X} \in \mathbb{R}^{2n\times 2n} | \boldsymbol{X\Omega A}^T + \boldsymbol{A\Omega X}^T = 0_{2n}\}. \tag{C.2.1}$$

It can be expressed in a compact way by parametrizing the tangent space at $\boldsymbol{A}$ using symmetric matrices with the tangency condition $\boldsymbol{X} = \boldsymbol{A\Omega T}$. Then we have:

$$T_{\boldsymbol{A}}\mathrm{Sp} = \{\boldsymbol{X} \in \mathbb{R}^{2n\times 2n} | \boldsymbol{X\Omega A}^T + \boldsymbol{A\Omega X}^T = 0_{2n}\} \tag{C.2.2}$$

$$= \{\boldsymbol{X} = \boldsymbol{A\Omega T} | \boldsymbol{T} = \boldsymbol{T}^T\}. \tag{C.2.3}$$

As a special case, the Lie algebra of Sp is the tangent space at the identity, i.e.

$$\mathrm{sp}(2n, \mathbb{R}) = T_e\mathrm{Sp}(2n, \mathbb{R}) \tag{C.2.4}$$

$$= \{\boldsymbol{X} \in \mathbb{R}^{2n\times 2n} | \boldsymbol{X\Omega} + \boldsymbol{\Omega X}^T = 0_{2n}\} \tag{C.2.5}$$

$$= \{\boldsymbol{\Omega A} | \boldsymbol{A} = \boldsymbol{A}^T\}. \tag{C.2.6}$$

With the same idea, we can calculate the tangent space of an orthogonal group $\mathrm{O}(2n)$ at $\boldsymbol{A}$:

$$T_{\boldsymbol{A}}\mathrm{O} = \{\boldsymbol{X} \in \mathbb{R}^{2n\times 2n} | \boldsymbol{X}^T\boldsymbol{A} + \boldsymbol{A}^T\boldsymbol{X} = 0_{2n}\} \tag{C.2.7}$$

$$= \{\boldsymbol{X} = \boldsymbol{A}\boldsymbol{T} | \boldsymbol{T}^T = -\boldsymbol{T}\}. \tag{C.2.8}$$

The tangent space of a unitary group $\mathrm{U}(n)$ at $\boldsymbol{A}$ is

$$T_{\boldsymbol{A}}\mathrm{U} = \{\boldsymbol{X} \in \mathbb{C}^{n\times n} | \boldsymbol{X}^\dagger\boldsymbol{A} + \boldsymbol{A}^\dagger\boldsymbol{X} = 0_{n}\} \tag{C.2.9}$$

$$= \{\boldsymbol{X} = \boldsymbol{A}\boldsymbol{T} | \boldsymbol{T}^\dagger = -\boldsymbol{T}\}. \tag{C.2.10}$$

C.3 Normal space

We can then define the normal space at $\boldsymbol{A}$ as the linear space containing all the elements that are orthogonal to $T_{\boldsymbol{A}}\mathrm{Sp}$:

$$N_{\boldsymbol{A}}\mathrm{Sp} = \{\boldsymbol{W} \in \mathbb{R}^{2n\times 2n} | \mathrm{Tr}(\boldsymbol{W}^T\boldsymbol{X}) = 0_{2n}, \forall \boldsymbol{X} \in T_{\boldsymbol{A}}\mathrm{Sp}\} \tag{C.3.1}$$

$$= \{\boldsymbol{W} = \boldsymbol{\Omega}\boldsymbol{A}\boldsymbol{N} | \boldsymbol{N} \in \mathbb{R}^{2n\times 2n}, \boldsymbol{N} = -\boldsymbol{N}^T\}. \tag{C.3.2}$$

The normal space at $\boldsymbol{A}$ for an orthogonal group is:

$$N_{\boldsymbol{A}}\mathrm{O} = \{\boldsymbol{W} \in \mathbb{R}^{2n\times 2n} | \mathrm{Tr}(\boldsymbol{W}^T\boldsymbol{X}) = 0_{2n}, \forall \boldsymbol{X} \in T_{\boldsymbol{A}}\mathrm{O}\} \tag{C.3.3}$$

$$= \{\boldsymbol{W} = \boldsymbol{N}\boldsymbol{A} | \boldsymbol{N} \in \mathbb{R}^{2n\times 2n}, \boldsymbol{N} = \boldsymbol{N}^T\}. \tag{C.3.4}$$

The normal space at $\boldsymbol{A}$ for a unitary group is:

$$N_{\boldsymbol{A}}\mathrm{U} = \{\boldsymbol{W} \in \mathbb{C}^{n\times n} | \mathrm{Tr}(\boldsymbol{W}^\dagger\boldsymbol{X}) = 0_{n}, \forall \boldsymbol{X} \in T_{\boldsymbol{A}}\mathrm{U}\} \tag{C.3.5}$$

$$= \{\boldsymbol{W} = \boldsymbol{N}\boldsymbol{A} | \boldsymbol{N} \in \mathbb{C}^{n\times n}, \boldsymbol{N} = \boldsymbol{N}^\dagger\}. \tag{C.3.6}$$

C.4 Riemannian gradient

The Euclidean gradient ∂f at the point $\boldsymbol{A}$ (which is computed using the embedding coordinates in $\mathbb{R}^{2n\times 2n}$) is related to the Riemannian gradient $\nabla L \in T_{\boldsymbol{A}}\mathrm{Sp}$. Now we will show the calculation of the Riemannian gradient with the symplectic group, orthogonal group, and unitary group.

The Riemannian gradient $\nabla_{\boldsymbol{A}} f$ at point $\boldsymbol{A}$ of a sufficiently regular function $f : \mathrm{Sp}(2n, \mathbb{R} \to \mathbb{R})$ associated to the Riemannian metric satisfies [59]

$$\nabla_{\boldsymbol{A}} f = \frac{1}{2}\left(\boldsymbol{A}\boldsymbol{A}^T\partial_{\boldsymbol{A}} f + \boldsymbol{A}\boldsymbol{\Omega}\partial_{\boldsymbol{A}}^T f\boldsymbol{A}\boldsymbol{\Omega}\right). \tag{C.4.1}$$

The Riemannian gradient $\nabla_{\boldsymbol{A}} f$ at point $\boldsymbol{A}$ of a sufficiently regular function $f : \mathrm{O}(2n, \mathbb{R} \to \mathbb{R})$ associated to the Riemannian metric satisfies [54]

$$\nabla_{\boldsymbol{A}} f = \frac{1}{2}\left(\partial_{\boldsymbol{A}} f - \boldsymbol{A}\partial_{\boldsymbol{A}}^T f\boldsymbol{A}\right). \tag{C.4.2}$$

The Riemannian gradient $\nabla_{\boldsymbol{A}} f$ at point $\boldsymbol{A}$ of a sufficiently regular function $f : \mathrm{U}(n, \mathbb{C} \to \mathbb{C})$ associated to the Riemannian metric satisfies

$$\nabla_{\boldsymbol{A}} f = \frac{1}{2}\left(\partial_{\boldsymbol{A}} f - \boldsymbol{A}\partial_{\boldsymbol{A}}^\dagger f\boldsymbol{A}\right). \tag{C.4.3}$$

Proof. According to the compatibility of the Riemannian gradient with the Riemannian metric (defined in Eq. (C.1.2)), we have:

$$\langle \nabla_{\boldsymbol{A}} f, \boldsymbol{T} \rangle_{\boldsymbol{A}} = \langle \partial_{\boldsymbol{A}} f, \boldsymbol{T} \rangle_{\mathrm{euc}}, \quad \forall \boldsymbol{T} \in T_{\boldsymbol{A}}\mathrm{Sp}, \tag{C.4.4}$$

that it,

$$\langle \partial_{\boldsymbol{A}} f - \boldsymbol{A}^{-\dagger}\boldsymbol{A}^{-1}\nabla_{\boldsymbol{A}} f, \boldsymbol{T} \rangle_{\mathrm{euc}} = 0. \tag{C.4.5}$$

This implies that $\partial_{\boldsymbol{A}} f - \boldsymbol{A}^{-\dagger}\boldsymbol{A}^{-1}\nabla_{\boldsymbol{A}} f \in N_{\boldsymbol{A}}\mathrm{U}$. So we have, with $\boldsymbol{A}\boldsymbol{A}^{\dagger} = \mathbb{1}$:

$$\partial_{\boldsymbol{A}} f - \boldsymbol{A}^{-\dagger}\boldsymbol{A}^{-1}\nabla_{\boldsymbol{A}} f = \boldsymbol{A}\boldsymbol{N} \tag{C.4.6}$$

$$\partial_{\boldsymbol{A}} f = \nabla_{\boldsymbol{A}} f + \boldsymbol{A}\boldsymbol{N}. \tag{C.4.7}$$

Using the tangency condition $\nabla_{\boldsymbol{A}} f \in T_{\boldsymbol{A}}\mathrm{U}$, we know

$$(\nabla_{\boldsymbol{A}} f)^{\dagger}\boldsymbol{A} + \boldsymbol{A}^{\dagger}\nabla_{\boldsymbol{A}} f = 0_n. \tag{C.4.8}$$

Together Eq. (C.4.8) and Eq. (C.4.7) with $\boldsymbol{N} = \boldsymbol{N}^{\dagger}$, we solve

$$\boldsymbol{N} = \frac{1}{2}\left(\boldsymbol{A}^{\dagger}\partial_{\boldsymbol{A}} f + \partial_{\boldsymbol{A}}^{\dagger} f \boldsymbol{A}\right). \tag{C.4.9}$$

Thus we obtain

$$\nabla_{\boldsymbol{A}} f = \partial_{\boldsymbol{A}} f - \boldsymbol{A}\frac{1}{2}\left(\boldsymbol{A}^{\dagger}\partial_{\boldsymbol{A}} f + \partial_{\boldsymbol{A}}^{\dagger} f \boldsymbol{A}\right) \tag{C.4.10}$$

$$= \frac{1}{2}\left(\partial_{\boldsymbol{A}} f - \boldsymbol{A}\partial_{\boldsymbol{A}}^{\dagger} f \boldsymbol{A}\right). \tag{C.4.11}$$

□

Appendix D

Code snippet

This section gives some code snippets to show the functionalities and feasibility of our libraries, Poenta and MrMustard.

D.1 Poenta

Poenta is a predefined library to generate layered quantum circuits and train them. In each layer, we predefine the Gaussian unitaries and non-Gaussian unitaries: the Gaussian unitaries are decomposed by the Bloch-Messiah algorithm, and we provide only Kerr gate as the non-Gaussian operation. The cost function is also predefined as the one minus the quantum state fidelity.

In Fig. D.1, we give an example to generate the single-photon state. One only needs to define the initial state, target state, and hyperparameters (layer, number of modes). A learning rate scheduler is employed for the optimization, and one can choose the initial learning rate and the optimizer. Moreover, one can choose to use the complex natural gradient or not.

```
import numpy as np
from poenta.circuit import Circuit
from poenta.nputils import single_photon,vaccum,NOON
import tensorflow as tf

scheduler_status = False

cutoff = 100
state_in = vaccum(1,cutoff)
target_out = single_photon(1,cutoff)

device = Circuit(num_layers=8, num_modes=1, num_seed=520, dtype=tf.complex128)

tuple_in_out = (state_in,target_out),
device.set_input_output_pairs(*tuple_in_out)
device.optimize(steps = 1500,optimizer = "Adam",learning_rate = 0.001,
                scheduler = False, nat_grad = False)

```

Figure D.1: The code snippet for the circuit in Poenta.

The cost function evolution is shown in Fig. D.2.

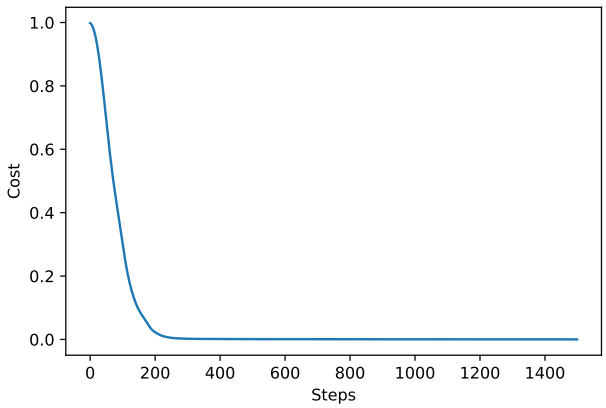


Figure D.2: The cost function of single-photon state preparation.

After training, the output state vector is $|\psi\rangle_{\text{out}} = [6.82938689e{-}04{-}5.17996126e{-}04j, 7.40702608e{-}01{+}6.71675751e{-}01j, {-}8.41573328e{-}04{+}3.11787605e{-}04j, {-}1.57652647e{-}04{+}1.68166768e{-}04j, 2.72393795e{-}04{+}6.03049731e{-}04j, 1.61324384e{-}04{+}1.58101522e{-}03j, 9.49653617e{-}04{+}2.01800991e{-}03j, 2.59675565e{-}03 - 1.78774201e - 03j, \dots]$. The probability is obtained by $||\psi\rangle_{\text{out}}|^2 = [7.34725240e - 07, 9.99788668e - 01, 8.05457178e - 07, 5.31344191e - 08, 4.37867358e - 07, 2.52563469e - 06, 4.97420600e - 06, 9.93916138e - 06, \dots]$, where the second term 0.9998 is the probability of getting single photon.

This example is single-mode, so we have four gates in each layer: displacement gate $\mathcal{D}(\gamma)$, rotation gate $\mathcal{R}(\phi)$, squeezing gate $\mathcal{S}(\zeta)$, and Kerr gate $K(\kappa)$ in each layer. The optimized circuit's parameters are shown in Tab. D.1.

Layer	γ	ϕ	ζ	κ
1	0.126+0.038j	-0.045	0.293-0.011j	-0.023
2	0.243+0.159j	-0.074	0.199+0.001j	0.118
3	0.099+0.061j	-0.091	0.190-0.039j	0.143
4	0.196-0.036j	-0.124	0.211-0.051j	0.140
5	0.072+0.090j	0.127	0.179-0.138j	0.140
6	0.150+0.046j	0.082	0.242-0.033j	0.178
7	0.200+0.060j	0.140	0.195+0.023j	0.310
8	0.212-0.010j	0.170	0.106+0.025j	-0.115

Table D.1: The values of the parameter in the circuit to prepare the single-photon state.

D.2 MrMustard

MrMustard is a more free library compared with Poenta. The user will not be contained in the layered structure and can choose any gates to design the circuit together with any kind of measurement.

Like the two examples in Figs. D.3 and D.4, one needs to define the cost function and connect all elements in the circuit.

```
#Loss function
def cost_fn():
    ket = output().ket(cutoffs=[cutoff])
    return -math.abs(math.sum(math.conj(gkp0_ket) * ket))**2
#Full train circuit
circ = Ggate(num_modes=3,symplectic_trainable=True)
def output():
    return Vacuum(3) >> circ << Fock(6, modes =[0], normalize=True)
           << Fock(6, modes =[1], normalize=True)
#Optimizer
opt = Optimizer(symplectic_lr = 0.005)
#Minimize the cost function
opt.minimize(cost_fn, by_optimizing=[circ], max_steps=2000)
```

Figure D.3: The code snippet for the circuit shown in Fig. 9.3a.

```
#Loss function
def cost_fn():
    ket = output().ket(cutoffs=[cutoff])
    return -math.abs(math.sum(math.conj(cat_ket) * ket))**2
#Full train circuit
circ = Ggate(num_modes=2,symplectic_trainable=True)
def output():
return Vacuum(2) >> circ << Fock(3, modes =[0], normalize=True)
#Optimizer
opt = Optimizer(symplectic_lr = 0.005)
#Minimize the cost function
opt.minimize(cost_fn, by_optimizing=[circ], max_steps=2000)
```

Figure D.4: The code snippet for the circuit shown in Fig. 9.4a.

Another advantage of MrMustard is one can set bound values for each parameter. For example, it is possible to set the upper bound u and lower bound l of a rotation operator's parameter ϕ when defining this object: In this way, we can have more reasonably optimized parameters to realize them in practice.

```
Rgate(angle = 0.2, angle_trainable=True, angle_bounds= (0,0.5))
```

Bibliography

[1] Sands Matthew Feynman Richard, Leighton Robert. The feynman lectures on physics. *California Institute of Technology*, 3, 1964, 2006, 2010.

[2] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6-7):467–488, June 1982.

[3] P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[4] Craig Gidney and Martin Ekerå. How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. *Quantum*, 5:433, apr 2021.

[5] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, oct 2019.

[6] Han-Sen Zhong, Hui Wang, Yu-Hao Deng, Ming-Cheng Chen, Li-Chao Peng, Yi-Han Luo, Jian Qin, Dian Wu, Xing Ding, Yi Hu, Peng Hu, Xiao-Yan Yang, Wei-Jun Zhang, Hao Li, Yuxuan Li, Xiao Jiang, Lin Gan, Guangwen Yang, Lixing You, Zhen Wang, Li Li, Nai-Le Liu, Chao-Yang Lu, and Jian-Wei Pan. Quantum computational advantage using photons. *Science*, 370(6523):1460–1463, dec 2020.

[7] Lars S. Madsen, Fabian Laudenbach, Mohsen Falamarzi. Askarani, Fabien Rortais, Trevor Vincent, Jacob F. F. Bulmer, Filippo M. Miatto, Leonhard Neuhaus, Lukas G. Helt, Matthew J. Collins, Adriana E. Lita, Thomas Gerrits, Sae Woo Nam, Varun D. Vaidya, Matteo Menotti, Ish Dhand, Zachary Vernon, Nicolás Quesada, and Jonathan Lavoie. Quantum computational advantage with a programmable photonic processor. *Nature*, 606(7912):75–81, jun 2022.

[8] Craig S Hamilton, Regina Kruse, Linda Sansoni, Sonja Barkhofen, Christine Silberhorn, and Igor Jex. Gaussian boson sampling. *Physical review letters*, 119(17):170501, 2017.

[9] Nicolás Quesada and Juan Miguel Arrazola. Exact simulation of gaussian boson sampling in polynomial space and exponential time. *Phys. Rev. Research*, 2:023005, Apr 2020.

[10] Kosuke Fukui, Akihisa Tomita, Atsushi Okamoto, and Keisuke Fujii. High-threshold fault-tolerant quantum computation with analog quantum error correction. *Phys. Rev. X*, 8:021054, May 2018.

[11] Kai Sun, Ze-Yan Hao, Yan Wang, Jia-Kun Li, Xiao-Ye Xu, Jin-Shi Xu, Yong-Jian Han, Chuan-Feng Li, and Guang-Can Guo. Optical demonstration of quantum fault-tolerant threshold. *Light: Science & Applications*, 11(1), July 2022.

[12] Alán Aspuru-Guzik and Philip Walther. Photonic quantum simulators. *Nature Physics*, 8(4):285–291, apr 2012.

[13] He-Liang Huang, Dachao Wu, Daojin Fan, and Xiaobo Zhu. Superconducting quantum computing: a review. *Science China Information Sciences*, 63(8), jul 2020.

[14] C S Adams, J D Pritchard, and J P Shaffer. Rydberg atom quantum technologies. *Journal of Physics B: Atomic, Molecular and Optical Physics*, 53(1):012002, dec 2019.

[15] Lieven M. K. Vandersypen and Mark A. Eriksson. Quantum computing with semiconductor spins. *Physics Today*, 72(8):38–45, aug 2019.

[16] David Kielpinski, Chris Monroe, and David J Wineland. Architecture for a large-scale ion-trap quantum computer. *Nature*, 417(6890):709–711, 2002.

[17] Mhd Hasan Sarhan, M. Ali Nasseri, Daniel Zapp, Mathias Maier, Chris P. Lohmann, Nassir Navab, and Abouzar Eslami. Machine learning techniques for ophthalmic data processing: A review. *IEEE Journal of Biomedical and Health Informatics*, 24(12):3338–3350, 2020.

[18] Vedran Dunjko, Jacob M. Taylor, and Hans J. Briegel. Quantum-enhanced machine learning. *Physical Review Letters*, 117(13), September 2016.

[19] Aamir Mandviwalla, Keita Ohshiro, and Bo Ji. Implementing grover's algorithm on the ibm quantum computers. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 2531–2537, 2018.

[20] Jarrod R McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, 18(2):023023, February 2016.

[21] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm, 2014.

[22] Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini. Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 4(4):043001, November 2019.

[23] Daoyi Dong, Chunlin Chen, Hanxiong Li, and Tzyh-Jong Tarn. Quantum reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(5):1207–1220, 2008.

[24] Philipp Hauke, Helmut G Katzgraber, Wolfgang Lechner, Hidetoshi Nishimori, and William D Oliver. Perspectives of quantum annealing: methods and implementations. *Reports on Progress in Physics*, 83(5):054401, may 2020.

[25] Samuel Loomis. Exploring hidden quantum markov models. *Bulletin of the American Physical Society*, 2017.

[26] Christopher Gerry and Peter Knight. *Introductory Quantum Optics*. Cambridge University Press, October 2004.

[27] Gianfranco Cariolaro. *Quantum Communications*. Springer International Publishing, 2015.

[28] Mohamed Amazioug, Mostafa Nassik, and Nabil Habiballah. Entanglement, EPR steering and gaussian geometric discord in a double cavity optomechanical systems. *The European Physical Journal D*, 72(10), oct 2018.

[29] M W Matsen. The standard gaussian model for block copolymer melts. *Journal of Physics: Condensed Matter*, 14(2):R21–R47, dec 2001.

[30] E.V. Doktorov, I.A. Malkin, and V.I. Man'ko. Dynamical symmetry of vibronic transitions in polyatomic molecules and the franck-condon principle. *Journal of Molecular Spectroscopy*, 56(1):1–20, apr 1975.

[31] R. Berger, C. Fischer, and M. Klessinger. Calculation of the vibronic fine structure in electronic spectra at higher temperatures. 1. benzene and pyrazine. *The Journal of Physical Chemistry A*, 102(36):7157–7167, aug 1998.

[32] Daniel Gruner and Paul Brumer. Efficient evaluation of harmonic polyatomic franck-condon factors. *Chemical Physics Letters*, 138(4):310–314, jul 1987.

[33] Scott M. Rabidoux, Victor Eijkhout, and John F. Stanton. A highly-efficient implementation of the doktorov recurrence equations for franck–condon calculations. *Journal of Chemical Theory and Computation*, 12(2):728–739, jan 2016.

[34] Joonsuk Huh. Multimode bogoliubov transformation and husimi's q-function. *Journal of Physics: Conference Series*, 1612(1):012015, aug 2020.

[35] Filippo M Miatto and Nicolás Quesada. Fast optimization of parametrized quantum optical circuits. *Quantum*, 4:366, 2020.

[36] Augustin Cauchy et al. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538, 1847.

[37] James Stokes, Josh Izaac, Nathan Killoran, and Giuseppe Carleo. Quantum Natural Gradient. *Quantum*, 4:269, May 2020.

[38] Lucas Hackl, Tommaso Guaita, Tao Shi, Jutho Haegeman, Eugene Demler, and Ignacio Cirac. Geometry of variational methods: dynamics of closed quantum systems. *SciPost Physics*, 9(4), oct 2020.

[39] Shun-ichi Amari. Neural learning in structured parameter spaces - natural riemannian gradient. In M.C. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, pages 127–133. MIT Press, 1996.

[40] Shun ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, February 1998.

[41] S. Amari and S. C. Douglas. Why natural gradient? In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98 (Cat. No.98CH36181)*, volume 2, pages 1213–1216 vol.2, 1998.

[42] Naoki Yamamoto. On the natural gradient for variational quantum eigensolver. *arXiv preprint arXiv:1909.05074*, 2019.

[43] Tobias Haug and MS Kim. Optimal training of variational quantum algorithms without barren plateaus. *arXiv preprint arXiv:2104.14543*, 2021.

[44] Bálint Koczor and Simon C. Benjamin. Quantum natural gradient generalised to non-unitary circuits. *arXiv preprint arXiv:1912.08660v4*, 2020.

[45] David Wierichs, Christian Gogolin, and Michael Kastoryano. Avoiding local minima in variational quantum eigensolvers with the natural gradient optimizer. *Physical Review Research*, 2(4):043246, 2020.

[46] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *Phys. Rev. A*, 99:032331, Mar 2019.

[47] Xiao Yuan, Suguru Endo, Qi Zhao, Ying Li, and Simon C. Benjamin. Theory of variational quantum simulation. *Quantum*, 3:191, Oct 2019.

[48] Ying Li and Simon C. Benjamin. Efficient variational quantum simulator incorporating active error minimization. *Phys. Rev. X*, 7:021050, Jun 2017.

[49] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, Aug 2018.

[50] Barnaby van Straaten and Bálint Koczor. Measurement cost of metric-aware variational quantum algorithms. *PRX Quantum*, 2(3):030324, 2021.

[51] Thomas R Bromley, Juan Miguel Arrazola, Soran Jahangiri, Josh Izaac, Nicolás Quesada, Alain Delgado Gran, Maria Schuld, Jeremy Swinarton, Zeid Zabaneh, and Nathan Killoran. Applications of near-term photonic quantum computers: software and algorithms. *Quantum Science and Technology*, 5(3):034010, may 2020.

[52] Yuan Yao, Pierre Cussenot, Richard A. Wolf, and Filippo Miatto. Complex natural gradient optimization for optical quantum circuit design. *Phys. Rev. A*, 105:052402, May 2022.

[53] S. Amari and H. Nagaoka. *Methods of Information Geometry*. Translations of mathematical monographs. American Mathematical Society, 2000.

[54] Simone Fiori and Yoshua Bengio. Quasi-geodesic neural learning algorithms over the orthogonal group: A tutorial. *Journal of Machine Learning Research*, 6(5), 2005.

[55] Y. Yang. Globally convergent optimization algorithms on riemannian manifolds: Uniform framework for unconstrained and constrained optimization. *Journal of Optimization Theory and Applications*, 132(2):245–265, dec 2006.

[56] Steven T Smith. Optimization techniques on riemannian manifolds. *Fields institute communications*, 3(3):113–135, 1994.

[57] D. Gabay. Minimizing a differentiable function over a differential manifold. *Journal of Optimization Theory and Applications*, 37(2):177–219, jun 1982.

[58] Hiroyuki Sato and Toshihiro Iwai. A new, globally convergent riemannian conjugate gradient method. *Optimization*, 64(4):1011–1031, 2015.

[59] Jing Wang, Huafei Sun, and Simone Fiori. A riemannian-steepest-descent approach for optimization on the real symplectic group. *Mathematical Methods in the Applied Sciences*, 41(11):4273–4286, 2018.

[60] Miatto research group. Poenta. https://github.com/Miatto-research-group/Poenta, 2020.

[61] XanaduAI. Mrmustard. https://github.com/XanaduAI/MrMustard, 2022.

[62] XanaduAI. Thewalrus. https://github.com/XanaduAI/thewalrus, 2019.

[63] Nathan Killoran, Josh Izaac, Nicolás Quesada, Ville Bergholm, Matthew Amy, and Christian Weedbrook. Strawberry Fields: A software platform for photonic quantum computing. *Quantum*, 3:129, 2019.

[64] J.R. Johansson, P.D. Nation, and Franco Nori. QuTiP: An open-source python framework for the dynamics of open quantum systems. *Computer Physics Communications*, 183(8):1760–1772, aug 2012.

[65] J.R. Johansson, P.D. Nation, and Franco Nori. QuTiP 2: A python framework for the dynamics of open quantum systems. *Computer Physics Communications*, 184(4):1234–1240, apr 2013.

[66] Nicolas Heurtel, Shane Mansfield, Jean Senellart, and Benoît Valiron. Strong simulation of linear optical processes, 2022.

[67] Nicolas Heurtel, Andreas Fyrillas, Grégoire de Gliniasty, Raphaël Le Bihan, Sébastien Malherbe, Marceau Pailhas, Boris Bourdoncle, Pierre-Emmanuel Emeriau, Rawad Mezher, Luka Music, Nadia Belabas, Benoît Valiron, Pascale Senellart, Shane Mansfield, and Jean Senellart. Perceval: A software platform for discrete variable photonic quantum computing, 2022.

[68] XanaduAI. Strawberryfields. https://github.com/XanaduAI/strawberryfields, 2019.

[69] Paul Nation, Robert Johansson. Qutip. https://github.com/qutip, 2013.

[70] Quandela. Perceval. https://github.com/Quandela/Perceval, 2022.

[71] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, June 2012.

[72] M. Fox. *Quantum Optics: An Introduction*. Oxford Master Series in Physics. OUP Oxford, 2006.

[73] Richard Jozsa. Fidelity for mixed quantum states. *Journal of Modern Optics*, 41(12):2315–2323, dec 1994.

[74] Rene Daendliker. Concept of modes in optics and photonics. In Jose Javier Sanchez-Mondragon, editor, *Sixth International Conference on Education and Training in Optics and Photonics*, volume 3831, pages 193 – 198. International Society for Optics and Photonics, SPIE, 2000.

[75] Gerald B Folland. Harmonic analysis in phase space.(am-122), volume 122. In *Harmonic Analysis in Phase Space.(AM-122), Volume 122*. Princeton university press, 2016.

[76] K. E. Cahill and R. J. Glauber. Density operators and quasiprobability distributions. *Phys. Rev.*, 177:1882–1902, Jan 1969.

[77] C Brif and A Mann. A general theory of phase-space quasiprobability distributions. *Journal of Physics A: Mathematical and General*, 31(1):L9–L17, January 1998.

[78] Alessio Serafini. *Quantum continuous variables: a primer of theoretical methods*. CRC press, 2017.

[79] Timjan Kalajdzievski and Nicolás Quesada. Exact and approximate continuous-variable gate decompositions. *Quantum*, 5:394, 2021.

[80] Xin Ma and William Rhodes. Multimode squeeze operators and squeezed states. *Physical Review A*, 41(9):4625–4631, may 1990.

[81] Mattia Walschaers. Non-gaussian quantum states and where to find them. *PRX Quantum*, 2(3):030204, 2021.

[82] Raphael Hunger. An introduction to complex differentials and complex differentiability. Technical report, Munich University of Technology, Inst. for Circuit Theory and Signal Processing, 2007.

[83] Chu Guo and Dario Poletti. Scheme for automatic differentiation of complex loss functions with applications in quantum physics. *Physical Review E*, 103(1), January 2021.

[84] Raphael Hunger. An introduction to complex differentials and complex differentiability. *Technical report, Munich University of Technology, Inst. for Circuit Theory and Signal Processing*, 2007.

[85] Chu Guo and Dario Poletti. Scheme for automatic differentiation of complex loss functions with applications in quantum physics. *Phys. Rev. E*, 103:013309, Jan 2021.

[86] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2016.

[87] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[88] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[89] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[90] Kosuke Fukui and Shuntaro Takeda. Building a large-scale quantum computer with continuous-variable optical technologies. *Journal of Physics B: Atomic, Molecular and Optical Physics*, 55(1):012001, jan 2022.

[91] 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):1–7, 2014.

[92] Jules Tilly, Hongxiang Chen, Shuxiang Cao, Dario Picozzi, Kanav Setia, Ying Li, Edward Grant, Leonard Wossnig, Ivan Rungger, George H. Booth, and Jonathan Tennyson. The variational quantum eigensolver: a review of methods and best practices, 2021.

[93] Yulin Chi, Jieshan Huang, Zhanchuan Zhang, Jun Mao, Zinan Zhou, Xiaojiong Chen, Chonghao Zhai, Jueming Bao, Tianxiang Dai, Huihong Yuan, Ming Zhang, Daoxin Dai, Bo Tang, Yan Yang, Zhihua Li, Yunhong Ding, Leif K. Oxenløwe, Mark G. Thompson, Jeremy L. O'Brien, Yan Li, Qihuang Gong, and Jianwei Wang. A programmable qudit-based quantum processor. *Nature Communications*, 13(1), mar 2022.

[94] T. F. Watson, S. G. J. Philips, E. Kawakami, D. R. Ward, P. Scarlino, M. Veldhorst, D. E. Savage, M. G. Lagally, Mark Friesen, S. N. Coppersmith, M. A. Eriksson, and L. M. K. Vandersypen. A programmable two-qubit quantum processor in silicon. *Nature*, 555(7698):633–637, feb 2018.

[95] Martina Gschwendtner and Andreas Winter. Infinite-dimensional programmable quantum processors. *PRX Quantum*, 2:030308, Jul 2021.

[96] Sepehr Ebadi, Tout T Wang, Harry Levine, Alexander Keesling, Giulia Semeghini, Ahmed Omran, Dolev Bluvstein, Rhine Samajdar, Hannes Pichler, Wen Wei Ho, et al. Quantum phases of matter on a 256-atom programmable quantum simulator. *Nature*, 595(7866):227–232, 2021.

[97] Nathan Killoran, Thomas R Bromley, Juan Miguel Arrazola, Maria Schuld, Nicolás Quesada, and Seth Lloyd. Continuous-variable quantum neural networks. *Physical Review Research*, 1(3):033063, 2019.

[98] Seth Lloyd and Samuel L. Braunstein. Quantum computation over continuous variables. *Phys. Rev. Lett.*, 82:1784–1787, Feb 1999.

[99] C. K. Hong, Z. Y. Ou, and L. Mandel. Measurement of subpicosecond time intervals between two photons by interference. *Phys. Rev. Lett.*, 59:2044–2046, Nov 1987.

[100] Frédéric Bouchard, Alicia Sit, Yingwen Zhang, Robert Fickler, Filippo M Miatto, Yuan Yao, Fabio Sciarrino, and Ebrahim Karimi. Two-photon interference: the hong–ou–mandel effect. *Reports on Progress in Physics*, 84(1):012402, dec 2020.

[101] Pieter Kok, W. J. Munro, Kae Nemoto, T. C. Ralph, Jonathan P. Dowling, and G. J. Milburn. Linear optical quantum computing with photonic qubits. *Rev. Mod. Phys.*, 79:135–174, Jan 2007.

[102] E. Knill, R. Laflamme, and G. J. Milburn. A scheme for efficient quantum computation with linear optics. *Nature*, 409(6816):46–52, January 2001.

[103] Krishna Kumar Sabapathy, Haoyu Qi, Josh Izaac, and Christian Weedbrook. Production of photonic universal quantum gates enhanced by machine learning. *Phys. Rev. A*, 100:012326, Jul 2019.

[104] Daiqin Su, Casey R. Myers, and Krishna Kumar Sabapathy. Conversion of gaussian states to non-gaussian states using photon-number-resolving detectors. *Phys. Rev. A*, 100:052301, Nov 2019.

[105] Marco Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, et al. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, 2021.

[106] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii. Quantum circuit learning. *Phys. Rev. A*, 98:032309, Sep 2018.

[107] J.C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341, March 1992.

[108] Jarrod R. McClean, Sergio Boixo, Vadim N. Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature Communications*, 9(1), November 2018.

[109] Marco Cerezo, Akira Sone, Tyler Volkoff, Lukasz Cincio, and Patrick J Coles. Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nature communications*, 12(1):1–12, 2021.

[110] Carlos Ortiz Marrero, Mária Kieferová, and Nathan Wiebe. Entanglement induced barren plateaus, 2020.

[111] Samson Wang, Enrico Fontana, M. Cerezo, Kunal Sharma, Akira Sone, Lukasz Cincio, and Patrick J. Coles. Noise-induced barren plateaus in variational quantum algorithms. *Nature Communications 12, 6961 (2021)*, 2020.

[112] Jacob Biamonte. Universal variational quantum computation. *Phys. Rev. A*, 103:L030401, Mar 2021.

[113] Oscar Higgott, Daochen Wang, and Stephen Brierley. Variational Quantum Computation of Excited States. *Quantum*, 3:156, July 2019.

[114] Zhihui Wang, Stuart Hadfield, Zhang Jiang, and Eleanor G. Rieffel. Quantum approximate optimization algorithm for maxcut: A fermionic view. *Phys. Rev. A*, 97:022304, Feb 2018.

[115] Carlos Bravo-Prieto, Ryan LaRose, M. Cerezo, Yigit Subasi, Lukasz Cincio, and Patrick J. Coles. Variational quantum linear solver, 2019.

[116] Hsin-Yuan Huang, Kishor Bharti, and Patrick Rebentrost. Near-term quantum algorithms for linear systems of equations, 2019.

[117] Michael Lubasch, Jaewoo Joo, Pierre Moinier, Martin Kiffner, and Dieter Jaksch. Variational quantum algorithms for nonlinear problems. *Phys. Rev. A*, 101:010301, Jan 2020.

[118] Eric R. Anschuetz, Jonathan P. Olson, Alán Aspuru-Guzik, and Yudong Cao. Variational quantum factoring, 2018.

[119] Maria Schuld, Alex Bocharov, Krysta M. Svore, and Nathan Wiebe. Circuit-centric quantum classifiers. *Phys. Rev. A*, 101:032308, Mar 2020.

[120] Maria Schuld and Nathan Killoran. Quantum machine learning in feature hilbert spaces. *Phys. Rev. Lett.*, 122:040504, Feb 2019.

[121] Jonathan Romero, Jonathan P Olson, and Alan Aspuru-Guzik. Quantum autoencoders for efficient compression of quantum data. *Quantum Science and Technology*, 2(4):045001, aug 2017.

[122] Samuel Yen-Chi Chen, Chao-Han Huck Yang, Jun Qi, Pin-Yu Chen, Xiaoli Ma, and Hsi-Sheng Goan. Variational quantum circuits for deep reinforcement learning, 2019.

[123] Jonathan Romero and Alan Aspuru-Guzik. Variational quantum generators: Generative adversarial quantum machine learning for continuous distributions, 2019.

[124] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Physical Review Letters*, 100(16), April 2008.

[125] Yong Liu, Dongyang Wang, Shichuan Xue, Anqi Huang, Xiang Fu, Xiaogang Qiang, Ping Xu, He-Liang Huang, Mingtang Deng, Chu Guo, Xuejun Yang, and Junjie Wu. Variational quantum circuits for quantum state tomography. *Physical Review A*, 101(5), May 2020.

[126] Ulysse Chabaud, Damian Markham, and Frédéric Grosshans. Stellar representation of non-gaussian quantum states. *Physical Review Letters*, 124(6):063605, 2020.

[127] Jan Provazník, Radim Filip, and Petr Marek. Taming numerical errors in simulations of continuous variable non-gaussian state preparation. *arXiv preprint arXiv:2202.07332*, 2022.

[128] Andreas Björklund, Brajesh Gupt, and Nicolás Quesada. A faster hafnian formula for complex matrices and its benchmarking on a supercomputer. *ACM Journal of Experimental Algorithmics*, 24:1–17, dec 2019.

[129] Jacob FF Bulmer, Bryn A Bell, Rachel S Chadwick, Alex E Jones, Diana Moise, Alessandro Rigazzi, Jan Thorbecke, Utz-Uwe Haus, Thomas Van Vaerenbergh, Raj B Patel, et al. The boundary for quantum advantage in gaussian boson sampling. *Science advances*, 8(4):eabl9236, 2022.

[130] V. V. Dodonov, O. V. Man'ko, and V. I. Man'ko. Multidimensional hermite polynomials and photon distribution for polymode mixed light. *Phys. Rev. A*, 50:813, 1994.

[131] Pieter Kok and Samuel L Braunstein. Multi-dimensional Hermite polynomials in quantum optics. *J. Phys. A: Math. Gen.*, 34(31):6185, 2001.

[132] Regina Kruse, Craig S Hamilton, Linda Sansoni, Sonja Barkhofen, Christine Silberhorn, and Igor Jex. Detailed study of gaussian boson sampling. *Physical Review A*, 100(3):032326, 2019.

[133] Nicolás Quesada. Franck-Condon factors by counting perfect matchings of graphs with loops. *J. Chem. Phys.*, 150(16):164113, 2019.

[134] N Quesada, LG Helt, J Izaac, JM Arrazola, R Shahrokhshahi, CR Myers, and KK Sabapathy. Simulating realistic non-Gaussian state preparation. *Phys. Rev. A*, 100(2):022341, 2019.

[135] Jacob F. F. Bulmer, Stefano Paesani, Rachel S. Chadwick, and Nicolás Quesada. Threshold detection statistics of bosonic states, 2022.

[136] Gianfranco Pierobon, Gianfranco Cariolaro, and Giuseppe Dattoli. On the role of hermite-like polynomials in the fock representations of gaussian states. *Journal of Mathematical Physics*, 62(8):082101, aug 2021.

[137] Man-Duen Choi. Completely positive linear maps on complex matrices. *Linear algebra and its applications*, 10(3):285–290, 1975.

[138] Andrzej Jamiołkowski. Linear transformations which preserve trace and positive semidefiniteness of operators. *Reports on Mathematical Physics*, 3(4):275–278, 1972.

[139] Min Jiang, Shunlong Luo, and Shuangshuang Fu. Channel-state duality. *Phys. Rev. A*, 87:022310, Feb 2013.

[140] J. Eli Bourassa, Nicolás Quesada, Ilan Tzitrin, Antal Száva, Theodor Isacsson, Josh Izaac, Krishna Kumar Sabapathy, Guillaume Dauphinais, and Ish Dhand. Fast simulation of bosonic qubits via gaussian functions in phase space, 2021.

[141] Gabriele Bressanini, Hyukjoon Kwon, and MS Kim. Noise thresholds for classical simulability of non-linear boson sampling. *arXiv preprint arXiv:2202.12052*, 2022.

[142] Girish S Agarwal. *Quantum optics*. Cambridge University Press, 2012.

[143] Gianfranco Cariolaro and Gianfranco Pierobon. Bloch-messiah reduction of gaussian unitaries by takagi factorization. *Physical Review A*, 94(6):062109, 2016.

[144] Gianfranco Cariolaro and Gianfranco Pierobon. Reexamination of bloch-messiah reduction. *Physical Review A*, 93(6):062115, 2016.

[145] William R Clements, Peter C Humphreys, Benjamin J Metcalf, W Steven Kolthammer, and Ian A Walmsley. Optimal design for universal multiport interferometers. *Optica*, 3(12):1460–1465, 2016.

[146] Michael Reck, Anton Zeilinger, Herbert J Bernstein, and Philip Bertani. Experimental realization of any discrete unitary operator. *Physical review letters*, 73(1):58, 1994.

[147] Hubert de Guise, Olivia Di Matteo, and Luis L Sánchez-Soto. Simple factorization of unitary transformations. *Physical review A*, 97(2):022328, 2018.

[148] Juan Miguel Arrazola, Thomas R Bromley, Josh Izaac, Casey R Myers, Kamil Brádler, and Nathan Killoran. Machine learning method for state preparation and gate synthesis on photonic quantum computers. *Quantum Science and Technology*, 4(2):024004, 2019.

[149] Yuan Yao and Filippo M. Miatto. Fast differentiable evolution of quantum states under gaussian transformations, 2021.

[150] J. P. Provost and G. Vallee. Riemannian structure on manifolds of quantum states. *Communications in Mathematical Physics*, 76(3):289–301, September 1980.

[151] F Wilczek and A Shapere. *Geometric Phases in Physics*. WORLD SCIENTIFIC, July 1989.

[152] W. Wirtinger. Zur formalen theorie der funktionen von mehr komplexen veränderlichen. *Mathematische Annalen*, 97:357–376, 1927.

[153] Ken Kreutz-Delgado. The complex gradient operator and the cr-calculus. *arXiv preprint arXiv:0906.4835*, 2009.

[154] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, pages 1–6, 2015.

[155] Brajesh Gupt, Josh Izaac, and Nicolás Quesada. The Walrus: a library for the calculation of hafnians, hermite polynomials and gaussian boson sampling. *Journal of Open Source Software*, 4(44):1705, 2019.

[156] Daniel Gottesman, Alexei Kitaev, and John Preskill. Encoding a qubit in an oscillator. *Phys. Rev. A*, 64:012310, Jun 2001.

[157] Kosuke Fukui, Rafael N. Alexander, and Peter van Loock. All-optical long-distance quantum communication with gottesman-kitaev-preskill qubits. *Phys. Rev. Research*, 3:033118, Aug 2021.

[158] Markus Rau, Tobias Heindel, Sebastian Unsleber, Tristan Braun, Julian Fischer, Stefan Frick, Sebastian Nauerth, Christian Schneider, Gwenaelle Vest, Stephan Reitzenstein, Martin Kamp, Alfred Forchel, Sven Höfling, and Harald Weinfurter. Free space quantum key distribution over 500 meters using electrically driven quantum dot single-photon sources—a proof of principle experiment. *New Journal of Physics*, 16(4):043003, April 2014.

[159] Barry C. Sanders. Quantum dynamics of the nonlinear rotator and the effects of continual spin measurement. *Physical Review A*, 40(5):2417–2427, September 1989.

[160] Hwang Lee, Pieter Kok, and Jonathan P. Dowling. A quantum rosetta stone for interferometry. *Journal of Modern Optics*, 49(14-15):2325–2338, 2002.

[161] Mikolaj Pochylski, Pietro Calandra, Francesco Aliotta, and Rosina C Ponterio. Electrically induced birefringence in nanoparticle dispersions for electrorheological applications. *Journal of Physics D: Applied Physics*, 47(46):465301, October 2014.

[162] The rise of integrated quantum photonics, April 2020.

[163] Jianwei Wang, Fabio Sciarrino, Anthony Laing, and Mark G. Thompson. Integrated photonic quantum technologies. *Nature Photonics*, 14(5):273–284, October 2019.

[164] Ali W. Elshaari, Wolfram Pernice, Kartik Srinivasan, Oliver Benson, and Val Zwiller. Hybrid integrated quantum photonic circuits. *Nature Photonics*, 14(5):285–298, April 2020.

[165] Gilles Brassard, Norbert Lütkenhaus, Tal Mor, and Barry C. Sanders. Limitations on practical quantum cryptography. *Phys. Rev. Lett.*, 85:1330–1333, Aug 2000.

[166] Osian Wolley, Thomas Gregory, Sebastian Beer, Takafumi Higuchi, and Miles Padgett. Quantum imaging with a photon counting camera. *Scientific Reports*, 12(1), may 2022.

[167] Mattias Jönsson and Gunnar Björk. Evaluating the performance of photon-number-resolving detectors. *Physical Review A*, 99(4), apr 2019.

[168] Francesco Marsili, Francesco Bellei, Faraz Najafi, Andrew E. Dane, Eric A. Dauler, Richard J. Molnar, and Karl K. Berggren. Efficient single photon detection from 500 nm to 5 μm wavelength. *Nano Letters*, 12(9):4799–4804, aug 2012.

[169] Ulysse Chabaud and Saeed Mehraban. Holomorphic representation of quantum computations. *Quantum*, 6:831, October 2022.

[170] Pei-Shun Yan, Lan Zhou, Wei Zhong, and Yu-Bo Sheng. A survey on advances of quantum repeater. *EPL (Europhysics Letters)*, 136(1):14001, October 2021.

[171] Filippo M. Miatto, Michael Epping, and Norbert Lutkenhaus. Hamiltonians for one-way quantum repeaters, 2017.

[172] Johannes Borregaard, Hannes Pichler, Tim Schröder, Mikhail D. Lukin, Peter Lodahl, and Anders S. Sørensen. One-way quantum repeater based on near-deterministic photon-emitter interfaces. *Phys. Rev. X*, 10:021071, Jun 2020.

[173] Sreraman Muralidharan, Chang-Ling Zou, Linshu Li, and Liang Jiang. One-way quantum repeaters with quantum reed-solomon codes. *Phys. Rev. A*, 97:052316, May 2018.

[174] Jialin Chen, Lingli Wang, and Bin Wang. Quantum FPGA architecture design. In *2013 International Conference on Field-Programmable Technology (FPT)*. IEEE, dec 2013.

[175] Gary Bécigneul and Octavian-Eugen Ganea. Riemannian adaptive optimization methods, 2018.

[176] Foivos Alimisis, Antonio Orvieto, Gary Becigneul, and Aurelien Lucchi. Momentum improves optimization on riemannian manifolds. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 1351–1359. PMLR, 13–15 Apr 2021.

ECOLE DOCTORALE

Titre : Conception automatisée de circuits quantiques photoniques

Mots clés : optique quantique, apprentissage d'automatique, optimisation, mécanique quantique gaussienne, photonique, circuits quantiques

Résumé : La photonique est une plate-forme de premier plan pour réaliser l'informatique quantique tolérante aux erreurs. Notre objectif principal est d'automatiser la conception de circuits quantiques photoniques et de leurs interconnexions. Avant de fabriquer un véritable ordinateur quantique photonique, il est indispensable de simuler numériquement et d'optimiser les circuits correspondants, qui sont en pratique construits à partir de composants gaussiens. Pour atteindre l'universalité, nous avons également besoin d'effets non gaussiens, qui peuvent être fournis par des détecteurs résolvant le nombre de photons. Nous concevons des circuits à partir de cette boîte à outils et les optimisons pour diverses applications en utilisant des algorithmes de descente de gradient.
Dans cette thèse, nous introduisons une représentation unifiée dans l'espace de Fock de tous les objets gaussiens en termes d'une seule relation de récurrence linéaire qui peut générer de manière récursive leurs amplitudes dans l'espace de Fock, de sorte que l'on peut ensuite inclure les effets non gaussiens dans la simulation. Nous proposons deux méthodes pour calculer le gradient d'un objet gaussien, ce qui nous permet d'adapter davantage différentes optimisations basées sur le gradient au problème d'optimisation de circuit. De plus, nous généralisons une version complexe du gradient naturel pour les circuits quantiques optiques afin d'accélérer la convergence du processus d'apprentissage. Nous donnons également quelques stratégies optimales basées sur les tâches pour utiliser nos relations de récurrence afin de réduire la complexité de calcul, au moins quadratique.
Avec la simulation sur des circuits quantiques photoniques différentiables construits à partir de la relation de récurrence, nous pouvons concevoir automatiquement des circuits quantiques photoniques.
Nous avons rendu ce travail disponible dans diverses bibliothèques open source : TheWalrus, StrawberryFields, Poenta et MrMustard.

Title : Automated design of photonic quantum circuits

Keywords : quantum optics, machine learning, optimization, Gaussian quantum mechanics, photonics, quantum circuits

Abstract : Photonics is a prominent platform for realizing fault-tolerant quantum computing. Our main goal is to automate the design of photonic quantum circuits and of their interconnects. Before a real photonic quantum computer can be manufactured, it is essential to numerically simulate and optimize the corresponding circuits, which in practice are built out of Gaussian components. To achieve universality, we also need non-Gaussian effects, which can be supplied by photon-number-resolving detectors. We design circuits from this toolbox and optimize them for various applications using gradient descent algorithms.
In this thesis, we introduce a unified Fock space representation of all Gaussian objects in terms of a single linear recurrence relation that can recursively generate their Fock space amplitudes, so that one can then include the non-Gaussian effects in the simulation. We propose two methods to calculate the gradient of a Gaussian object, which enables us to further adapt different gradient-based optimizations to the problem of circuit optimization. In addition, we generalize a complex version of the natural gradient for optical quantum circuits to accelerate the convergence of the training process. We also give some optimal task-based strategies for using our recurrence relations to reduce the computation complexity, at least quadratic.
With the simulation on differentiable photonic quantum circuits built from the recurrence relation, we can design photonic quantum circuits automatically.
We made this work available in various open-source libraries: TheWalrus, StrawberryFields, Poenta, and MrMustard.

Institut Polytechnique de Paris
91120 Palaiseau, France