Approximation of continuous-variable quantum computing on NISQ devices

Luca Mondada
Lady Margaret Hall, University of Oxford

Under the supervision of Stefano Gogioso

1st September 2020

A dissertation submitted for the degree of
MSc Mathematics and Foundations of Computer Science

Acknowledgements

My year at Oxford has been fantastic. I am grateful to every member of faculty and my college for their welcoming support throughout – but most importantly, I want to thank my dissertation supervisor, who has done an incredible job at guiding my research and my academic progress, and without whom you would not be reading this thesis. Finally, all of this long and lonesome work would not be possible without the untiring emotional understanding of my partner and family. Thank you!

Contents

Abstract. Classical digital computing creates the illusion of a continuum by hiding the complexity of floating point operations, making (approximate) continuous mathematics available at low cost. Unfortunately, a comparably cheap abstraction for continuous operations is not available in near-term digital quantum computers. For example, a quantum implementation of the truncated Fock basis approximation to continuous-variable (CV) quantum computing – the preferred method for use in classical simulations – would result in non-unitary circuits with extensive long-range connectivity, requiring an exponential number of gates for even the simplest of Gaussian operations.

After reviewing existing approaches and the theory of Gaussian computations, we present our first steps in the development of an alternative framework for the simulation of CV quantum computations on near-term digital devices. In this framework, arbitrary CV states are expressed as superposition of standard Gaussian states, which are in turn mapped onto the computational basis. This results in Gaussian operations which are cheap and easy to implement, with general CV gates obtained through the use of non-Gaussian ancillary states.

Introduction

Quantum mechanics, as a physical theory, is continuous in nature. A quantum system can typically be described by a wave function over its real-valued position space; its evolution is given by differential equations, describing a smooth system transformation over time; and measurements of observables such as position or momentum can be performed, yielding continuous-valued samples drawn from probability distributions. This is in stark contrast to computations on quantum machines, which, albeit based on the same underlying physical theory, are typically designed using discrete physical systems and discrete operations [1].

Many applications from simulation of quantum mechanical systems themselves [2] to quantum machine learning [3, 4] (see below) would benefit from a continuous model of quantum computation. Does the discrete design of quantum computing represent a real limitation of the computing power or can any continuous quantum system be easily discretised and encoded in a qubit machine?

This work's primary objective is to explore the computational differences between the continuous-valued model of quantum mechanics and the discrete framework of quantum computer science. There is both a theoretical and a practical interest in such research: the ability to encode arbitrary continuous operations on qubits could be instrumental in furthering the scope of computational tasks that can be tackled on qubit machines. On the other hand, such advances might give new ways to reason about continuous quantum computations and improve our understanding of some aspects of quantum mechanics that remain difficult to grasp.

This naturally yields a research programme in two stages. The first step will

be an exploration of the theoretical fundament of continuous-variable quantum computing, which from now on, we will refer to as CV computing. We will present the current state of CV computing and build on this discussion to explore discrete representations of continuous computations. This will highlight some of the main distinguishing CV features in comparison to qubit computations, as well as the trade-offs that arise in the discretisation of continuous quantum resources. In the second part, we will then apply the gathered insights to design computational approximations of CV operations on qubit machines. The focus will lie in developing approaches that can be implemented in the near future on Noisy Intermediate-Scale Quantum (NISQ) devices, with the inherent constraints on the number of qubits and the circuit depth. We will also perform proof-of-concept numerical simulations of the different approaches to support our analysis.

In the first chapter, we will motivate our research by introducing some of the challenges of continuous operations on quantum systems and discussing some possible applications of CV computing. A comparison of CV operations in the quantum case to the well-known classical equivalent, floating point arithmetic, will highlight some of the main differences to the classical case that must be addressed. We will then present the theory of quantum CV computation in chapter 2; we will extend the traditional Gaussian formalism to develop a new theory of arbitrary CV computation based on superpositions of Gaussians (SoG). This will lead us to explore different approaches to CV approximations in chapter 3, where comparing traditional approximations with the SoG approach will make clear the advantages and disadvantages of each approach.

1 Continuous computations: an overview

1.1 A brief history of classical continuous computations

Before we start our discussion of CV quantum computation, it is in order to briefly recall the situation in classical computing in order to draw some analogies. Indeed, most features of classical continuous computations are found again in the quantum case, and many of the discretisation challenges of quantum CV are all too familiar with the classical case in mind.

It became clear early on in the development of computers that discrete operations and discrete systems were preferable to continuous systems: not only are discrete systems significantly less sensitive to noise, their operations can also easily be described in exact and succinct form. In binary representation, any operation on a discrete system can be expressed as a truth table: see Figure 1 for an example of this. This representation also yields directly a straightforward way of building electronic circuits from arbitrary operations [5, 6]. This is not to say that intrinsically continuous systems have never been used or cannot be used for computations: FM radio and music discs are good examples of continuous signals, and specific operations such as Fourier transforms can be implemented efficiently in analog signal processing [7]. However, as the cost of technology fell and processing power increased, the few use case-specific advantages of continuous operations have for the vast majority been overtaken by the simplicity, scalability and reliability of digital technology.

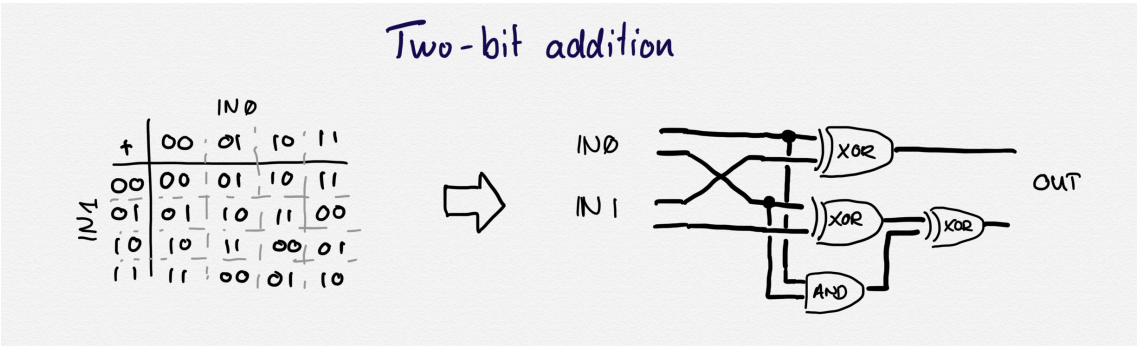


Figure 1: In classical computing, any operation can be viewed as a binary truth table (on the left). From there, the optimal electrical circuit can be computed (on the right).

Instead, we rely on approximations of continuous operations that run on discrete hardware with arbitrary precision, given enough resources. The most fundamental example of this paradigm is the floating point number representation, which has become the standard abstraction for arbitrary continuous-valued real numbers. Processors support all standard arithmetic operations on floating points approximations, so that the differences behind discrete integers and the real con-

tinuum become invisible [8]. Behind the scenes, any non-integer value is stored as two values, similar to the familiar scientific notation:

$$5.673 = 5673 \cdot 10^{-3}.$$

The first value stores the most significant bits of the non-integer value (called the mantissa — in our case 5673), whilst the second value stores the exponent (in our case -3). This allows to represent arbitrary numbers up to a certain precision, at the cost of more complex operations: additions of floating point numbers, for example, now require first that the exponents be compared and in the case that they do not match, that one of the summands be zero-padded until the exponents match.

This approximation of continuous values has been widely successful and is nowadays found in every processor. Nevertheless, this does not mean that all quirks of classical continuous computation have been resolved: beyond mere approximation errors, it still remains to the programmer to take care of some of the unexpected behaviours of floating point numbers. For example, for possibly negative numbers, the sign of the number must be stored separately from the mantissa and exponent; a consequence of that is that $+0$ and -0 are recognised as two different values to a processor. This goes to show that discretisation of continuous operations is often not as straightforward as might be assumed, and that even as an approximation might be successful, it might not be that every operation is simple to perform.

We will see in the quantum case that many of these considerations remain valid. As opposed to the classical case, however, it is yet unclear how to best discretise quantum CV operations on NISQ devices. Indeed, beyond encoding continuous values in discrete systems, discretisations of quantum CV computations must be able encode arbitrary continuous superposition of states. We discuss this in the next section.

1.2 The challenges of quantum CV state representation

In the previous section, we introduced some ideas for discretisation of continuous operations by examining the current consensus in the more familiar case of classical computing. We will introduce the quantum CV framework more formally in the next chapter, but before we do so, we wish here to highlight briefly some of the main challenges that distinguishes quantum CV from the classical case.

First and foremost, the quantum Hilbert structure of state space gives us the freedom to choose an appropriate basis for the discretisation: as opposed to the classical case, where we know that we need to transform a real number $v \in \mathbb{R}$ into a sequence of bits $x_1 \dots x_n$, we can choose to express a given state $|\psi\rangle$ as a

superposition $|\psi\rangle = \sum_{i=0}^{m} \alpha_i |\phi_i\rangle$, and then encode each of $|\phi_i\rangle$ as $\left|x_1^{(i)} \dots x_n^{(i)}\right\rangle$:

$$|\psi\rangle = \sum_{i=0}^{m} \alpha_i |\phi_i\rangle = \sum_{i=0}^{m} \alpha_i \left|x_1^{(i)} \dots x_n^{(i)}\right\rangle .$$

This is a liberty that of course we should exploit to find an optimal discretised representation.

On the downside, however, given any basis of the Hilbert space, the infinite dimensional construction also means that there will always be CV states that cannot be expressed as the superposition of a finite number of terms. For example, suppose we express states as superpositions of eigenstate of position $|x\rangle$, $x \in \mathbb{R}$. Then arbitrary two-mode states are of the form

$$|\psi\rangle = \int_{-\infty}^{\infty} \mathrm{d}x\mathrm{d}y \; \psi(x,y) \, |x\rangle \otimes |y\rangle,$$

where $\psi(x,y)$ denotes the normalised wave function of $|\psi\rangle$:

$$\psi(x,y) = \langle\psi|xy\rangle = \langle\psi| \left(|x\rangle \otimes |y\rangle\right) .$$

It should be clear why such states cannot be expressed in general as finite superpositions of $|x\rangle$ states. A particularly interesting example of this is given by the idealised wave function $\psi(x,y) = \delta(x^3 - y)f(x)$, where f is some real function $f : \mathbb{R} \to \mathbb{R}$ with norm one. The choice of the term x^3 is irrelevant, as long as it is non-linear in x. The state $|\psi\rangle$ simplifies to

$$|\psi\rangle = \int_{-\infty}^{\infty} \mathrm{d}x \, f(x) \, |x\rangle \otimes |x^3\rangle.$$

We can see that there is a perfect correlation between the first and second mode – but this correlation is non-linear. Such continuous superpositions and the resulting non-linear entanglement have no discrete equivalent (or classical equivalent, for that matter). As such, it is impossible to encode this exactly in some discretised $|x\rangle$ basis.

Finally, given the current limitations of quantum hardware [9], an important criterion in the search for a discrete approximation of continuous operations is the locality of the resulting simulated operations. That is, we wish that "simple" continuous operations[1] result in approximated operations that involve as few qubits as possible, as an indication of feasibility in real-world application in the near future. Note that the classical implementation of addition is highly non-local: the simple operation +1 will flip every bit of eg. `01111`. At this point in the development of quantum computer science, this behaviour is highly undesirable and would make any approximation approach impractical to implement and impossible to scale.

[1] We will elaborate on this in the next chapter.

1.3 Continuity in Quantum Machine Learning

In practice, the CV model of quantum computation has many benefits in diverse applications [2–4]. In the absence of CV devices and for the same reliability and scalability reasons we discussed in section 1.1, obtaining an efficient CV approximation scheme on qubit devices would be a significant milestone. We will complete this first introductory chapter by presenting a concrete application and the advantages of the CV model.

In recent years, efforts towards Quantum Machine Learning (QML) have grown to significant prominence. The main realisation driving developments in the field is that quantum mechanics and the theory behind machine learning (ML) have some aspects in common. In fact, both theories rely heavily on linear algebra, so that by translating a linear algebra computation from machine learning into a physical quantum mechanical process, we can hope to perform machine learning optimisations faster than we could on classical computers. Furthermore, the operator formalism of quantum mechanics also makes it easy to compute derivatives in certain circumstances, so that we can hope to leverage that too to evaluate optimisation gradients [10].

Several approaches have been followed to encode ML feature vectors into qubits. The most straightforward attempt is to interpret a feature vector directly as an element of the Hilbert space of quantum states [11–13]. The feature vector thus defines a quantum state and the vector components are stored as amplitudes in the quantum state. Amplitude-encoding makes linear operations on feature vectors very easy to be performed. However, machine learning also crucially depends on non-linear operations in order to represent non-linear models. Introducing non-linearity in amplitude-encoded feature vectors is physically impossible without state measurements and repeated state preparations, rendering it likely that the additional computational overhead will more than compensate the speed-up from quantum computations.

Alternative qubit encodings exist [4, 14, 15], but in the absence of amplitude-encoding, they all suffer from a common problem: they are encoding continuous information in a discretised system. On the other hand, the Gaussian representation of CV states, which we will present in the second chapter, offers an elegant framework to store feature vectors as mean position and momentum of CV systems [3, 16]. It turns out that using unitary transformations of the CV states, we can perform both linear and non-linear operations on the feature vectors.

2 The CV model of quantum computation

2.1 The quantum harmonic oscillator (QHO) model

The quantum CV computational model is typically based on the quantum harmonic oscillator (QHO) framework. This corresponds to the simplest (non-trivial) one-dimensional CV system, in which the dynamics are given by a quadratic potential, i.e. by a first-order approximation of the force acting on the system.

QHO admits a very elegant algebraic theory that gives a complete characterisation of its energy levels and its respective eigenstates. In the following, we will briefly recall the main properties of QHO but we will not go into detail and assume that the reader already has some level of familiarity with the formalism. For a detailed overview, we refer to a reference such as [17].

The Hamiltonian, the eigenstates and the position and momentum observables of the QHO can all be obtained from the definition of an annihilation operator $\hat{a}$, and its conjugate $\hat{a}^\dagger$, the creation operator:

$$\hat{x} = \frac{1}{\sqrt{2}}(\hat{a}^\dagger + \hat{a}) \qquad \hat{p} = \frac{i}{\sqrt{2}}(\hat{a}^\dagger - \hat{a}) \qquad H = \hbar\omega(\hat{a}^\dagger\hat{a} + \frac{1}{2})$$
$$\hat{a}^\dagger|n\rangle = \sqrt{n+1}\,|n+1\rangle \qquad \hat{a}|n\rangle = \sqrt{n}\,|n-1\rangle \qquad \hat{a}|0\rangle = 0,$$

where $|n\rangle, n = 0, 1, 2\ldots$ denotes the energy eigenstates of H with energy levels $H_n = \hbar\omega(n+1)$. An important property of QHO is the commutation relation $[\hat{a}, \hat{a}^\dagger] = 1$, which can equivalently be expressed as $[\hat{x}, \hat{p}] = i$.

Such a QHO system is typically called a mode, or qumode. CV computations become meaningful once several such qumodes are considered together and are entangled. In this case, it is common to define for n qumodes

$$\hat{\mathbf{R}} = \begin{bmatrix}\hat{q}_1 & \hat{p}_1 & \ldots & \hat{q}_n & \hat{p}_n\end{bmatrix}^\top.$$

The q, p commutation relations can then be rewritten succinctly as

$$[\hat{R}_k, \hat{R}_l] = i\Omega_{kl},$$

where Ω_{kl} is the symplectic matrix

$$\mathbf{\Omega} = \bigotimes_{k=1}^{n} \boldsymbol{\omega}, \quad \boldsymbol{\omega} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}.$$

A quantum electromagnetic field corresponds pretty closely to such a system. The number of qumodes in the electromagnetic field is determined by the specific boundary conditions of the system considered, similar to how the geometry of a water glass or a bell determines the resonance frequencies. We typically imagine a CV quantum computer to have a finite and fixed number of modes k, and think of them as the CV equivalent of qubits on a digital machine.

2.2 Complete CV state descriptions

The QHO energy eigenstates $|n\rangle$ introduced in the previous section form a orthonormal basis of the Hilbert state space. This countably infinite-dimensional Hilbert space is also known as Fock space, and the energy eigenstates $|n\rangle$ as Fock states. Hence, a complete description of a CV state with k modes is given by vectors from the tensor product of k Fock spaces $\mathcal{F}_1, \ldots, \mathcal{F}_k$:

$$|\psi\rangle \in \bigotimes_{i=1}^{k} \mathcal{F}_i.$$

In Fock basis

Using orthonormality, we immediately obtain the description of any pure state $|\psi\rangle$ in the Fock basis

$$|\psi\rangle = \sum_{n=0}^{\infty} \langle n|\psi\rangle \, |n\rangle \, . \qquad [1]$$

In the rest of this work when the basis is not specified, the state vector of a state $|\psi\rangle$ refers to the vector given by the coefficients $\langle n|\psi\rangle$, i.e. its state vector in the Fock basis.

Using the standard generalisation of quantum mixed states ρ as density matrices, as is wildly used in qubit quantum information, we obtain that a CV state is fully described by an infinite square matrix A

$$\rho = \sum_{n_1, n_1', \ldots, n_k, n_k'=0}^{\infty} a_{(n_1,n_1'),\ldots,(n_k,n_k')} \, |n_1 \ldots n_k\rangle\langle n_1' \ldots n_k'|$$

$$A = \begin{bmatrix} a_{(1,1)\ldots(1,1)} & a_{(1,1)\ldots(1,2)} & \cdots & a_{(1,1)\ldots(1,k)} & a_{(1,1)\ldots(1,2),(1,1)} & \cdots \\ a_{(1,1)\ldots(2,1)} & a_{(1,1)\ldots(2,2)} & \cdots & a_{(1,1)\ldots(2,k)} & a_{(1,1)\ldots(1,2),(2,1)} & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \\ a_{(1,1)\ldots(k,1)} & a_{(1,1)\ldots(k,2)} & \cdots & a_{(1,1)\ldots(k,k)} & a_{(1,1)\ldots(1,2)(k,1)} & \cdots \\ a_{(1,1)\ldots(2,1)(1,1)} & a_{(1,1)\ldots(2,1)(1,2)} & \cdots & a_{(1,1)\ldots(2,1)(1,k)} & a_{(1,1)\ldots(2,2)(1,1)} & \cdots \\ \vdots & \vdots & & \vdots & \vdots & \ddots \end{bmatrix}$$

Note that this matrix must be symmetric positive definite and have trace smaller than unity: $0 \leqslant \mathrm{Tr}A \leqslant 1$.

As a wave function

Sometimes, working with infinite dimensional matrices is not the most convenient approach. Another representation that is usual in quantum mechanics is the wave function formalism. This is equivalent to replacing the Fock basis of equation 1 by the uncountable set of orthogonal position eigenstates $|x\rangle$:

$$|\psi\rangle = \int \langle x|\psi\rangle \, |x\rangle \qquad [2]$$

Plugging equations 1 and 2 together, we get an expression of the wave function in terms of the state vector in Fock basis:

$$\begin{aligned}\psi(x) = \langle x|\psi\rangle &= \sum_{n=0}^{\infty} \langle x|n\rangle \langle n|\psi\rangle \\ &= \sum_{n=0}^{\infty} \langle n|\psi\rangle \mathcal{N}_n \cdot H_n(x) e^{-x^2/2},\end{aligned}$$

where we used the well-known expression of the Fock state wavefunction in terms of the n-th Hermite polynomials $H_n(x)$:

$$\langle x|n\rangle = \mathcal{N}_n \cdot H_n(x) e^{-x^2/2}, \qquad [3]$$

where $\mathcal{N}_n$ is some normalisation factor.

Wavefunctions are a helpful representation insofar as they are arguably easier to reason about than infinite matrices and have a clear physical interpretation. Computationally, however, they remain as hard to manipulate as the infinite matrix representation, as in general in the absence of an elegant approximation, they ultimately rely on a similar representation with infinitely many coefficient.

However, an important pattern emerged in equations 1 and 2: by choosing a set of vectors from Hilbert space, we can obtain new descriptions of a given state, with possibly different properties: the Fock basis can be used to express arbitrary states as a discrete superposition of countably many basis elements, whereas the wave function description is uncountable and continuous. However, wave function representations have advantages too, as they are better suited to model position measurements and compute outcomes.

As Wigner function

Finally, a third state description close to the wave function representation for a k-mode mixed state ρ is given by its Wigner function $W_\rho(\boldsymbol{\xi})$ [18]. In contrast to the wave function that is only a function of one observable (e.g. position or momentum), the Wigner function is defined over the entire phase space given by position and momentum simultaneously: $\boldsymbol{\xi} = (x_1, p_1, \dots, x_k, p_k)^\top$.

It is given in terms of the position eigenvectors $|\mathbf{x}\rangle$ by

$$W(\boldsymbol{\xi}) = W(\mathbf{x}, \mathbf{p}) = \frac{1}{\pi^k} \int_{\mathbb{R}^k} \mathrm{d}^k\mathbf{q} \ \langle \mathbf{x} + \mathbf{q}| \rho |\mathbf{x} - \mathbf{q}\rangle \, e^{2i\mathbf{q}\cdot\mathbf{p}}.$$

Given the Heisenberg uncertainty principle, this function cannot be a probability distribution of position and momentum simultaneously. However, the Wigner function is a so-called *quasi-probability distribution*: it is real-valued and normalised, and has the further property that its marginal distribution are probability distributions:

$$\langle x_k| \rho |x_k\rangle = \int_{\mathbb{R}^{2k-1}} \mathrm{d}p_1 \dots \mathrm{d}p_k \mathrm{d}x_1 \dots \mathrm{d}x_{k-1} \ W_\rho(\mathbf{x}, \mathbf{p}).$$

The Wigner function offers an alternative to the wave function formalism that is symmetric in the quadrature operators $\hat{\mathbf{x}}$ and $\hat{\mathbf{p}}$ and, crucially, that generalises well to non-pure multi-mode states. We will see in the coming sections that Wigner functions form the basis for the description of arbitrary Gaussian states.

2.3 Gaussian states

The complete CV state descriptions of the previous section will be impractical for discrete approximations, as they are infinite and/or continuous. The most naive solution would be to attempt to discretise the function domain or select a finite subset of the basis sets as an approximation. Several drawbacks, which we will discuss in more detail section 3.1, render this approach difficult in practice. Instead, we will now present a well-behaved strict subset of CV states that form a promising avenue of work for CV state approximation schemes: these states can be completely defined in finite dimension, making them ideally suited for digital simulations.

These states are known as Gaussian states: in the pure case they correspond to states with Gaussian wave functions. A pure single-mode Gaussian state $|\Gamma\rangle$ is expressed by the wave function

$$\langle x|\Gamma\rangle = \left(\frac{\pi}{\mathrm{Re}\gamma}\right)^{\frac{1}{4}} e^{ip_0 x} e^{-\frac{1}{2}\gamma(x-x_0)^2}, \qquad [4]$$

parametrised by two real parameters $p_0, x_0 \in \mathbb{R}$ and the complex parameter $\gamma \in \mathbb{C}^+ = \{z \in \mathbb{C} : \mathrm{Re}\, z > 0\}$ [19]. This can be generalised to arbitrary multi-mode mixed states ρ using their Wigner function. A k-mode mixed state ρ is Gaussian if it can be expressed as

$$W_\rho(\boldsymbol{\xi}) = \frac{1}{\pi^k}\frac{1}{\sqrt{\det(\boldsymbol{\sigma})}}\exp\Big(-(\boldsymbol{\xi}-\mathbf{d})^\top \boldsymbol{\sigma}^{-1}(\boldsymbol{\xi}-\mathbf{d})\Big),$$

parametrised by the displacement vector $\mathbf{d} \in \mathbb{R}^{2k}$ and the symmetric positive covariance matrix $\boldsymbol{\sigma} \in \mathbb{R}^{2k\times 2k}$. Given that $\boldsymbol{\sigma}$ is symmetric, any k-mode Gaussian state can thus be parametrised by $2k^2+3k$ real values. This is in stark contrast with any representation based on finite approximation of state vectors, where the number of basis elements grows exponentially in the number of modes k: if a single mode is represented using n basis elements, then describing k modes require n^k parameters.

As their names suggest, the displacement and covariance parameters of Gaussian states also have a straightforward physical interpretation. They correspond to the first and second (symmetric) statistical moments of the $\hat{\mathbf{x}}$ and $\hat{\mathbf{p}}$ observ-

ables [18, 20]:

$$\begin{aligned} \mathbf{d} &= \left(\mathbb{E}_\rho \hat{x}_1, \mathbb{E}_\rho \hat{p}_1, \dots, \mathbb{E}_\rho \hat{x}_k, \mathbb{E}_\rho \hat{p}_k\right)^\top && \text{(first order moments)} \\ \sigma_{ij} &= \frac{1}{2}\mathbb{E}_\rho[\hat{R}_i\hat{R}_j + \hat{R}_j\hat{R}_i] - \mathbb{E}_\rho[\hat{R}_i]\mathbb{E}_\rho[\hat{R}_j] && \text{(second order moments)} \end{aligned} \qquad [5]$$

Using equation 5, we can even imagine to obtain Gaussian approximations of any CV state ρ by truncating the Wigner function to first and second moments. In many cases, this is of course a very crude approximation, especially as higher energy components of the wave function begin to induce high frequency terms. We will expand on this idea in more detail in section 2.5 and examine strategies to approximate arbitrary states.

The Gaussian formalism is widely used in practice, both by theorists and experimentalists. Its popularity stems from the simplicity and ubiquity of Gaussian states: most well-known states, as well as the vast majority of experimentally feasible states happen to be Gaussian already. Perhaps even more significantly, most experimentally feasible transformations preserve Gaussians, making Gaussian computations central to CV computation theory. These Gaussian-preserving operations will be the subject of the next section.

Gaussian transformations

Operations that preserve Gaussian states are called Gaussian transformations and correspond to affine transformations of phase space [21]

$$\boldsymbol{\xi} \mapsto \mathbf{S}\boldsymbol{\xi} + \boldsymbol{\Delta}.$$

For the covariance matrix σ, this corresponds to the transformation

$$\sigma \mapsto \mathbf{S}\sigma\mathbf{S}^\top. \qquad [6]$$

The preservation of the standard commutation relations $[\hat{R}_i, \hat{R}_j] = i\,\boldsymbol{\Omega}_{ij}$ requires that $\mathbf{S}$ be symplectic, i.e. $\mathbf{S}\boldsymbol{\Omega}\mathbf{S}^\top = \boldsymbol{\Omega}$. Quantum operations on Gaussian CV states form the symplectic group $\mathrm{Sp}(2k, \mathbb{R})$ and are fully described by linear operations on the phase space.

The most common Gaussian operations are displacements, rotations and squeezings. These operations take their names from their effect in phase space: they can be elegantly visualised in phase space by plotting their effect on the Wigner function of CV states, see figure 2. Displacements provoke a global translation in phase space, leaving the shape of the Wigner function (i.e. the covariance) unchanged. Squeezing operations scale the two axes of the Gaussian by some factor s and s^{-1} respectively, yielding ellipse-shaped Gaussians. Finally, rotations act as expected, by rotating phase space.

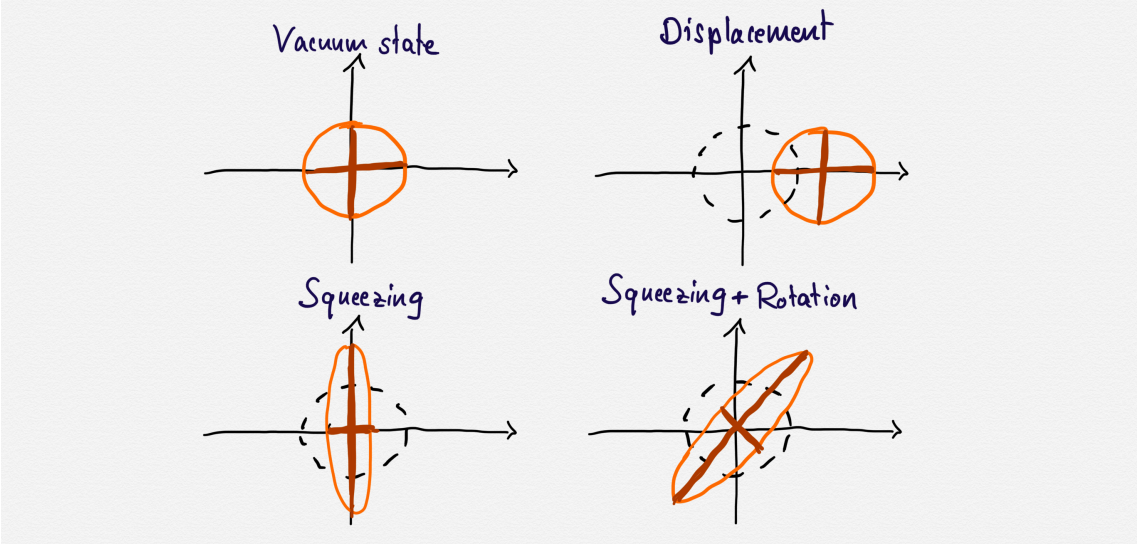


Figure 2: Wigner functions for Gaussian transformations acting on the vacuum state ($\mathbf{d} = \mathbf{0}, \sigma = \mathbb{1}_2$).

There are also Gaussian operations acting on multiple modes: the most used two-mode transformation is the beamsplitter, which is easiest understood as a rotation in the combined four dimensional phase space of the 2 modes. Any single-mode Gaussian operation can be obtained by a combination of displacements, squeezing operations and rotations, whilst any multi-mode Gaussian operations are generated by beamsplitters together with arbitrary single-mode Gaussian operations.

These transformations can also be related to specific Hamiltonians, expressed using the $\hat{a}^\dagger, \hat{a}$ operator formalism: affine displacements are generated by the terms linear in $\hat{a}^\dagger$ or $\hat{a}$, while the symplectic transformation $\mathbf{S}$ are precisely those generated by quadratic Hamiltonians

$$\hat{H} = \hat{\boldsymbol{\xi}}^\dagger \mathbf{H} \hat{\boldsymbol{\xi}}$$

for some Hermitian matrix $\mathbf{H} \in \mathbb{C}^{2k \times 2k}$. Hamiltonians at most quadratic in the quadrature operators generate operations that we are able to implement in experimental setups; higher order terms require high energies to be performed and remain a challenge in practice up to this day. However, Gaussian CV operations are not universal for quantum computing, and as such non-Gaussian transformations are a key resource if we wish to perform arbitrary quantum operations [22, 23].

Non-Gaussian transformations

From the Wigner visualisation of CV transformations in figure 3, it is clear that transformations that are not affine in phase space do not preserve Gaussianity. In

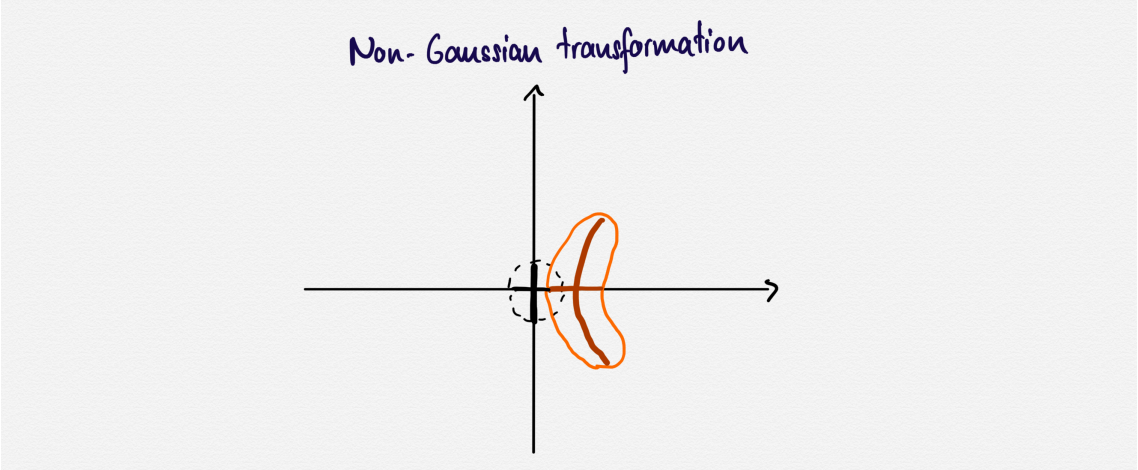


Figure 3: Non-linear transformations of phase space do not preserve Gaussian states.

contrast to Gaussian operations, that form a well-understood theory, no framework has been established that properly characterises general CV states and operations; studying entanglement properties of arbitrary CV systems for example remains a very active area of research among others [21, 24, 25]. Implementation of non-Gaussian gates in practice has proven challenging too and remains the main challenge in the development of CV quantum computers [26].

Notably, however, it has long been known that any source of non-linearity is universal; that is, the Gaussian CV framework, together with any one non-Gaussian gate is enough to arbitrarily approximate any CV operation [22, 23]. Given this, several approaches have been developed to achieve universal CV computation.

Two of the most popular non-Gaussian gates are the Kerr gate $K(\kappa)$ and the cubic phase gate $V(\gamma)$

$$V(\gamma) = e^{i(\gamma/3\hbar)\hat{x}^3} \qquad \text{ie} \qquad V(\gamma)^\dagger \begin{bmatrix} \hat{x} \\ \hat{p} \end{bmatrix} V(\gamma) = \begin{bmatrix} \hat{x} \\ \hat{p} + \gamma\hat{x}^2 \end{bmatrix} \qquad [7]$$

$$K(\kappa) = e^{i\kappa\hat{n}^2} = e^{i\kappa(\hat{a}^\dagger\hat{a})^2} \qquad \text{ie} \qquad K(\kappa)^\dagger \hat{a} K(\kappa) = \text{I WILL SOLVE THIS} \qquad [8]$$

The cubic phase gate is arguably the simplest cubic Hamiltonian that can be devised and has a very nice interpretation in phase space as a parabola. On the other hand, the Kerr interaction is popular in numerical Fock basis simulations because it is diagonal and, as such, is trivial to compute. In experimental setups, other gates such as the photon-number addition and substraction gates (PNA/PNS) are also used; see e.g. [26] for a review.

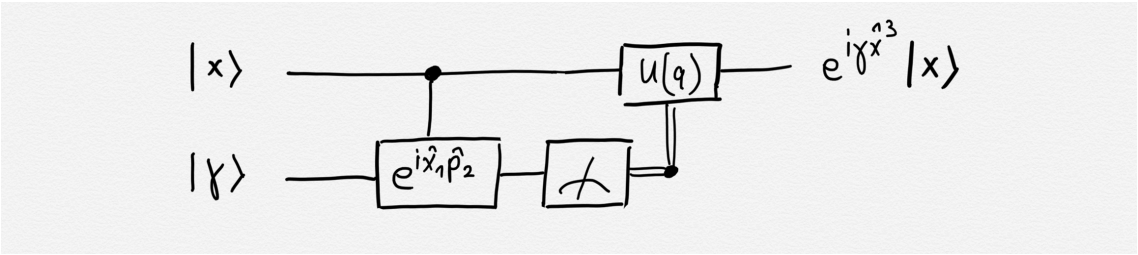
As another resource of non-Gaussianity, one can consider Gaussian operations followed by tracing on parts of the system (the ancilla modes), similarly to how non-unitary CPTP maps can be obtained on qubits by tracing out ancilla qubits. Crucially, however, the ancilla modes have to be prepared in a non-

Gaussian state. In this way, universal CV computing can still be achieved by replacing the non-Gaussian resource from a non-Gaussian operation to a non-Gaussian state that can be prepared ahead of time and independently of the rest of the computation.

Consider for example the cubic phase gate defined in equation 7. Ghose and Sanders have shown that this can be performed using an ancilla mode prepared in the so-called cubic phase state $|\gamma\rangle$ [27]

$$\langle x|\gamma\rangle = e^{i\gamma x^3}. \tag{9}$$

Indeed, for any position eigenstate $|x\rangle$, the following circuit



where $e^{i\hat{x}_1\hat{p}_2}$ is a Gaussian transformation – also known as the inverse of the CX gate – and the measurement designates a x-basis measurement with outcome q. This yields the output $e^{i\gamma x^3}|x\rangle$, thus being equivalent to the cubic phase gate.

2.4 Gaussian Measurements

Finally, we will look at measurements on Gaussian states. It turns out that measurements of e.g. x- or p-quadrature preserves Gaussians. Measurements of quadrature operators can be achieved experimentally through homodyne detection [18]. In its most general form, a Gaussian measurement on a subsystem B with modes $B_1, \dots, B_{N_B}$ is given by the POVM

$$\hat{\Pi}_B(\boldsymbol{\eta}) = \pi^{-N_B} \left(\prod_{j=1}^{N_B} \hat{D}_{B_j}(\eta_j) \right) \boldsymbol{\Lambda}_B^{\hat{\Pi}} \left(\prod_{j=1}^{N_B} \hat{D}_{B_j}^{\dagger}(\eta_j) \right) \tag{10}$$

where

$$\hat{D}_B(\eta_j) = \exp\left(\eta_j \hat{b}_j^{\dagger} - \overline{\eta}_j \hat{b}_j \right)$$

is the displacement operator. The result of the POVM is given by $\boldsymbol{\eta}$, which indexes the set of projections of equation 10, whilst $\boldsymbol{\Lambda}_B^{\hat{\Pi}}$ is the density matrix of some Gaussian state characterising the measurement. For equation 10 to be a POVM, it must additionally hold that $\pi^{-N_B} \int \mathrm{d}^{2N_B}\boldsymbol{\eta}\, \hat{\Pi}_B(\boldsymbol{\eta}) = 1$.

For an ideal x-homodyne measurement, for instance, the measurement is given by

$$\boldsymbol{\Lambda}_B^{\hat{\Pi}} = |0\rangle\langle 0|,$$

where $|0\rangle\langle 0|$ refers to the 0-position eigenstate (infinitely x-squeezed Gaussian). Another well-known Gaussian measurement, the heterodyne measurement, is given by $\mathbf{\Lambda}_B^{\hat{\Pi}} = |\nu\rangle\langle\nu|$, the Gaussian vacuum state (zero-th energy state).

Now, assume that the Gaussian state $\mathbf{\Lambda}_B^{\hat{\Pi}}$ is given by some covariance matrix $\mathbf{\Gamma}_B^{\hat{\Pi}}$ and assume the measured CV state is composed of subsystems A and B. For the Gaussian state AB, split its covariance matrix σ_{AB} into terms belonging to subsystems A and B:

$$\sigma_{AB} = \begin{bmatrix} \sigma_A & \epsilon_{AB} \\ \epsilon_{AB}^\top & \sigma_B \end{bmatrix}.$$

Then, after the measurement $\mathbf{\Gamma}_B^{\hat{\Pi}}$ on B, the state of subsystem A is Gaussian and can be expressed as [18]

$$\sigma_A^{\hat{\Pi}} = \sigma_A - \epsilon_{AB}(\sigma_B + \mathbf{\Gamma}_B^{\hat{\Pi}})^{-1}\epsilon_{AB}^\top. \tag{11}$$

In practice, it suffices to be able to perform a single Gaussian measurement, as any other can be obtained through Gaussian transformations [18]. Thus, equation 11 gives us a ready formula for simulating arbitrary measurements.

There are also non-Gaussian measurements such as photon counting. The resulting states from such measurements cannot be expressed in the Gaussian formalism. However, closed analytical expressions exist in some cases; photon counting outcomes from Gaussian states for example can be obtained through Hafnians [28]. We will not discuss this further and will focus on Gaussian measurements for the rest of this work.

Putting these observations together, we can conclude that to simulate arbitrary CV computations, we need to be able to

(i) represent and store arbitrary states,

(ii) simulate Gaussian transformations on arbitrary states,

(iii) prepare one non-Gaussian resource state (such as the cubic phase state),

(iv) perform homodyne measurements on arbitrary states.

This will be the topic of the next chapter. The remainder of this chapter will be spent on studying the power of superposition of Gaussians to approximate arbitrary states.

2.5 Arbitrary states as superposition of Gaussians

The Gaussian approximation is a very convenient and successful description of simple CV states and operations. It makes excellent predictions for most currently feasible experiments and low energy states. However, we saw that Gaussian-preserving operations do not form a universal set for CV computation – in fact, all Gaussian operations can be efficiently simulated on classical computers, using the finite-dimensional description given in equation 5, so that the ability of simulating Gaussian operations only is in effect useless.

In the coming sections and for much of this dissertation, we will study how to expand the Gaussian formalism beyond the limited scope of Gaussian CV computing, with the aim of leveraging its elegant description that can be easily simulated in a broader context. To that end, we propose to express an arbitrary pure CV state $|\psi\rangle$ as superpositions of Gaussians $|\Gamma_1\rangle, \dots, |\Gamma_M\rangle$:

$$|\psi\rangle = \sum_{i=1}^{M} \alpha_i |\Gamma_i\rangle, \quad \text{for some } \alpha_1, \dots, \alpha_M \in \mathbb{C}.$$

A Gaussian operation $U_{\mathcal{G}}$ can then easily be computed for arbitrary states by applying them separately on each Gaussian of the superposition:

$$U_{\mathcal{G}} |\psi\rangle = \sum_{i=1}^{M} \alpha_i U_{\mathcal{G}} |\Gamma_i\rangle.$$

Such a superposition of Gaussians (SoG) expression can be obtained for any CV state $|\psi\rangle$. This follows directly from a straightforward generalisation of the Wigner-Wiehl transform [29].

In practice, we are interested in knowing how well the SoG approximation works with a small number of Gaussians. We thus now proceed to perform some numerical experiments for SoG approximations of two of the most common non-Gaussian states: Fock states and cubic phase states.

Fock states

A good first proof of concept is to try to express the energy eigenstates of QHO as superpositions of Gaussians. Expressing energy eigenstates as Gaussian superpositions is of interest, as it automatically generalises to arbitrary CV states, given that the eigenstates span the Fock space.

To find the best SoG approximation numerically, we will use least squares optimisation with trust regions [30], as implemented in `scipy.optimize` by the routine `optimize.least_squares(method='dogbox')` [31]. Given that Fock wave functions have no imaginary part (i.e. the amplitudes are real), we can simplify our SoG ansatz to the following:

$$\psi_{|n\rangle}(x) \approx \tilde{\psi}_{|n\rangle}(x) = \sum_{i=1}^{M} a_i \exp\Big(-\frac{1}{2}\gamma_i (x - \mu_i)^2 \Big). \qquad [12]$$

The number of Gaussians M is fixed and we optimise the real parameters

$$a_i, \mu_i, \gamma_i \in \mathbb{R} \quad \text{for } i = 1, \dots, M$$

The least squares reminder is given by

$$R^2 = \int_I \Big(\psi_{|n\rangle}(x) - \tilde{\psi}_{|n\rangle}(x) \Big)^2 \mathrm{d}x \approx \sum_{j=1}^{n_s} \Big(\psi_{|n\rangle}(x_j) - \tilde{\psi}_{|n\rangle}(x_j) \Big)^2 \cdot \Delta x_j,$$

where $x_1, \dots, x_{n_s}$ is some sampling of the interval I being integrated over, and Δx_j denotes the size of the interval around x_j.

Let us make a few observations about the Fock states in order to fix a value for the number of superpositions M and choose an educated initial guess for the optimisation parameters. Recalling equation 3, we can find the roots and local extrema of the Fock states analytically:

$$\begin{aligned} \psi_{|n\rangle}(x) &= \mathcal{N}_n \cdot H_n(x) e^{-x^2} \stackrel{!}{=} 0 \\ &\Leftrightarrow H_n(x) = 0 \qquad [13] \\ \frac{\mathrm{d}}{\mathrm{d}x}\psi_{|n\rangle}(x) &= \mathcal{N}_n \cdot (-2xe^{-x^2}) H_n(x) + \mathcal{N}_n \cdot e^{-x^2} \frac{\mathrm{d}}{\mathrm{d}x} H_n(x) \stackrel{!}{=} 0 \\ &\Leftrightarrow (-2xe^{-x^2}) H_n(x) + e^{-x^2}(2xH_n(x) - H_{n+1}(x)) = 0 \\ &\Leftrightarrow H_{n+1}(x) = 0 \qquad [14] \end{aligned}$$

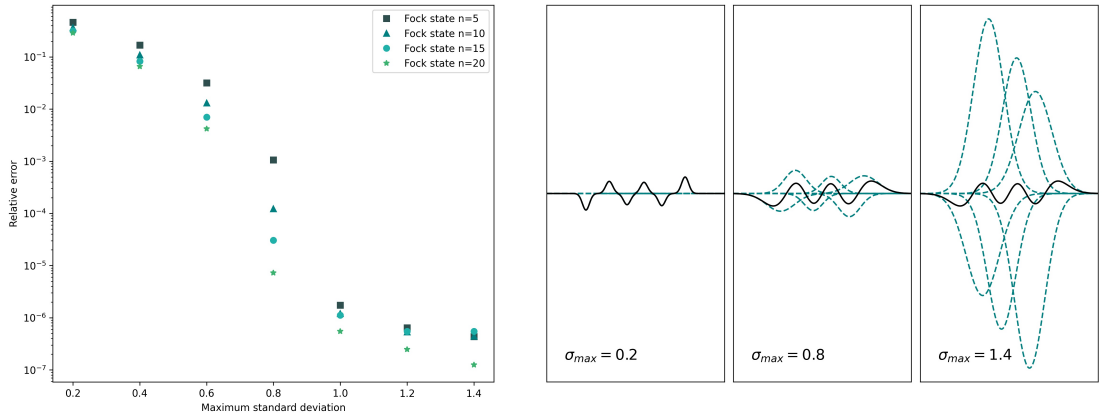
That is, the roots of $|n\rangle$ correspond to the roots of the n-th Hermite polynomial and the local extrema correspond to the roots of the $(n+1)$-th Hermite polynomial. Since H_n is a polynomial of order n, we deduce that the n-th energy state has n roots and $n+1$ local extrema. The case $n = 5$ can be seen in Figure 4b (black line on rightmost plot). In order to be able to approximate each local extremum with a Gaussian, we can initially set $M := n + 1$ – we will get back to discussing this choice towards the end of this section.

Pursuing the idea of using one Gaussian for each local extrema $\ell_1, \dots, \ell_{n+1}$, we can set the initial means at the local extrema: $\mu_i := \ell_i$. We then set the sign of the respective amplitudes to match maxima and minima: $a_i = \mathrm{sign}(\ell_i) \cdot A$, where A is some default value for the amplitude. In our experiments, $A = 0.4$ worked well. Finally, the initial standard deviations are set to some fixed value; we used $\sigma_i = \frac{1}{\sqrt{\gamma_i}} = 0.2$.

The lowest approximation error is achieved by superposed Gaussians with large amplitudes and large standard deviations, but such approximations are prone to overfitting and noise: large amplitudes and large standard deviations cancelling eachother are undesirable as they are non-local, require more entanglement and are more susceptible to noise.

We study this closer in figure 4. The left plot (fig. 4a) shows how the relative approximation error decreases as maximal standard deviation increases, until the errors plateau at $10^{-6} - 10^{-7}$, reaching the limits of machine precision. We can also see that required standard deviations required decrease as we consider higher energy fock states. However, the right plot (fig. 4b) highlights that this increased precision comes at the cost of overfitting: between the middle ($\sigma_{\max} = 0.8$) and rightmost plot ($\sigma_{\max} = 1.4$), the dashed teal Gaussians that sum up to the Fock state approximation (in black) increase dramatically in amplitude and standard deviation, with a very minimal increase in precision. While the resulting approximation of the Fock state in the leftmost plot is perceptibly imperfect, the approximation in the two plots on the right is indistinguishable from the actual Fock

state: the relative errors for the leftmost plot is 46%, whereas the two other are of order 10^{-3} and 10^{-5} respectively. Thus, in this case, setting the maximum standard deviation to $\sigma_{\max} = 0.8$ yields an acceptable approximation whilst limiting overfitting. Using $M = n + 1$ Gaussians, we obtain indeed a faithful SoG approximation of the n-th Fock state. For $n = 5$, the relative error drops below 0.1% for $\sigma_{\max} \geqslant 0.8$, becoming even smaller for higher energy states.



(a) Approximation error as a function of maximum standard deviation.

(b) Gaussian approximations for three different maximum standard deviation constraints.

Figure 4: Influence of maximum standard deviation on approximation error and overfitting, for Gaussian approximations of the $n = 5$ Fock state.

This result can be extended to arbitrary CV states: consider a CV state $|\psi\rangle$ with maximal energy level $n_{\max}$, i.e. that can be expressed as a superposition of energy states $|0\rangle, \ldots, |n_{\max}\rangle$. Since the n-th energy state can be expressed as SoG of $n + 1$ Gaussians, it follows that the $|\psi\rangle$ state can be approximated as SoG of

$$\sum_{n=0}^{n_{\max}} n + 1 = \frac{(n_{\max} + 1)(n_{\max} + 2)}{2} = \frac{1}{2} n_{\max}^2 + O(n_{\max}). \qquad [15]$$

Gaussians.

This result provides a practical benchmark for SoG approximations of CV states. In implementations, the limiting factors will come from the number of superpositions that can be constructed and the precision that can be achieved for the respective amplitudes, as well as the minimum standard deviation that can be resolved: if the approximation is run on CV hardware, this will for instance be given by the squeezing amounts achievable.

Finally we can consider whether the number of Gaussians M in the SoG approximation of the n-th energy eigenstate can be reduced to less than $n+1$. Figure 5 shows the error rates achievable for smaller values of M. The number of Gaussians is given as a proportion of the number of local extrema: $\frac{M}{n+1}$. For $M = n+1$, error rates of order 10^{-6} are achievable, as we saw in Figure 4. For any value

$M < n + 1$, however, the Fock state approximations fail with error rates of unit order of magnitude (above the dashed 40% error threshold). This confirms the choice $M = n + 1$: for smaller values of $M \leqslant n$, the approximation error immediately jumps to over 40%, confirming that the bound in equation 15 is tight for a Fock-based approximation scheme.

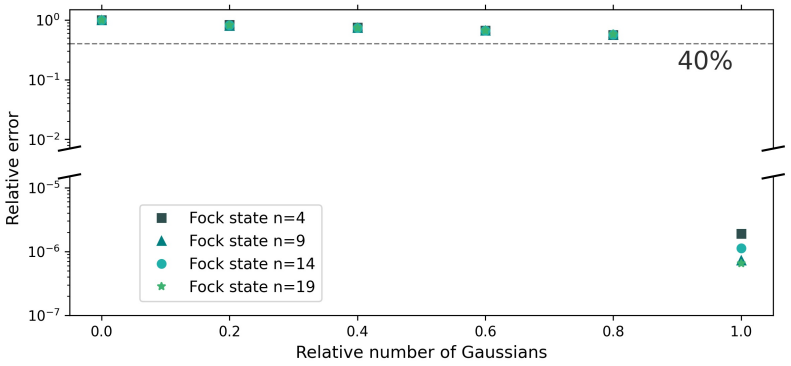


Figure 5: Approximation error for four Fock states as a function of the number of superposed Gaussians used in the approximation.

The cubic phase state

We had motivated the SoG approximation ansatz to be able to simulate arbitrary non-Gaussian transformations. The cubic phase state, as we saw above, is one of the most popular resources for non-Gaussian computations. Following the same ansatz as for Fock states, we are now interested in finding an SoG approximation for the cubic phase state. Recall that the cubic phase state (eq. 9) is given by

$$\psi_{\text{cubic}}(x) = e^{i\gamma x^3}.$$

First of all, it is important to remark that the cubic phase state cannot be normalised, and as such it can only be approximately reproduced experimentally. Note also that in contrast to the approximation of Fock states discussed above, the cubic phase state has complex amplitude, so that we can no longer ignore the imaginary part of the wave function. We generalise our previous ansatz to a superposition of complex Gaussians

$$\tilde{\psi}_{|n\rangle}(x) = \sum_{i=1}^{m} a^{(i)} \exp(i p_0^{(i)} x) \cdot \exp\Big(-\frac{1}{2}(\gamma_{\text{Re}}^{(i)} + i\gamma_{\text{Im}}^{(i)})(x - \mu^{(i)})^2 \Big),$$

where $a^{(i)}, p_0^{(i)}, \gamma_{\text{Re}}^{(i)}, \gamma_{\text{Im}}^{(i)}$ and $\mu^{(i)}$ are real variables to be optimised. We further adapt the least squares optimiser to minimise for both the real part and imaginary part of residuals.

To obtain a normalised state, we clip the cubic phase state to a finite interval around zero. We (arbitrarily) choose the initial parameters to coincide with the local extrema of the real part of the wave function. The real part oscillates as given by the term $\cos(\gamma x^3)$, so that there are infinitely many local extrema. Choosing a value of M and picking the M extrema closest to zero will fix the interval used for the cubic phase state approximation.

In practice, the size of the finite support interval will be dictated by the number of Gaussians that can be superposed, or by the smallest standard deviation achievable. For the latter, we can use a first-order approximation to see how the distance between two local extrema $\Delta\xi$ decreases with larger $|x|$

$$\gamma(x + \Delta\xi)^3 - \gamma x^3 = 2\pi \qquad \Longleftrightarrow \qquad \Delta\xi \approx \frac{2\pi}{3\gamma x^2}.$$

The distance between peaks $\Delta\xi$ gives a good estimate of the resolution necessary for the standard deviation σ of the Gaussians, as we expect them to scale linearly:

$$\sigma \propto \Delta\xi \propto \frac{1}{x^2}.$$

Figure 6 shows the resulting SoG approximation. The cubic phase parameter was set to $\gamma = 0.0118$, the support was clipped to the interval $[-15, 15]$ and the standard deviations were clipped to $\sigma_{\max} = 2.045$. The small relative error (0.4%) makes the resulting approximation given in black indistinguishable from the target cubic phase state.

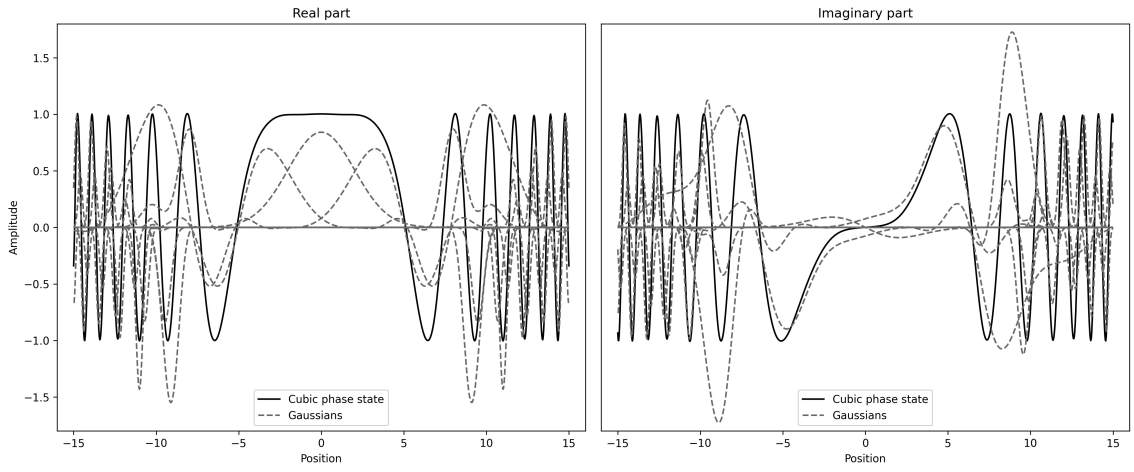


Figure 6: Gaussian approximation of the cubic phase state. The Gaussians (dashed line) sum up to an approximation of the cubic phase state (solid line).

The cubic phase state is a promising non-Gaussianity resource in theory, as we saw that it can be used to implement the cubic phase gate using Gaussian operations. In practice, however, the fact that it is impossible to normalise means that

successful use of the cubic phase state will always depend on how accurately it can be approximated. We followed the arguably simplest normalisation approach, which consists in clipping the state to finite support on the x-axis. Note that this is equivalent to clipping to low energy modes: the cubic phase state exponent grows with $|x|$, and thus so does its energy.

The impact of the size of the finite support interval on the approximation of the cubic state can be visualised in figure 7. There, the Wigner function of the approximated cubic phase state is plotted for different interval sizes, increasing from left to right. In the plots, the x-axis was rescaled to be independent of γ, as the cubic phase state only depends on γx^3. The interval sizes on both x and p dimensions are multiples of the plotted phase space: from left to right corresponds to x and p support intervals that are double, respectively 4, 6 and 8 times the size of the plotted phase space.

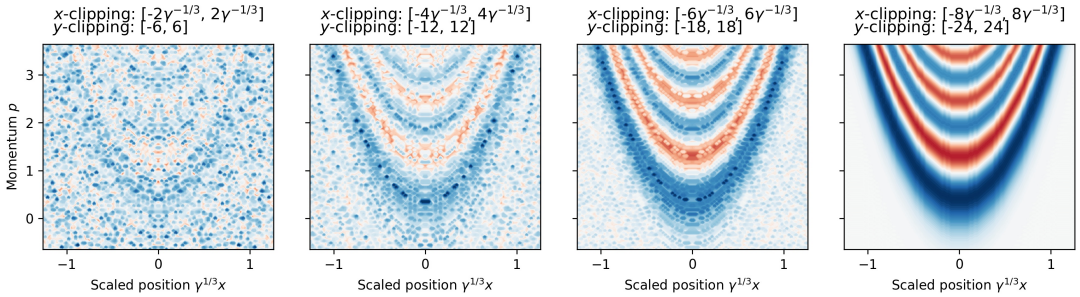


Figure 7: Wigner function of cubic phase state approximation for different clippings. The x-axis was rescaled to be independent of γ. Note that in these visualisations both x and p were clipped for numerical reasons.

We see that clipping the support interval induces significant noise in the state: while, the Wigner function in the rightmost plot coincides with the Wigner function, the other plots suffer from large approximation errors. Even in the third plot from the left, where the plotted position and momentum only represents one sixth of the clipping window for each axis, we obtain a relative approximation error of 49%. For the cubic phase state plotted in figure 6, this would correspond to a x-clipping to $[-26, 26]$ and a superposition of 132 Gaussians – as opposed to the $[-15, 15]$ interval of figure 6 that uses 43 Gaussians.

We can also visualise the approximation error of figure 6 by directly plotting the Wigner function of the SoG approximation. This can be seen in figure 8. The left plot shows the cubic phase state SoG approximation at the same scale as the plots in figure 7. The approximation at that scale is so poor that it is impossible to even recover the cubic phase state. This is not surprising given that the $[-15, 15]$ clipping interval corresponds to $\approx 3\gamma^{-1/3}$, i.e. to a cubic phase state approximation between the first and second plot on the left. The approximation will be better for smaller values of γ, since the support interval will be larger relative to the scaled position: this can be seen in the right plot showing the Wigner function

of the SoG cubic phase state approximation for $\gamma = 0.06$.

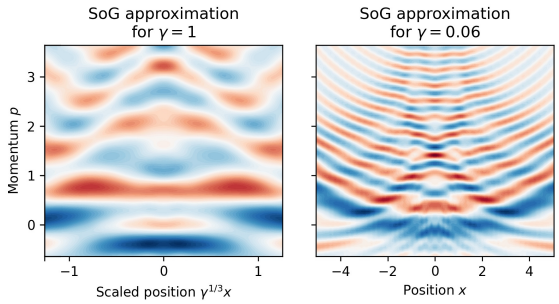


Figure 8: The Wigner function for the SoG cubic phase state approximation.

2.6 Entangled Gaussian states as superpositions

Given the success of the SoG approach to approximate arbitrary CV states, we now wish to apply it to two-mode entangled Gaussians. Indeed, if it is possible to express arbitrary single mode states as superpositions of Gaussians, i.e. Gaussians form an overcomplete basis of single mode CV states, then it must be that product of Gaussians span the entire space of multi mode CV states, and in particular entangled Gaussians. We call this approach superposition of product Gaussians (SoPG).

In contrast to n mode arbitrary Gaussians, whose covariance matrix is expressed by $O(n^2)$ terms, product states can be expressed by n matrices of size 2×2, i.e. with $O(n)$ terms. Not only is the necessary storage space smaller for product states, but computations are also cheaper: consider an arbitrary n mode Gaussian with covariance

$$\sigma_{AB} = \begin{bmatrix} \sigma_A & \epsilon_{AB} \\ \epsilon_{AB}^\top & \sigma_B \end{bmatrix}.$$

Assume that the subsystem A has 1 mode and the subsystem B has the remaining $n-1$ modes. An arbitrary one mode Gaussian transformation on A is given by some 2×2 symplectic matrix $\mathbf{S}$, and by equation 6, acts on σ_{AB} as follows:

$$\sigma_{AB} \mapsto \left(\mathbf{S} \oplus \mathbb{1}_{2(n-1)}\right) \sigma_{AB} \left(\mathbf{S}^\top \oplus \mathbb{1}_{2(n-1)}\right) = \begin{bmatrix} \mathbf{S}\sigma_A\mathbf{S}^\top & \mathbf{S}\epsilon_{AB} \\ \epsilon_{AB}^\top\mathbf{S}^\top & \sigma_B \end{bmatrix} \qquad [16]$$

In contrast, product states are characterised by $\epsilon_{AB} = 0$, so that the symplectic operation simplifies to

$$\sigma_{AB} \mapsto \begin{bmatrix} \mathbf{S}\sigma_A\mathbf{S}^\top & 0 \\ 0 & \sigma_B \end{bmatrix}. \qquad [17]$$

Hence, local single mode operations on arbitrary entangled n mode states require $O(n)$ operations, whereas the same operation on product states can be performed in $O(1)$ operations.

In order to be applicable in practice, we are focusing on approximating entangled Gaussians as SoPG expressions that are as simple as possible, at the cost of higher errors. This can be achieved using a least squares approach similar to the SoG approximations in 2.5. In the rest of the section, we will look at the simplest possible SoPG approximation, given by superpositions of two Gaussians for two mode entangled states.

Consider a two mode Gaussian $|\Gamma\rangle$ obtained from the vacuum state by one single-mode squeezing on each mode, followed by a two-mode beamsplitter:

$$|\Gamma\rangle = \mathbf{BS}(\theta)\,(\mathbf{S}(\xi_1) \oplus \mathbf{S}(\xi_2))\,|\nu\rangle\,,$$

where $|\nu\rangle$ is the vacuum state, characterised by zero displacement and identity covariance matrix. The covariance matrix of $|\Gamma\rangle$ is given by

$$\begin{aligned}
\sigma_\Gamma &= \mathbf{BS}(\theta)\,(\mathbf{S}(\xi_1) \oplus \mathbf{S}(\xi_2))\,\mathbb{1}_4\left(\mathbf{S}(\xi_1)^\top \oplus \mathbf{S}(\xi_2)^\top\right)\mathbf{BS}(\theta)^\top \\
&= \mathbf{BS}(\theta)\begin{bmatrix} e^{-2\xi_1} & 0 & 0 & 0 \\ 0 & e^{2\xi_1} & 0 & 0 \\ 0 & 0 & e^{-2\xi_2} & 0 \\ 0 & 0 & 0 & e^{2\xi_2} \end{bmatrix}\mathbf{BS}(\theta)^\top \\
&= e^{-2\xi_1}\,|\mathbf{R}_x\mathbf{e}_1\rangle\langle\mathbf{R}_x\mathbf{e}_1| + e^{2\xi_1}\left|\mathbf{R}_p\mathbf{e}_2\right\rangle\left\langle\mathbf{R}_p\mathbf{e}_2\right| + e^{-2\xi_2}\,|\mathbf{R}_x\mathbf{e}_3\rangle\langle\mathbf{R}_x e_3| \\
&\quad + e^{2\xi_2}\left|\mathbf{R}_p\mathbf{e}_4\right\rangle\left\langle\mathbf{R}_p\mathbf{e}_4\right|, \qquad [18]
\end{aligned}$$

where $\mathbf{e}_1, \ldots, \mathbf{e}_4 \in \mathbb{R}^4$ represent the canonical basis and

$$\mathbf{R}_x = \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{R}_p = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & 0 & -\sin\theta \\ 0 & 0 & 1 & 0 \\ 0 & \sin\theta & 0 & \cos\theta \end{bmatrix}$$

are rotation matrices. The reason for the slightly awkward notation in equation 18 is to highlight how the beamsplitter in effect rotates the squeezed state in the x- and p-modes independently. We can thus restrict our considerations to the x-axes, so that in effect we are only considering a two dimensional space. The p dimensions are identical (and independent).

Considering only the x_1- and x_2-modes with covariance matrix given by σ_x, we can express the Gaussian $|\Gamma\rangle$ using the three parameters $a, b, c \in \mathbb{R}$:

$$\sigma_x^{-1} = \begin{bmatrix} a & c \\ c & d \end{bmatrix}.$$

The Gaussian can be rewritten as follows (for some normalisation $\mathcal{N}_\Gamma \in \mathbb{R}$)

$$\begin{aligned}
&\frac{1}{\mathcal{N}_\Gamma}\exp\left(-\frac{1}{2}\left(ax_1^2 + 2cx_1x_2 + bx_2^2\right)\right) \\
&= \frac{1}{\mathcal{N}_\Gamma}\exp\left(-\frac{1}{2}a\left(x_1 + \frac{c}{a}x_2\right)^2 - \frac{1}{2}\left(b - \frac{c^2}{a}\right)x_2^2\right) \\
&= \frac{1}{\mathcal{N}_\Gamma}\exp\left(-\frac{1}{2}a\left(x_1 + \frac{c}{a}x_2\right)\right)\exp\left(-\frac{1}{2}\left(b - \frac{c^2}{a}\right)x_2^2\right).
\end{aligned}$$

This is almost a product of single mode Gaussians, except that the mean of the x_1-Gaussian depends on x_2. We can thus imagine a SoPG approximation scheme by replacing the x_2 dependency by a sum of Gaussian with fixed means along the $\left(-\frac{c}{a}x_2, x_2\right)$-axis. In other words, we make the following ansatz to approximate $|\Gamma\rangle$ as a superposition of two product Gaussians:

$$\underline{\alpha}\sum_{i=1}^{2}\exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{d}_i)^\top\begin{bmatrix}a & 0\\ 0 & \underline{\beta}\end{bmatrix}(\mathbf{x}-\mathbf{d}_i)\right), \text{ where } \begin{cases}\mathbf{x} &= \begin{bmatrix}x_1 & x_2\end{bmatrix}^\top \\ \mathbf{d}_{1,2} &= \pm\begin{bmatrix}-\frac{c}{a}\underline{\gamma} & \underline{\gamma}\end{bmatrix}^\top.\end{cases} \quad [19]$$

with model parameters $\underline{\alpha}, \underline{\beta}, \underline{\gamma} \in \mathbb{R}$. Note how we have used the symmetry of Gaussian state in the ansatz. These parameters can be optimised with least squares. An example of the resulting approximation is shown in figure 9.

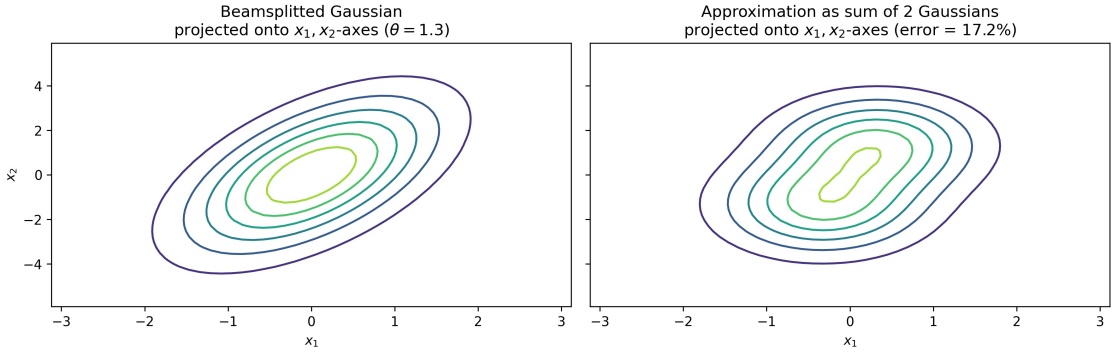


Figure 9: SoPG approximation of an entangled Gaussian state as a superpostion of 2 Gaussians. Only the x_1 and x_2 position quadratures are plotted. This example corresponds to the parameters $a = 0.4721$, $b = 0.1380$ and $c = -0.1005$.

3 Simulating CV computations

Given the potential of CV quantum computations and the current lack of hardware capable of executing continuous operations natively, the research in simulation of CV operations is a topic of central importance. This is precisely the focus of this work.

Research into CV simulation has so far been limited and focused on simulations on classical hardware, with arguably the most recognised effort being the Strawberry Fields project [32]. While advances have been made towards genuinely continuous quantum computers [26, 33], the recent progress in qubit-based quantum computing [34–36] suggests that the first genuine applications in quantum computing are likely to be qubit-based.

Our discussion of CV descriptions in the last chapter gives us an excellent foundation to design and compare simulation models. In this chapter, we will discuss existing simulations of CV operations on classical hardware and make first steps to design an approach suitable for qubit hardware. We postulate that SoG approximations as presented in section 2.5 can leverage the non-classical computational power of quantum qubit devices. This could allow to run arbitrary CV computations on qubit devices and take advantage of quantum speedup in the absence of CV computers.

3.1 The classical Fock based approximation

The most straightforward representation of a CV state on a discrete machine is through projection onto a finite dimensional Hilbert space. CV states can then be stored using the finite state vector. This corresponds to choosing a basis of CV state space and fixing the finite dimensional Hilbert space by selecting a finite subset of the basis.

The Fock basis is a popular choice as it is discrete and orthogonal. A finite subset is obtained by setting a maximum energy level E_n and discarding the Fock states of higher energies

$$|\psi\rangle \approx \sum_{k=0}^{n} \alpha_k |k\rangle , \quad \alpha_0, \dots, \alpha_n \in \mathbb{C}.$$

Using orthogonality, the components can easily be obtained by $\alpha_k = \langle k|\psi\rangle$ for $k = 0, \dots, n$. This representation can simulate arbitrary CV operations and works well for low energy states and a small number of modes. It is widely used in practice for classical simulations [32].

However, this representation has several drawbacks. Firstly, like any exact classical simulation of quantum computation, it suffers from exponential scaling in the number of modes: the state vector of a k-mode system with energy levels $\leqslant n$ has n^k components. This could be solved by a simulation on a qubit quantum

computer if the Fock basis vectors were encoded in an orthonormal basis of a qubit Hilbert space (e.g. in the computational basis).

The second, more fundamental, problem is that many usual CV operations are hard to express in the truncated Fock basis. As an example, consider the simplest Gaussian operation: displacement. Recall that a displacement $D(\alpha)$ in the Gaussian formalism is given by

$$D(\alpha)\begin{bmatrix} x \\ p \end{bmatrix} = \sqrt{2}\begin{bmatrix} \mathrm{Re}(\alpha) \\ \mathrm{Im}(\alpha) \end{bmatrix} + \begin{bmatrix} x \\ p \end{bmatrix},$$

that is, the displacement operator shifts the mean position and momentum while leaving the shape of the wave function unchanged. The closed form expression for $D(\alpha)$ in the Fock basis is given by [20]

$$\begin{aligned} D(\alpha)\,|n\rangle = &\sum_{k=0}^{n} \sqrt{\frac{k}{n}} \exp\left(-\frac{1}{2}|\alpha|^2\right)(-\alpha^*)^{n-k} L_n^{n-k}(|\alpha|^2)\,|k\rangle \\ &+ \sum_{k=n+1}^{\infty} \sqrt{\frac{n}{k}} \exp\left(-\frac{1}{2}|\alpha|^2\right)\alpha^{k-n} L_n^{k-n}(|\alpha|^2)\,|k\rangle\,, \end{aligned} \tag{20}$$

where $L_n^d(x)$ are the extended Laguerre polynomials.

In the case of a Fock-based simulation on a qubit device, implementing the displacement operation $D(\alpha)$ would correspond to performing the unitary with entries given by the summands of equation 20. With current technology, such an operation is very hard to decompose into circuit gates and will be extremely sensitive to noise. Given that the matrix is dense and non-local, it will also scale poorly with increasing maximum energy and truncating it to finite dimension will mean that the approximated operation will be non-unitary.

In summary, Fock-based approximation schemes are hard and impractical to implement on qubit devices with limited resources. Other similar approaches based on projections onto finite dimensional Hilbert spaces, for example based on a finite discrete subset of position eigenfunctions, suffer from the same limitations, where simple Gaussian operations become hard to implement in circuits.

For near-term implementation of CV operations on qubit devices, an alternative approach is needed. Ideally, this would combine the expression power of the Fock basis representation with the efficiency of the Gaussian representation for Gaussian operations. Given that arbitrary CV operations can be expressed using Gaussian operations and one single type of non-Gaussian gate, a speedup in Gaussian simulations could result in a significant overall performance increase. This is precisely what SoG approximations promise.

3.2 Gaussian operations on digital quantum hardware

The first step in devising a quantum implementation of SoG approximations is to fix how Gaussians and their operations are represented. We will explore two

alternatives, which we will call amplitude encoding and qubit encoding respectively. In amplitude encoding, we leverage the density matrix representation of qubit states to encode a covariance matrix σ as a mixed states ρ. The qubit encoding on the other hand maps Gaussian states to the computational basis.

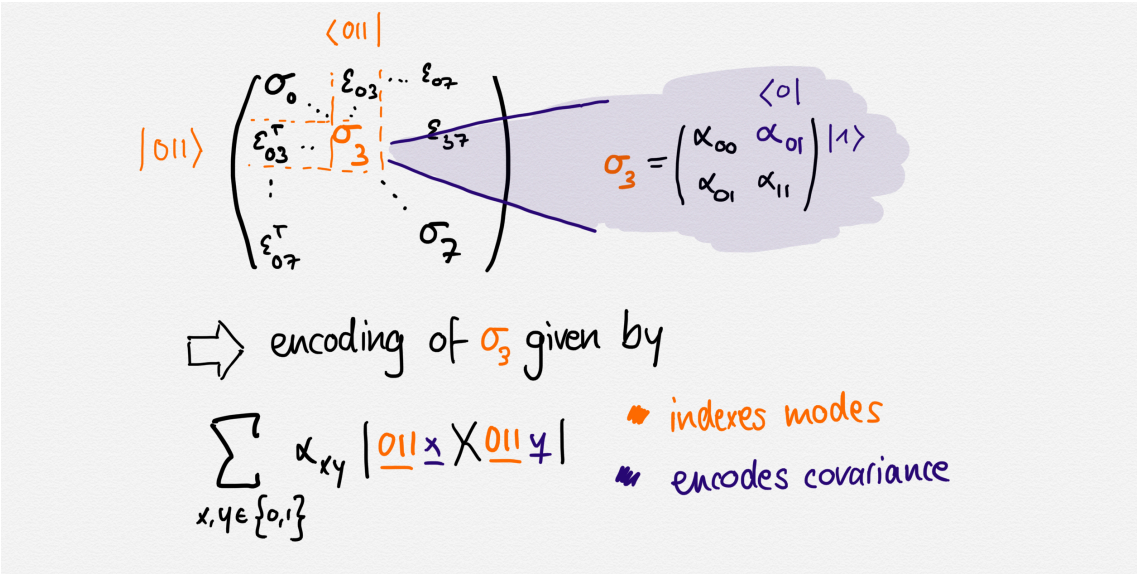
Amplitude encoding

Given that both mixed states and the covariance matrix of Gaussian states are described by positive-definite symmetric matrices, it is natural to suggest a Gaussian state representation using mixed states. We assign to each Gaussian state a qubit state with density matrix given by the covariance matrix of that state, unique up to trace normalisation:

$$\sigma \mapsto \rho := \frac{1}{\mathrm{Tr}\sigma}\sigma.$$

Displacement and normalisation factors need also be stored separately. Displacement can be represented and manipulated in a very similar way to the covariance matrix, whilst normalisation factors are most easily stored classically (n scalar values for n modes). In the following, we will focus on representing the most interesting part, the covariance matrix.

In this encoding, single mode Gaussian CV states correspond to a single qubit. The number of qubits grows logarithmically with the number of modes: 8 modes are described by a 16×16 covariance matrix, given by 4 qubits. One way to view the correspondence between modes and qubits in amplitude encoding is to split the qubits in two groups: the least significant qubit (by convention) encodes the covariance matrix of each mode, whilst all the remaining qubits serve to index the right covariance matrix:



In this example, the mode is indexed using binary encoding. Performing a local operation on a subset of modes can then be achieved by controlling the desired

operation on the index qubits. This can become expensive for many modes depending on the cost of Toffoli gates. Alternative encodings to the binary encoding could solve this problem at the cost of less efficient encodings (i.e. more indexing qubits): for instance, using a linear encoding $b_0 \dots b_n$ where $b_i = \delta_{ik}$ encodes mode k would only require simply controlled operations (i.e. CNOTs) at the expense of a number of qubits linear in the number of modes. Arbitrary encodings can be devised to take advantage of the trade-off between the number of qubits and complexity of local operations depending on the computational costs incurred in practice.

How do we then implement Gaussian operations? Gaussian transformations are given by symplectic transformations S of the density matrix:

$$\sigma \mapsto \mathbf{S}\sigma\mathbf{S}^T.$$

These must be implemented by unitary evolution of the type $\mathbf{U}^\dagger\sigma\mathbf{U}$, or more generally by completely positive trace-preserving (CPTP) maps obtained from unitaries performed on an augmented state, following by tracing and possibly post selection (Stinespring dilation) [37]. This can be achieved using the Euler-Messiah decomposition [20]: for any symplectic transformation S, there are orthogonal matrices O, O' and a diagonal matrix D such that $\mathbf{S} = \mathbf{O}_1\mathbf{D}\mathbf{O}_2$, and hence

$$\sigma \mapsto \mathbf{O}_1^T\mathbf{D}^T\mathbf{O}_2^T\sigma\mathbf{O_1}\mathbf{D}\mathbf{O_2}.$$

The orthogonal transformations are in particular unitary, so that they can be seen as usual quantum operations on density matrices. The diagonal transformation, on the other hand, can be achieved by a CPTP map, albeit with a non-zero probability of failure. Here too, the operation must be normalised first

$$\mathbf{D} \mapsto \frac{1}{\det(\mathbf{D})}\mathbf{D}.$$

In the following, we assume without loss of generality that this is already the case.

An operation given by a diagonal $2^n \times 2^n$ matrix can be achieved by a $2^{n+1} \times 2^{n+1}$ unitary transformation. The quantum computation on the enlarged system can then be traced out to the original system to obtain the desired diagonal transformation $\mathbf{D}$. Given that the transformation must be normalised, all diagonal entries $\mathrm{diag}(\mathbf{D}) = (d_1, \dots, d_{2^n})^\top$ satisfy $0 \leqslant d_i \leqslant 1$ for $1 \leqslant i \leqslant 2^n$. The columns of $\mathbf{D}$ can thus be completed to orthogonal 2^{n+1}-vectors:

$$d_i\mathbf{e}_i \mapsto \underbrace{d_i\mathbf{e}_i + \sqrt{1-d_i^2}\mathbf{e}_{2^n+i}}_{\mathbf{g}_i :=} \in \mathbb{R}^{2^{n+1}}.$$

These can be put together into an isometry $\mathbf{X} \in \mathbb{R}^{2^{n+1}\times 2^n}$

$$\mathbf{X} := \begin{bmatrix}\mathbf{g}_1 & \dots & \mathbf{g}_{2^n}\end{bmatrix} = \begin{bmatrix}\mathbf{D} \\ \sqrt{\mathbb{1}_{2^n} - \mathbf{D}^2}\end{bmatrix} \quad \text{with } \mathbf{X}^\dagger\mathbf{X} = \mathbb{1}_{2^{n+1}} \qquad [21]$$

The $\mathbf{X}$ matrix of size $2^{n+1} \times 2^n$ can be completed to a $2^{n+1} \times 2^{n+1}$ unitary matrix $\mathbf{U}$, for instance using Gram-Schmidt orthonormalisation. This yields a $2^{n+1} \times 2^n$ matrix $\mathbf{Y}$ such that

$$\mathbf{U} := \big[\mathbf{X}, \mathbf{Y}\big] \quad \text{with } \mathbf{U}^\dagger \mathbf{U} = \mathbf{U}\mathbf{U}^\dagger = \mathbb{1}_{2^{n+1}}.$$

From $\mathbf{U}$, it is easy to recover the matrix $\mathbf{D}$ by post-selecting the ancillary qubit A:

$$\langle 0_A | U | 0_A \rangle |\psi\rangle = \frac{1}{\mathcal{K}} D |\psi\rangle, \tag{22}$$

where the renormalisation factor $\mathcal{K}$ implies that the computation will fail with probability $f = 1 - \frac{1}{\mathcal{K}}$.

A high failure probability f can make computations considerably slower, or even infeasible as the failure rates multiply with the number of gates. f typically depends on the state $|\psi\rangle$ of the system. In the worst case, we get

$$f_{\max} = 1 - \min_{1 \leqslant i \leqslant 2^n} d_i.$$

This value will be related to the amount of squeezing performed by the operation: for the case of single mode squeezing by a factor s, we have $\mathbf{D} = \begin{bmatrix} e^{2s} & 0 \\ 0 & e^{-2s} \end{bmatrix}$, and thus

$$f \approx 1 - e^{-2s} \approx= 2s.$$

These significant failure rates are obviously bad news for practical applications. One approach to reducing the failure rate is to look for alternative CPTP maps where states from failed attempts can be corrected to the required form instead of being discarded. One promising avenue of work is to change how the columns of $\mathbf{D}$ are normalised: instead of defining $\mathbf{X}$ as in equation 21, we can consider obtaining an isometry $\mathbf{X}'$ by using permutations of $\mathbf{D}$: $\sigma_1(\mathbf{D}), \dots, \sigma_k(\mathbf{D})$, where the σ_i denote some permutation of $\{1, \dots, 2^n\}$ and $\sigma_i(\mathbf{D})$ denotes the matrix $\mathbf{D}$ with diagonal entries permuted by σ_i:

$$\sigma_i(\mathbf{D}) := \operatorname{diag}\Big(\sigma_i(d_1, \dots, d_{2^n})\Big), \quad 1 \leqslant i \leqslant k.$$

We choose permutations $\sigma_1, \dots, \sigma_k$ such that there exists $R = \operatorname{diag}(r_1, \dots, r_{2^n}) > 0$, with entries $r_1, \dots, r_{2^n}$ as small as possible that satisfy

$$\mathbf{X} := \begin{bmatrix} \mathbf{D} \\ \sigma_1(\mathbf{D}) \\ \vdots \\ \sigma_k(\mathbf{D}) \\ \mathbf{R} \end{bmatrix} \quad \Longrightarrow \quad \mathbf{X}^\dagger \mathbf{X} = \mathbb{1}_{2^n}. \tag{23}$$

This can then be extended to a unitary matrix with Gram-Schmidt in the same way as earlier. Note that we are now not using one ancilla qubit but $\lceil \log_2 (k+2) \rceil$ ancilla.

The way to read equation 23 is to realise that the entries of the unitary that correspond to X completely determine the image of the unitary when the most significant qubit (by our convention, that is the ancilla qubit) is $|0\rangle$. Conversely, after the measurement of the ancilla qubit, each row of X corresponds to the operation that was performed as a result of the state collapse. Hence, initialising the ancilla qubit to $|0_A\rangle$ as we considered in equation 22 and measuring the ancilla qubits, we obtain that the collapsed output state must be one of

$$\mathbf{D}|\psi\rangle, \quad \sigma_1(\mathbf{D})|\psi\rangle, \ldots, \sigma_k(\mathbf{D})|\psi\rangle, \quad \mathbf{R}|\psi\rangle.$$

The motivating idea is that we should be able to recover states that are permutations of $\mathbf{D}$ by permutating the diagonal elements back in the correct order after the computation. The probability of failure would then entirely depend on the reminder $\mathbf{R}$; and the smaller the diagonal elements become, the smaller the entries of $\mathbf{R}$!

However, this does not work as hoped: the original diagonal transformation is recovered from a permutated diagonal matrix if a basis change given by the inverse permutation can be performed. Suppose that after the transformation the state collapses into the $\sigma_1(\mathbf{D})|\psi\rangle$ state. The only way we would recover the $\mathbf{D}|\psi\rangle$ state is with

$$\mathbf{D}|\psi\rangle = \left(\sigma_1^{-1} \circ \sigma_1(\mathbf{D}) \circ \sigma_1\right)|\psi\rangle.$$

But this would require that we perform σ_1 before we even know in which state the measurement will be collapsed – ie we cannot perform different permutations depending on the output of the measurement.

The high failure probability is the main drawback of the amplitude encoding strategy. Note however that in contrast to Fock-based approximation, or the qubit encoding approximation that we will discuss next, arbitrary Gaussian operations on amplitude encoded CV states are not only simple to express, they are also local (up to the control operations on the indices): a single mode Gaussian operation is encoded as a single qubit operation. In particular, operations on off-diagonal covariance terms come for free in amplitude encoding. For a two mode state, for instance, recall that the evolution of the covariance matrix is given by (eq. 16)

$$\sigma_{12} \mapsto (\mathbf{S} \oplus \mathbb{1}_2)\,\sigma_{12}\,(\mathbf{S}^\top \oplus \mathbb{1}_2) = \begin{bmatrix} \mathbf{S}\sigma_1\mathbf{S}^\top & \mathbf{S}\epsilon_{12} \\ \epsilon_{12}^\top\mathbf{S}^\top & \sigma_2 \end{bmatrix}.$$

Computing this classically, or using qubit encoding, requires a number of operations that scales linearly with the number of modes. Instead, in amplitude encoding, the operation $\mathbf{S} \oplus \mathbb{1}_2$ corresponds precisely to the controlled operation that is performed. This means further that "structure" from the Gaussian transformation will also be preserved in the simulation: for instance, sparsity will be preserved, and diagonal symplectic matrices will also result in diagonal qubit transformations.

Qubit encoding

Encoding information in amplitudes is very sensitive to noise and requires quantum circuits that provide high levels of accuracy. Together with the significant failure rate, this makes amplitude encoding challenging to implement in near-term quantum devices.

We thus explore another approach that reduces Gaussian operations to classical circuits, which can usually be implemented with high accuracy. The idea is to discretise the set of Gaussian functions and map them onto the orthonormal computation basis: for instance, the single mode Gaussian state given by a displacement $\mathbf{d} = \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}$ and the covariance matrix $\sigma = \begin{bmatrix} a & c \\ c & b \end{bmatrix}$ may be encoded as

$$|\psi\rangle = |\text{bin}(a)\text{bin}(b)\text{bin}(c)\text{bin}(d_1)\text{bin}(d_2)\rangle\,,$$

where $\text{bin}(x)$ is some discretised binary encoding of x.

Since Gaussian operations send Gaussian states onto Gaussian states, the implementation of any such operation in a qubit circuit will be given by a classical circuit: any computational basis element is sent onto another element of the computational basis. The classical circuit for a symplectic transformation S is precisely given by the boolean circuit that performs the binary arithmetic that map $\mathbf{d} \mapsto \mathbf{Sd}$ and $\sigma \mapsto \mathbf{S}\sigma\mathbf{S}^\top$. Finding an optimal circuit that implements this operation is a task of classical computer science, with the exception that quantum operations are restricted to reversible gates. There are efficient circuits known for this task [38, 39].

The main drawback that arises from this approach has been hinted at a few times. It is a consequence of equation 16 that descrbes the evolution of covariance matrices under Gaussian operations: the cost of a single mode operation grows linearly with the number of modes, and as such this implementation cannot scale well in practice.

The superposition of product of Gaussians (SoPG) approximation presented in 2.6 offers a solution to this problem: instead of discretising the entire Gaussian state space, we only discretise Gaussians that are product states (i.e. not entangled). We then represent entangled Gaussians as superpositions of product Gaussians.

As shown by equation 17, this sets all off-diagonal terms of the covariance matrix to zero, so that the computational effort for single mode operations is constant. On the downside, however, this means that arbitrary Gaussian operations are no longer necessarily classical: entanglement-creating operations such as beamsplitters map product states to entangled states, which must be approximated by SoPG states.

In effect, finding an implementation for the two mode beamsplitter in the SoPG approximation can be reduced to expressing rotations of Gaussian states as SoPG states. Figure 9 gives an example of this: in that figure, the plot on the left

would the rotated Gaussian that we wish to obtain from the Gaussian operation – since this is not a SoPG state, this Gaussian would instead be approximated by the SoPG state in the right plot.

In the SoPG approximation with 2 Gaussians presented in section 2.6, implementing such a rotation comes down to mapping the displacement and covariance matrix of one Gaussian to the parameters $\underline{\alpha}, \underline{\beta}, \underline{\gamma} \in \mathbb{R}$ of the 2 Gaussian SoPG parametrisation given in equation 19. For simplicity, let us restrict ourselves to input states with zero displacement and axes-aligned covariance matrices – the most general case can then be obtained with additional rotations and displacements. For a fixed rotation of angle θ, a simple geometric argument shows that the variances $\underline{\alpha}$ and $\underline{\beta}$ of the output state will scale linearly with the entries of the input covariance matrix, while the displacement $\underline{\gamma}$ will scale with the square root – a linear growth in the variance is in fact quadratic in the standard deviation. Since we assumed that the covariance matrix of the input state is diagonal, we conclude that the parameters $\underline{\alpha}, \underline{\beta}, \underline{\gamma} \in \mathbb{R}$ only depend on the major/minor axis ratio of the input covariance matrix

$$\frac{\sigma_1}{\sigma_2}, \quad \text{where} \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} \text{ is the input covariance matrix.}$$

Thus, all we need to implement a beamsplitter for the SoPG approximation for fixed θ is to express explicitly the functional dependency of $\underline{\alpha}, \underline{\beta}$ and $\underline{\gamma}$ on the major/minor axis ratio. Note that for Gaussian pure states, this corresponds in fact to the amount of squeezing in the initial state. Finding a closed form algebraic expression has proven challenging, thus we resorted to numerical tests that we present in figure 10. These show the parameters $\underline{\alpha}, \underline{\beta}$ and $\underline{\gamma}$ of the two product Gaussian approximation defined in equation 19 of the image of an axes-aligned initial Gaussian, as a function of the major/minor axis ratio.

It should be fairly straightforward in practice to implement a beamsplitter based on these observed dependencies, even though further work in understanding the mathematical relationship between these parameters would be helpful. Depending on the accuracy achievable in the circuits, a first simple beamsplitter might be achieved by implementing the crude linear approximation given in orange in figure 10.

Measurements

Both for amplitude and qubit encoding, the issue of measurement has not been addressed. For practical use, the ability of simulating CV measurements is crucial. This does not appear to be an easy task in either of the approximation strategies.

Two types of measurements can be differentiated: destructive measurements and non-destructive measurements. Destructive measurements perform measurements on the CV state and return the experiment outcome – the CV state,

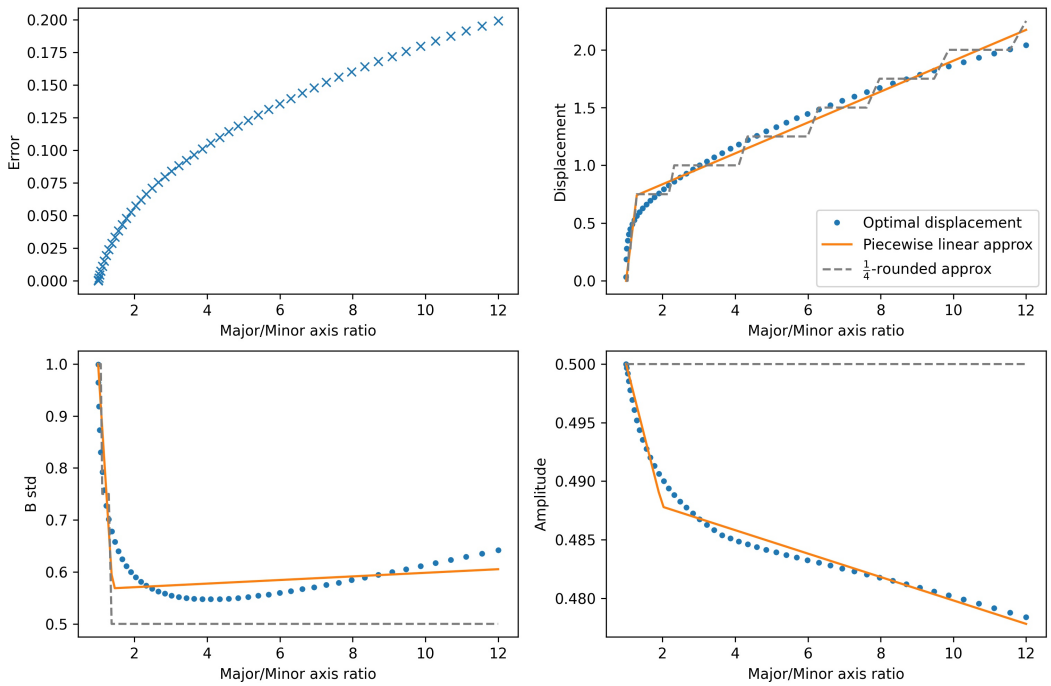


Figure 10: Dependence of approximation error and model parameters of the 2 Gaussian SoPG approximation (eq. 19), as a function of the major/minor axis ratio of the input state.

however, is destroyed and cannot be used for further computations. This is in contrast with non-destructive measurements, which, as their name implies, allow measurements on subsets of CV modes, after which the remaining modes are still available for further computations.

In theory, arbitrary quantum operations can be obtained from destructive measurements at the cost of potentially high failure probability: measurements to be performed are delayed until every other operation has been executed – in the end, all measurements are performed simultaneoulsy and, in the case of post-selection, sampled outcomes that do not correspond to the desired measurement outcomes are discarded. In practice, however, this approach is highly undesirable, as the induced failure probability is high. Furthermore, non-Gaussian CV measurements such as photon-counting are a precious resource for non-Gaussianity: non-Gaussian states can be obtained from Gaussian operations followed by measurements and post-selection.

In both encoding strategies, measurements are hard to simulate because the non-orthogonal set of Gaussian states is mapped onto orthogonal elements of the computational basis. Performing measurements on the qubits (e.g. in the com-

putational basis) projects the measured state into orthogonal spaces, discarding the interference terms that arise from non-orthogonal Gaussians. One avenue of work that could solve this issue is implementing an operation mapping Gaussian states, as they are represented in the respective strategies, onto a discretised set of position eigenstates, encoded in the computational basis. This would in effect construct a discretised wave function of the CV state. Performing a computational basis measurement would yield the outcome of a homodyne measurement of x-quadrature. This would be a destructive measurement, as a Gaussian-to-position eigenstates map is not invertible.

Implementing non-destructive Gaussian measurements would require implementing the formula given in equation 11:

$$\sigma_A^{\hat{\Pi}} = \sigma_A - \epsilon_{AB}(\sigma_B + \mathbf{\Gamma}_B^{\hat{\Pi}})^{-1}\epsilon_{AB}^{\top}.$$

This could be achievable in the qubit encoding strategy, the biggest hurdle being the implementation of the matrix inverse operation. These two options seem promising and more work should assess their viability.

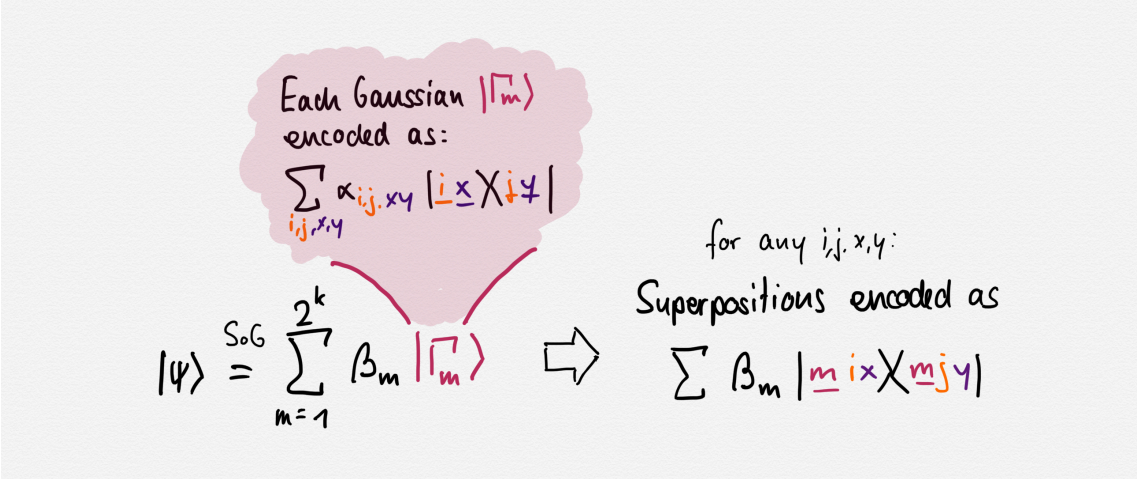
For the time being, in the absence of any quantum operation for measurement simulation, it is always possible to simulate measurements classically using quantum state tomography [40]. The represented CV state can then be reconstructed and classical computations can perform simulated measurements.

3.3 Non-Gaussian states in amplitude and qubit encoding

In the previous section, we have presented two strategies, amplitude and qubit encoding, to simulate Gaussian states and operations on discrete quantum hardware. Given the SoG approximation we have introduced in section 2.5, the Gaussian capabilities can be extended to arbitrary computations by simulating superpositions of Gaussians.

In qubit encoding, simulating superpositions comes for free: a single Gaussian corresponds to a computational basis element, and so superpositions of Gaussian states can be mapped to superpositions of computational basis elements. The same qubit-encoded operations that act on Gaussian states will linearly act on each element of the superposition.

In the amplitude encoding, on the other hand, superpositions of Gaussian states can be achieved by extending the mode-indexing qubits: a single n mode Gaussian state is given by one LSB qubit encoding the components of the covariance matrices and $\lceil \log(n) \rceil$ qubits that index the modes. We can then express superpositions of 2^k Gaussian by adding k indexing qubits:



With this representation, amplitude-encoded operations also come for free: if **S** is to be applied on a single Gaussian, then the operation on k superpositions is given by $\mathbb{1}_k \otimes \mathbf{S}$, i.e. by applying **S** to the subsystem of mode indices and covariance-encoding qubit.

Thus, amplitude and qubit encoding can be used to simulate arbitrary operations. The main limitation is the difficulty of achieving accurate approximations of the (normalised) cubic phase state, as was discussed in figures 7 and 8. Alternative resources for non-Gaussianity which are more suited to numerical approximations are needed to be able to simulate non-Gaussian states with greater accuracy. In theory, any non-Gaussian state can be used to generate non-Gaussian operations. The authors in [27] suggest for instance using Fock states as non-Gaussianity resource, which have simple SoG approximations. However, the use of such non-Gaussian operations is limited by our ability to decompose arbitrary CV computations into these gates, so that more work is needed that combines the search for non-Gaussian states that are numerically stable with new decomposition methods that can take advantage of those particular states.

3.4 Proof-of-concept implementations

We are completing this work with a short demonstration of CV simulation on qubit devices in practice. We implemented both the qubit and amplitude encoding strategy in Python using the Pennylane [41] and Strawberryfields [32] libraries, which provide a comprehensive toolset for simulation of both digital and continuous quantum computations.

The idea is to implement simulation backends for CV computations, which themselves rely on a qubit simulation backend. Put in different words, the implementation provides a wrapper that, given a qubit simulation backend, returns a backend for CV simulations. Once Pennylane and the plugin have been installed, the CV simulation can be used like any other Pennylane CV device, entirely abstracting away the qubit simulation:

```
import pennylane as qml

# ampenc or qubitenc provide simulators
# with amplitude resp. qubit encoding
dev_str = "CVSimulation.ampenc"

# this example has 3 modes
dev = qml.device(dev_str, wires=3)
```

Both simulators provided rely on a qubit backend to perform the simulation; this can be specified with the `dev_qubit` argument. Note that in contrast to the qubit encoding, the amplitude encoding strategy is based on mixed states so that instead of the default Pennylane qubit simulation device, a custom `qubit_mixed` device is required, which is based on the Pennylane default qubit simulation device, with the additional support for simulation of mixed state computations: it supports mixed state preparation with the `MixedStatePrep` operation. Using another qubit backend means the simulation can immediately be run on any available qubit hardware.

Any CV computation can then be performed using the standard Pennylane interface:

```
@qml.qnode(dev)
def circuit():
    # let variables mu, cov, r, phi and d be defined
    qml.GaussianState(mu, cov, wires=[2]])
    qml.TwoModeSqueezing(r, phi, wires=[1,2])
    qml.Displacement(d, wires=[1]])

    return qml.expval(qml.NumberOperator(2))
```

Note that only Gaussian measurements are supported.

This approach gives a flexible framework to develop further CV simulation strategies by providing the abstraction of a CV simulation device. Whilst the current implementation and our advances in the SoG ansatz are too limited in scope to be of any public use, we hope that this framework can be used in more mature future work. In this spirit of contribution to future work, we will highlight two particular challenges that future CV simulations will have to face.

Controlling the classical overhead

One of the main challenges in the implementation of CV simulation strategies on qubits is that there always remains a significant classical computational overhead to preparing and executing CV simulations: even as the resulting qubit circuits are efficient, expensive classical computations for circuit synthesis might well make it impossible to scale the approach to large systems. This difficulty is perhaps most

evident in the Fock-based and other basis projection representations we presented initially (section 3.1). In the absence of more structure, implementing CV operations in these basis must rely on explicit matrix representations, that must then be decomposed into elementary qubit gates, which scales poorly, even if the final qubit circuit is efficient.

SoG strategies have been designed precisely to make simple CV operations simple to implement. We can thus hope that in the future arbitrary CV computations can be run on qubit hardware efficiently. For our proof-of-concept, we performed some timed simulations on amplitude-encoded Gaussian operations to compare the potential quantum speedup in a purely Gaussian setting. In figure 11, we see that only a small proportion (the leftmost bar) of overall computing is dedicated to classical pre- and post-processing. The three Pauli measurements correspond to quantum computations run on the classical quantum simulation backend. These could be run on real quantum hardware and provide a significant speedup to the overall computation. The remaining operations are the classical overhead, taking up less than 5% of the overall execution time. An implementation of CV operations on genuine quantum hardware would thus provide a significant speedup.

Note that the classical overhead of quantum circuit construction in our example is by itself already bigger than the total computational effort required by the Pennylane CV classical simulator given by the dashed line. This points to the optimisation potential of our approach. More fundamentally, however, the qubit simulation approach can be generalised to arbitrary CV operations, in stark contrast to the Gaussian classical simulator – while the Pennylane simulator might be currently better optimised for Gaussian operations, non-Gaussian operations will never be supported.

Qubit encoding: high approximation errors

Compared to amplitude encoding, qubit encoding is simpler to implement on error-prone hardware with limited gate precision, given that much of the computation can be achieved with classical circuits and limited entanglement. However, achieving the desired precision for the simulation might require a significant number of qubits: rotation matrices, on the one hand, must be expressed by values between -1 and 1, whilst displacement operations can take arbitrary values $\gg 1$. Hence, if we choose to discretise Gaussian covariance matrices and displacement vectors up to a precision of $\frac{1}{64}$ for values in the range $[-3, 4]$, a single mode Gaussian will require (2 components for displacement and 3 for upper triangle of covariance matrix)

$$5 \cdot (\log_2 8 + \log_2 64) = 45 \text{ qubits.}$$

Since this represents a single mode, this holds for both SoG approximations – in which case the number of qubits will scale quadratically with the number of modes – and for the SoPG approximation – in which case the number of qubits

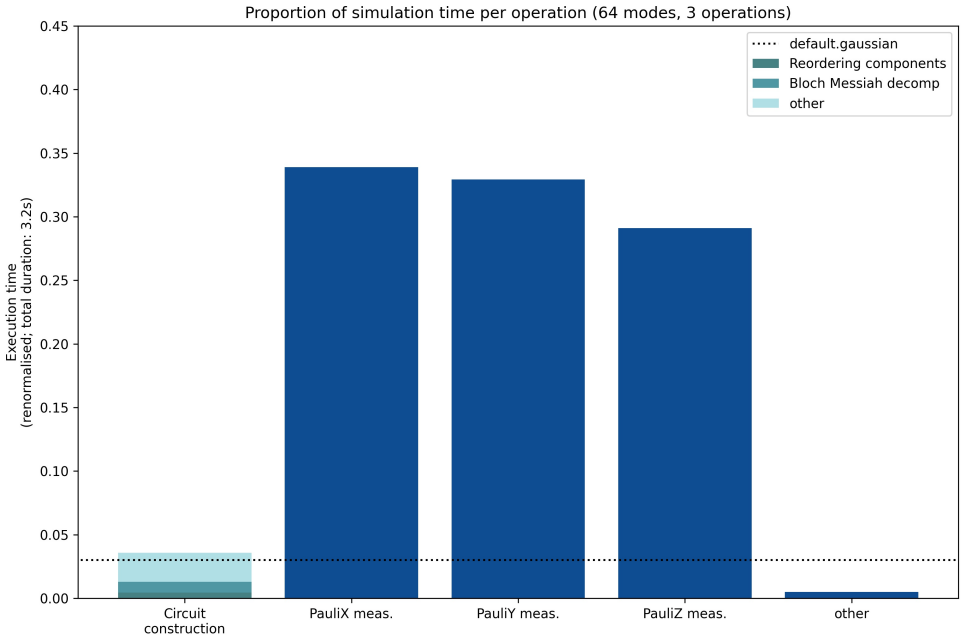


Figure 11: Mean execution time of CV simulation over 10 runs, split by task.

will grow linearly.

On top of this, we will require further ancillary qubits, as well as ancillary modes if we wish to implement non-Gaussian operations – this is beyond what we are able to simulate today, let alone run on quantum hardware. Using floating points could reduce the number of qubits required, at the expense of more complex quantum circuits, which also quickly reach the depth of circuit constraints.

For this proof-of-concept, we implemented a single mode Gaussian encoding using 3 bits (of which 2 bits for decimal precision) for each of the three components of the covariance matrix and 2 bits (of which 0 bits for decimal precision) for the 2 displacement components. This corresponds to a precision of $\frac{1}{4}$ on the interval $[0, 1]$ for the covariance components and a precision of 1 on the interval $[0, 3]$ for displacements. In our naive implementation based on general-purpose arithmetic boolean circuits, this resulted in a total of 31 qubits, including the auxiliary qubits.

The results of a rotation on a superposition of two Gaussians in the qubit encoding is shown and compared to a Fock simulation in Figure 12. The actual result, as can be computed by applying the rotation to each Gaussian separately, is shown on the left. The middle shows the result from the classical Fock simulation of the rotation. The right is the result obtained by the qubit encoded simulation (3 bits for covariance components, 2 bits for displacement components).

The considerable error after a single operation makes it clear that more qubits (and smarter implementations) are needed in order to make qubit encoding feasible in practice. One promising approach that deserves further work is to base qubit encoded operations on finite fields instead of approximated floating point operations. There might be significant advantages to this approach when considering near term applications with significant limitations in the number of qubits. First suggestions on this approach will be elaborated in our next and final chapter, in which we will discuss the possible next steps.

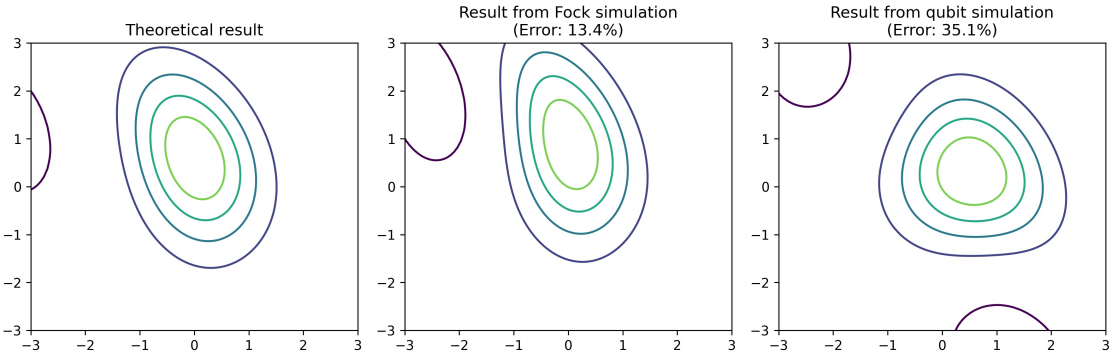


Figure 12: Approximations of the Wigner function of a rotated 2 Gaussian SoPG state, as obtained analytically (left), using a Fock simulator (middle) and using qubit encoding (right). The border effects that can be seen are due to numerical approximations.

4 Conclusion

This dissertation has presented some first steps towards the approximation of continuous-variable quantum operations on discrete qubit devices. We have detailed why existing approximation strategies for arbitrary CV operations implemented on classical machines are not suited for qunatum implementations and have suggested approaches based on superpositions of Gaussians as an alternative. These approaches are characterised by a native support for Gaussian states and simple implementations of Gaussian CV operations. Previous research has shown promising signs that, together with non-Gaussian state as offline resources, the implementation of Gaussian operations should be sufficient for efficient universal simulation of CV operations.

We have presented two flavours of SoG approximations: amplitude encoding and qubit encoding. The former provides a concise representation of CV states using few qubits, but requires hardware that can executed circuits with high precision to guarantee accurate simulations. On the other hand, many operations take a very simple form in qubit encoding, but precision must be guaranteed by using a larger number of qubits.

For short to medium term applications, the latter approach seems more likely to succeed, given that noise is expected to remain one of the largest limitations of quantum computing. A first priority for further research should thus be to aim to mitigate the limited precision that can be achieved on small systems by exploring smart state encodings and specialised implementations of Gaussian operations, as opposed to the general-purpose arithmetic circuits that we implemented in this work.

The calculus of finite fields may provide fertile ground precisely for such approaches. Instead of relying on floating point operations to compute state evolution, finite fields operations could be used as approximation of Gaussian operations. This would eliminate the approximation error due to the discretisation of the Gaussian state space by making all arithmetic operations exact. Of course, approximation error will come instead from differences between the real and the finite fields calculus – nonetheless, this could provide reliable operations away from the "edges" of the finite field using a limited number of qubits. Furthermore, in contrast to the floating point approximation, in which arithmetic operations are no longer unitary, all Gaussian operations would be unitary. Preliminary considerations have even raised the prospect that in the finite fields framework the basis change transformation from Gaussian states to position eigenstates might be easy to implement, yielding the prospect of CV measurement simulation on qubit hardware.

Beyond the issue of measurements and limited precision, there are many further open questions that must be tackled in future work, such as engineering non-Gaussian states that are both practical to approximate as SoG and that are convenient to use in practice as a source of non-Gaussianity. Nonetheless, we

are confident that this research has lain the ground work for the development of CV simulation on qubit machines; it is in any case beyond doubt that a successful implementation of continuous operations on qubit computers would enable quantum applications for a wide range of computational problems and thus be pivotal for quantum computing.

References

[1] Paul Benioff. 'The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines'. In: *Journal of Statistical Physics* 22.5 (May 1980), pp. 563–591. DOI: 10.1007/bf01011339.

[2] Kevin Marshall et al. 'Quantum simulation of quantum field theory using continuous variables'. In: *Physical Review A* 92.6 (Dec. 2015). DOI: 10.1103/physreva.92.063825.

[3] Nathan Killoran et al. 'Continuous-variable quantum neural networks'. In: *Physical Review Research* 1.3 (Oct. 2019). DOI: 10.1103/physrevresearch.1.033063.

[4] Maria Schuld and Nathan Killoran. 'Quantum Machine Learning in Feature Hilbert Spaces'. In: *Physical Review Letters* 122.4 (Feb. 2019). DOI: 10.1103/physrevlett.122.040504.

[5] David Money Harris and Sarah L Harris. *Digital design and computer architecture*. 2nd ed. Amsterdam ; Boston: Elsevier/Morgan Kaufmann, 2012.

[6] Fredrick J Hill and Gerald R Peterson. *Introduction to switching theory and logical design*. eng. New York: Wiley, 1968.

[7] Simon S. Haykin and Barry Van Veen. *Signals and systems*. eng. 2nd ed. New York, NY: Wiley, 2003.

[8] 'IEEE Standard for Binary Floating-Point Arithmetic'. In: *IEEE Std. 754-1985* (1985). DOI: 10.1109/ieeestd.1985.82928.

[9] Frederic T. Chong, Diana Franklin and Margaret Martonosi. 'Programming languages and compiler design for realistic quantum hardware'. In: *Nature* 549.7671 (Sept. 2017), pp. 180–187. DOI: 10.1038/nature23459.

[10] Maria Schuld et al. 'Evaluating analytic gradients on quantum hardware'. In: *Physical Review A* 99.3 (Mar. 2019). DOI: 10.1103/physreva.99.032331.

[11] E. Torrontegui and J. J. García-Ripoll. 'Unitary quantum perceptron as efficient universal approximator'. In: *EPL (Europhysics Letters)* 125.3 (Mar. 2019), p. 30004.
DOI: 10.1209/0295-5075/125/30004.

[12] Yudong Cao, Gian Giacomo Guerreschi and Alán Aspuru-Guzik. 'Quantum Neuron: an elementary building block for machine learning on quantum computers'. In: *e-prints* (Nov. 2017).
arXiv: 1711.11240 [quant-ph].

[13] Maria Schuld et al. 'Circuit-centric quantum classifiers'. In: *Physical Review A* 101.3 (Mar. 2020).
DOI: 10.1103/physreva.101.032308.

[14] Edward Farhi and Hartmut Neven. 'Classification with Quantum Neural Networks on Near Term Processors'. In: *e-prints* (Feb. 2018).
arXiv: 1802.06002 [quant-ph].

[15] Kwok Ho Wan et al. 'Quantum generalisation of feedforward neural networks'. In: *npj Quantum Information* 3.1 (Sept. 2017).
DOI: 10.1038/s41534-017-0032-4.

[16] Hoi-Kwan Lau et al. 'Quantum Machine Learning over Infinite Dimensions'. In: *Physical Review Letters* 118.8 (Feb. 2017).
DOI: 10.1103/physrevlett.118.080501.

[17] Paul E. Parris. *Lecture notes in Quantum One*. [online — accessed 28.08.20]. Jan. 2005.
Available at http://web.mst.edu/~parris/QuantumOne/Class_Notes.

[18] Gerardo Adesso, Sammy Ragy and Antony R. Lee. 'Continuous Variable Quantum Information: Gaussian States and Beyond'. In: *Open Systems & Information Dynamics* 21.01n02 (Mar. 2014), p. 1440001.
DOI: 10.1142/s1230161214400010.

[19] Bonny L. Schumaker. 'Quantum mechanical pure states with gaussian wave functions'. In: *Physics Reports* 135.6 (Apr. 1986), pp. 317–408.
DOI: 10.1016/0370-1573(86)90179-1.

[20] Alessandro Ferraro, Stefano Olivares and Matteo Paris. 'Gaussian states in continuous variable quantum information'. In: *e-prints* (Mar. 2005).
arXiv: quant-ph/0503237.

[21] Quntao Zhuang, Peter W. Shor and Jeffrey H. Shapiro. 'Resource theory of non-Gaussian operations'. In: *Physical Review A* 97.5 (May 2018).
DOI: 10.1103/physreva.97.052317.

[22] Seth Lloyd and Samuel L. Braunstein. 'Quantum Computation over Continuous Variables'. In: *Physical Review Letters* 82.8 (Feb. 1999), pp. 1784–1787.
DOI: 10.1103/physrevlett.82.1784.

[23] Stephen D. Bartlett and Barry C. Sanders. 'Universal continuous-variable quantum computation: Requirement of optical nonlinearity for photon counting'. In: *Physical Review A* 65.4 (Mar. 2002).
DOI: 10.1103/physreva.65.042304.

[24] Mattia Walschaers et al. 'Entanglement and Wigner Function Negativity of Multimode Non-Gaussian States'. In: *Physical Review Letters* 119.18 (Oct. 2017).
DOI: 10.1103/physrevlett.119.183601.

[25] R. M. Gomes et al. 'Quantum entanglement beyond Gaussian criteria'. In: *Proceedings of the National Academy of Sciences* 106.51 (Dec. 2009), pp. 21517–21520.
DOI: 10.1073/pnas.0908329106.

[26] Olivier Pfister. 'Continuous-variable quantum computing in the quantum optical frequency comb'. In: *Journal of Physics B: Atomic, Molecular and Optical Physics* 53.1 (Nov. 2019), p. 012001.
DOI: 10.1088/1361-6455/ab526f.

[27] Shohini Ghose and Barry C. Sanders. 'Non-Gaussian ancilla states for continuous variable quantum computation via Gaussian maps'. In: *Journal of Modern Optics* 54.6 (Mar. 2007), pp. 855–869.
DOI: 10.1080/09500340601101575.

[28] Alexander Barvinok. *Combinatorics and Complexity of Partition Functions*. Springer International Publishing, 2016.
DOI: 10.1007/978-3-319-51829-9.

[29] Gerald B. Folland. *Harmonic Analysis in Phase Space. (AM-122)*. Princeton University Press, 1989.

[30] C Voglis and IE Lagaris. 'A rectangular trust region dogleg approach for unconstrained and bound constrained nonlinear optimization'. In: *WSEAS International Conference on Applied Mathematics*. 2004, p. 7.

[31] Pauli Virtanen et al. 'SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python'. In: *Nature Methods* 17 (2020), pp. 261–272.
DOI: 10.1038/s41592-019-0686-2.

[32] Nathan Killoran et al. 'Strawberry Fields: A Software Platform for Photonic Quantum Computing'. In: *Quantum* 3 (2019), p. 129.
DOI: 10.22331/q-2019-03-11-129.

[33] Y. Zhang et al. 'Single-mode quadrature squeezing using dual-pump four-wave mixing in an integrated nanophotonic device'. In: *arXiv e-prints*, arXiv:2001.09474 (Jan. 2020), arXiv:2001.09474.
arXiv: 2001.09474 [physics.optics].

[34] Frank Arute et al. 'Quantum supremacy using a programmable superconducting processor'. In: *Nature* 574.7779 (Oct. 2019), pp. 505–510.
DOI: 10.1038/s41586-019-1666-5.

[35] Yuanhao Wang et al. '16-qubit IBM universal quantum computer can be fully entangled'. In: *npj Quantum Information* 4.1 (Sept. 2018).
DOI: 10.1038/s41534-018-0095-x.

[36] Peter J Karalekas et al. 'A quantum-classical cloud platform optimized for variational hybrid algorithms'. In: *Quantum Science and Technology* 5.2 (Apr. 2020), p. 024003.
DOI: 10.1088/2058-9565/ab7559.

[37] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2009.
DOI: 10.1017/cbo9780511976667.

[38] Feng Wang et al. 'Improved quantum ripple-carry addition circuit'. In: *Science China Information Sciences* 59.4 (Feb. 2016).
DOI: 10.1007/s11432-015-5411-x.

[39] Igor L. Markov and Mehdi Saeedi. 'Constant-Optimized Quantum Circuits for Modular Multiplication and Exponentiation'. In: *e-prints* (Feb. 2012).
arXiv: 1202.6614 [cs.ET].

[40] Marcus Cramer et al. 'Efficient quantum state tomography'. In: *Nature Communications* 1.1 (Dec. 2010).
DOI: 10.1038/ncomms1147.

[41] Ville Bergholm et al. 'PennyLane: Automatic differentiation of hybrid quantum-classical computations'. In: *e-prints* (Nov. 2018).
arXiv: 1811.04968 [quant-ph].