QuGIT: a numerical toolbox for Gaussian quantum states

I. Brandão, D. Tandeitnik, and T. Guerreiro

Departamento de Física, Pontifícia Universidade Católica do Rio de Janeiro, 22451-900 Rio de Janeiro, RJ, Brazil

Simulating quantum states on a classical computer is hard, typically requiring prohibitive resources in terms of memory and computational power. Efficient simulation, however, can be achieved for certain classes of quantum states, in particular the so-called Gaussian quantum states of continuous variable systems. In this work we introduce QuGIT - a python numerical toolbox based on symplectic methods specialized in efficiently simulating multimode Gaussian states and operations. QuGIT is exact, requiring no truncation of Hilbert space, and provides a wide range of Gaussian operations on arbitrary Gaussian states, including unitaries, partial traces, tensor products, general-dyne measurements, conditional and unconditional dynamics. To illustrate the toolbox, several examples of usage relevant to quantum optics and optomechanics are described.

1 Introduction

Simulating arbitrary quantum systems is a difficult task for classical computers. The memory necessary to store quantum states in cache and the complex computations required to emulate their dynamics generally scales exponentially with the number of modes for finite-dimensional Hilbert spaces [1]. For infinite-dimensional systems the situation is even worse since dynamics cannot be reproduced with limited memory in general, regardless of the number of modes. To overcome these limitations, one often works with truncated Hilbert space dimensions, which works well if the system has a small number of modes and excitations. Dimensional truncation is commonly used in a variety of quantum numerical packages [2, 3], and most prominently in the Quantum Toolbox in Python (QuTiP) [1, 4], which has been extensively used to simulate a wide range of systems from parametric amplifiers and frequency converters to superconducting qubits and quantum Monte-Carlo trajectories in cavity QED [1].

While dimension truncation offers a viable path to simulating a myriad of quantum systems, it is desirable to have complementary tools which are exact. Of particular interest to us is the special class of continuous variable Gaussian quantum states, which comprise a number of interesting situations, particularly in quantum optomechanics experiments involving strong coherent states [5] and nanomechanical resonators interacting with thermal environments possessing large occupation numbers [6, 7], both intractable

I. Brandão: igorbrandao@aluno.puc-rio.br
D. Tandeitnik: tandeitnik@aluno.puc-rio.br
T. Guerreiro: barbosa@puc-rio.br

using truncated Hilbert spaces. Quantum optical circuits involving Gaussian states with a large number of modes such as in Boson sampling [8] and photonic quantum computing [9] are also examples for which exact methods are desirable. For these needs we have developed the **Qu**antum **G**aussian **I**nformation **T**oolbox - or QuGIT for short - a python numerical toolbox for Gaussian quantum information applications.

Gaussian states are completely characterized by the first and second moments of their phase-space distribution [10]. This lifts the requirement of dealing with the infinite-dimensional objects characteristic of continuous variable systems, requiring only a finite covariance matrix for the complete and exact description of the system's state and dynamics. Consequently, memory requirements grow only quadratically with the number of modes N; specifically only $4N^2 + 2N$ double-precision floating point numbers in memory are needed to represent an N-mode Gaussian state. QuGIT builds upon the formalism of Gaussian quantum states to implement a series of useful tools for continuous variable quantum systems.

This paper is organized as follows. In Section 2, we present a brief introduction to Gaussian quantum states, its symplectic representation and Gaussian-preserving dynamics. The framework of QuGIT and how the toolbox is divided is presented in Section 3, while a detailed discussion is provided in Section 4 through a series of illustrative examples covering a wide range of the toolbox capabilities. The associated codes for these simulations are presented in Appendix B, and the list of QuGIT's built-in functions is given in Appendix A. In Section 5 we discuss the performance of QuGIT to simulate dynamics in comparison to QuTiP. Section 6 concludes with final considerations on the current state of the toolbox and how it contributes to the list of already existing packages for simulating quantum systems.

2 Brief review of Gaussian states and notation

2.1 Gaussian states

QuGIT is based on the formalism of *continuous variable quantum systems* whose states live in an infinite-dimensional Hilbert space described by observables with continuous spectra. In the remaining of this text we follow Refs. [10, 11, 12, 13] and a brief summary of the main properties regarding continuous variable (CV) systems is provided in this section.

An N-mode CV state is characterized by annihilation and creation operators $\hat{a}_j$ and $\hat{a}_j^\dagger$ ($j = 1 \dots N$) obeying the standard bosonic commutation relations $\left[\hat{a}_j, \hat{a}_k^\dagger\right] = \mathbb{1}\,\delta_{j,k}$. We choose to work in units where $\hbar = 2$, such that the canonical observables associated to each mode are defined as $\hat{x}_j \equiv \hat{a}_j^\dagger + \hat{a}_j$ and $\hat{p}_j \equiv i(\hat{a}_j^\dagger - \hat{a}_j)$. These can be interpreted as the position and momentum operators of a harmonic oscillator or field quadratures of an optical mode. In the remaining of this work we shall refer to $\hat{x}_j, \hat{p}_j$ as *quadrature operators* and it follows from the bosonic commutation relations that $[\hat{x}_j, \hat{p}_k] = 2i\mathbb{1}\,\delta_{j,k}$ and $[\hat{x}_j, \hat{x}_k] = [\hat{p}_j, \hat{p}_k] = 0$.

A convenient representation of quadrature operators is via a vector of the form $\hat{\boldsymbol{r}} = \left(\hat{x}_1, \hat{p}_1, \hat{x}_2, \hat{p}_2, \dots\right)^T$. This allows one to write the canonical commutation relations in compact notation,

$$[\hat{r}_j, \hat{r}_k] = 2i\,\Omega_{j,k} \ , \quad \Omega \equiv \bigoplus_{k=1}^{N} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} . \tag{1}$$

where Ω is the so-called symplectic form matrix.

The continuous spectra of the quadrature operators span a real symplectic space [10] referred to as *phase-space*. This allows for a convenient representation of arbitrary continuous variable states: the complete information contained in an N-mode bosonic quantum state ρ can be represented as a quasi-probability distribution in a $2N$-dimensional phase-space. In this work we are interested in the special class of continuous variable states for which the phase-space distribution assumes a Gaussian form. These so-called *Gaussian states* can be completely described by the first and second moments of their quadrature operators.

Let ρ_G be an N-mode Gaussian state; its first moment is given by the quadrature vector $\boldsymbol{R} \in \mathbb{R}^{2N}$, defined as

$$\boldsymbol{R} \equiv \langle \hat{\boldsymbol{r}} \rangle = \operatorname{tr}(\rho_G \hat{\boldsymbol{r}})\,, \tag{2}$$

while the second moments can be arranged into a real symmetric *covariance matrix* $V \in \mathbb{R}^{2n \times 2n}$, with entries given by

$$V_{j,k} = \frac{1}{2}\langle \hat{r}_j \hat{r}_k + \hat{r}_k \hat{r}_j \rangle - \langle \hat{r}_j \rangle \langle \hat{r}_k \rangle\,. \tag{3}$$

These are the basic data elements of QuGIT.

2.2 Gaussian preserving dynamics

Physical transformations that map Gaussian states to Gaussian states are called *Gaussian-preserving*. To be a *Gaussian-preserving unitary* the operator $\hat{S} = \exp\Big(-i\hat{H}_0/2\Big)$ must be generated by a Hamiltonian $\hat{H}_0$ that is *at most* quadratic in the quadrature operators and thus its most general form is

$$\hat{H}_0 = \frac{1}{2}\hat{\boldsymbol{r}}^T H(t) \hat{\boldsymbol{r}} + \boldsymbol{\alpha}_H^T \hat{\boldsymbol{r}} \tag{4}$$

where $H(t)$ is a time-dependent real symmetric $2N \times 2N$ matrix and $\boldsymbol{\alpha}_H$ is a real vector of length $2N$. Using the Heisenberg equations for the quadrature vector alongside its commutation relations given in Eq. (1) we find that Gaussian unitaries can be described through an affine mapping of the quadrature operators

$$\dot{\hat{\boldsymbol{r}}} = \Omega H(t) \hat{\boldsymbol{r}} + \Omega \boldsymbol{\alpha}_H \mathbb{1}\,. \tag{5}$$

The discussion on Gaussian-preserving transformations may also be generalized to account for open quantum system dynamics and continuous measurements. The open dynamics is modelled by considering that the system of interest is weakly coupled to a large reservoir in an M-mode Gaussian state with first and second moments $\boldsymbol{R}_B = \boldsymbol{0}$ and V_B. The environment is assumed to satisfy the *white noise* condition

$$\langle \Big\{ \hat{\boldsymbol{r}}_B(t), \hat{\boldsymbol{r}}_B^T(t') \Big\} \rangle = 2V_B\ \delta(t - t')\,, \tag{6}$$

implying Markovian dynamics. Moreover, the environment and system are allowed to interact through a quadratic coupling Hamiltonian, which preserves the Gaussian character of the global state,

$$\hat{H}_{\text{int}} = \hat{\boldsymbol{r}}^T C \hat{\boldsymbol{r}}_B\,, \tag{7}$$

where C is a $2N \times 2M$ real matrix describing the system-environment coupling and $\hat{\boldsymbol{r}}_B$ is the quadrature operator vector for the environment.

Continuous measurements are modelled by a general-dyne detection scheme acting on the environment giving rise to a conditional dynamics on the system. Note this scheme preserves the Gaussianity of the system. The choice of measurement is characterized by the post-measurement covariance matrix of the monitored modes V_M, while the probability of measuring an outcome $\boldsymbol{r}_M$ from these modes follows a Gaussian probability density with mean $\boldsymbol{R}_B$ and covariance $(V_B + V_M)^{-1/2}$. For a detailed description of general-dyne measurements see Ref [13].

After tracing out the environment, the conditional dynamics induced on the system by the total Hamiltonian $\hat{H} = \hat{H}_0 + \hat{H}_{\text{int}}$ plus continuous general-dyne measurements is described by a set of stochastic Langevin equations together with a deterministic Riccati equation for the first moments,

$$\dot{V} = AV + V^T A^T + D - \chi(V)\,, \tag{8}$$

$$d\boldsymbol{R} = (A\boldsymbol{R} + \boldsymbol{N})dt + (V\mathcal{C}^T + \Gamma^T)d\boldsymbol{w}\,, \tag{9}$$

where $d\boldsymbol{w} = (V_B + V_M)^{-1/2}(\boldsymbol{r}_m - \boldsymbol{R}_B)$ is a Wiener process with $\langle\{d\boldsymbol{w}, d\boldsymbol{w}^T\}\rangle = \mathbb{1}dt$. The *drift matrix* A, *diffusion matrix* D and *driving term* $\boldsymbol{N}$ dictate the unconditional dynamics on the system whilst the monitoring of the system is introduced through the positive definite matrices $\mathcal{C}, \Gamma$ and $\chi(V)$,

$$\begin{aligned} A &= \Omega H + \frac{1}{2}\Omega C \Omega C^T\,, & \chi(V) &= (V\mathcal{C}^T + \Gamma^T)(\mathcal{C}V + \Gamma)\,, \\ D &= \Omega C V_B C^T \Omega^T\,, & \Gamma &= (V_B + V_M)^{-1/2} V_B C \Omega\,, \\ \boldsymbol{N} &= \Omega \boldsymbol{\alpha}_H\,, & \mathcal{C} &= (V_B + V_M)^{-1/2}\Omega C\,. \end{aligned} \tag{10}$$

We note that when $\mathcal{C} = \Gamma = 0$ the deterministic unconditional dynamics is recovered from the above equations and the Riccati equation reduces to a Lyapunov equation. We also observe that the effect of continuous monitoring introduces stochasticity only in the mean quadrature vector while the covariance matrix follows a deterministic dynamics.

The above matrices and vectors, alongside the initial state for the system are the basic data elements necessary for QuGIT to calculate arbitrary Gaussian quantum dynamics.

3 QuGIT framework

3.1 Emulating Gaussian states

The toolbox is able to emulate an arbitrary multi-mode Gaussian state, perform Gaussian operations and retrieve data from these simulations. This is achieved through the custom Python class `gaussian_state`. The attributes of this class encompass all the necessary information to characterize the states: their number of modes, mean quadrature vectors, covariance matrices and the associated symplectic form matrix, summarized in Table 1.

Table 1: Attributes of the gaussian_state class

Attribute	Description
R	Mean quadrature vector
V	Covariance matrix
N_modes	Number of modes
Omega	Symplectic form matrix

Manipulations of the quantum state can be performed in two ways. One is through class methods that alter the class instance. The other is via homonym built-in functions that take as argument a `gaussian_state` and return a modified copy of the original class instance. Details regarding these operations are listed in the Appendix A.

We now present a first working example using the `gaussian_state` class. Consider a pair of two-mode Gaussian states. With the following code, we can find the their quantum fidelity:

```
import numpy as np
import quantum_gaussian_toolbox as qgt

R = np.array([1, 2, 3, 4])          # Mean quadrature vector for state 0
V = 10*np.eye(4)                    # Covariance matrix        for state 0
state_0 = qgt.gaussian_state(R, V) # Multimode Gaussian state

alpha = 1 - 2.0j                    # Coherent state 1 complex amplitude
state_1 = qgt.coherent(alpha)       # Single mode Gaussian state

# Tensor product of two copies of a coherent state (state_1)
state_2 = qgt.tensor_product([state_1, state_1]) # Library function
state_1.tensor_product([state_1])                # Class method

F = qgt.fidelity(state_0, state_1) # Quantum fidelity between states
```

The first lines of the code import the QuGit and Numpy packages. Note Numpy is necessary for the functioning of QuGIT. These imports and their aliases "qgt." and "np." are assumed in all examples from now on. The creation of an arbitrary two-mode Gaussian state is achieved by initialising the first two moments of the state (declared as numpy.ndarrays) and passing these to the `gaussian_state` class constructor. We can initialise certain elementary Gaussian states - such as the vacuum, coherent, squeezed and thermal states - by a builtin function with its associated parameter. In the above example we can see the definition of a complex number (`alpha`) and the coherent state initialised with corresponding complex amplitude (`gqgt.coherent(alpha)`). Next, we calculate the tensor product of the initialised coherent state with itself. This tensor product can be achieved in two different ways: first with the QuGIT library function `qgt.tensor_product`, which takes a list of `gaussian_state` class instances and returns another instance with the tensor product. Second, we can make use of the `gaussian_state` class method to alter the variable `state_1` to the result of the tensor product. Finally, we calculate the quantum fidelity between `state_0` and `state_1`.

We note that while the above example contains some of the core mechanics of the class `gaussian_state`, there are numerous capabilities which can be achieved using the toolbox; see Appendix A for more details. These capabilities include Gaussian unitaries, phase-space representations, entanglement witnesses, general-dyne measurements and calculation of density matrices in different basis. By retrieving the density matrix on the number basis, the toolbox allows for almost effortless synergy with the widely used QuTiP package [1].

3.2 Simulating time evolution

In addition to the state class, QuGIT contains the `gaussian_dynamics` class, for simulating Gaussian-preserving time evolution of a given initial state.

The class attributes record the necessary information to calculate the time evolution, meaning the initial state (which is an instance of `gaussian_state`), the matrices appearing in equations (8) and (9) (defined as numpy.ndarrays) and the list of time-evolved states.

The class methods are able to perform unconditional and conditional time evolution of multi-mode Gaussian states following the theory outlined in Section 2.1, as well as calculate the semi-classical dynamics for the system and their steady states.

For the unconditional dynamics the class performs numerical integration of the deterministic differential equations and finds the associated steady state by solving the algebraic form of the Lyapunov and Langevin equations. Regarding the conditional case, the deterministic Riccati equation is solved using the same standard numerical integration, while a Monte Carlo method is employed to integrate the stochastic Langevin equation and yield quantum trajectories, induced by continuous monitoring, upon the Hilbert space of the system. Finally, for the semi-classical dynamics we generalize the unconditional case by considering the effect of stochastic forces acting on the Langevin equation with zero mean value and auto-correlation dictated by the diffusion matrix D; the resulting semi-classical dynamics gives rise to semi-classical trajectories.

4 Examples

We now proceed to discuss some illustrative examples using QuGIT. The code to each example is presented in the Appendix B. The associated lines of code used for plotting are omitted for simplicity, however, we note that throughout the remaining of this work, the matplotlib package [14] is used for data visualization.

4.1 Unitary field quadrature dynamics

A standard plot in quantum optics textbooks [15] is the unitary time development of the mean field quadrature and its corresponding variances for the electromagnetic field in various quantum states. We begin with this simple example as the information required can be directly retrieved from the `gaussian_state` class attributes and serves to illustrate how one can use the toolbox to initialise states of interest, evaluate their unitary dynamics and visualize results via expectation values of canonical observables.

We consider two states: a coherent state with complex amplitude parameter $\alpha = 2$ and a squeezed-coherent state with amplitude $\alpha = 2$ and squeezing parameter and phase given by $r = 1.2, \phi = 0$. The result of this simulation is shown in Figure 1.

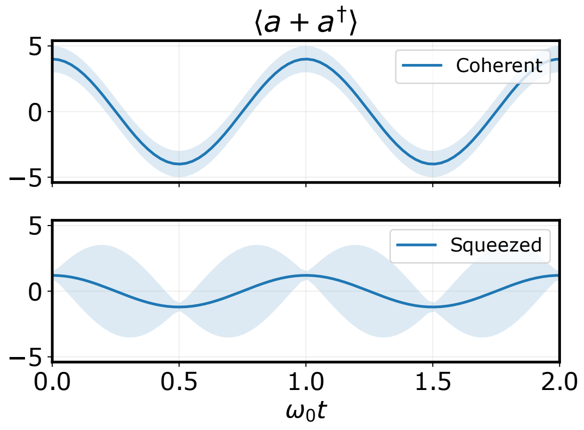

Figure 1: Time evolution of mean (solid line) and variance (shaded region) of the field quadrature for a single mode. Top: a coherent state with $|\alpha|^2 = 4$. Bottom: a squeezed-coherent state with $|\alpha|^2 = 4$ and $r = 1.2, \phi = 0$, displaying reduced noise at specific times showing that at those moments, the state is closer to an eigenstate of the electric field (X quadrature) than a coherent state.

4.2 Damped harmonic oscillator

We can extend the above example to include the effects of damping in the field. We consider the time evolution of a damped quantum harmonic oscillator governed by the Lindblad equation,

$$\dot{\rho} = -\frac{i}{\hbar}\left[\hbar\omega\hat{a}^\dagger\hat{a}, \rho\right] + \gamma\left(\hat{a}\rho\hat{a}^\dagger - \frac{1}{2}\hat{a}^\dagger\hat{a}\rho - \frac{1}{2}\rho\hat{a}^\dagger\hat{a}\right) , \tag{11}$$

where ω is the frequency of the mode and γ is the damping constant. The commutator in the master equation dictates unitary dynamics, while the second term governs the interaction with the environment. The open quantum dynamics is modelled through amplitude damping on the number of excitations of the harmonic oscillator. The associated Langevin and Lyapunov equations entailed by Equation (11) are characterized by a vanishing driving vector and the following drift and diffusion matrices,

$$A = \begin{bmatrix} -\gamma/2 & +\omega \\ -\omega & -\gamma/2 \end{bmatrix} , \quad D = \begin{bmatrix} \gamma & 0 \\ 0 & \gamma \end{bmatrix} . \tag{12}$$

We consider the system initially in a coherent state $|\alpha\rangle$ with $\alpha = 2$. A visualization of the dynamics can be seen in Figure 2. On the lower plots, we show snapshots of the Wigner function for the state at different moments before a full oscillation is complete. The dashed circles represent the expected mean trajectories produced by the unitary dynamics, for comparison. One can observe that mean position of the Gaussian distribution undergoes a circular damped motion, eventually settling at the origin corresponding to the stationary vacuum state.

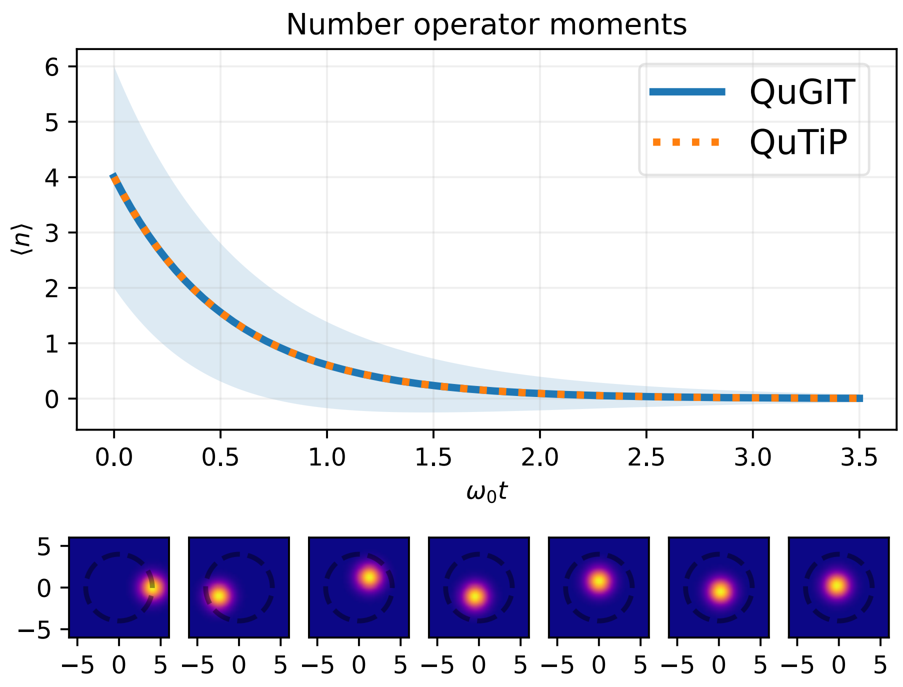

Figure 2: Damped time evolution of an initial coherent state, $|\alpha = 2\rangle$. Top: occupation number calculated through QuGIT (solid blue line) compared to QuTiP (orange dots), together with the variance of the mode number operator (shaded light blue region). Bottom: Wigner functions evaluated using QuGIT; time flows from left to right. Parameters used in the simulation are $\omega_0 = 2\pi$ and $\gamma = 2\pi \times 0.3$. For QuTiP, the Hilbert space dimension truncation was $N = 30$.

This damped motion of the distribution can also be visualized on the top plot, through the mean mode occupation number and its variance, both monotonically decreasing. For comparison we present the same simulation performed using QuTiP, dotted orange line. As a final check, one can also compute the quantum fidelity between the steady state of the system and the vacuum state and verify it to be 100%.

We conclude this section on open quantum dynamics with an example of special interest to quantum optics, namely squeezed states. These states are sensitive to damping and photon-loss decoherence [16], as we now verify using QuGIT. We simulate an analogous time evolution to the previous example, now with an initial squeezed-coherent state $|\alpha = 2,\, r = 1.2\rangle$ subject to a damping constant $\gamma = 2\pi \times 0.1$. On the top plot of Figure 3 we observe a dynamics that initially resembles the unitary evolution of the field quadrature described in the previous section. However, damping quickly acts on the field causing it to approach the vacuum. Note the reduced variance associated to squeezing of the quadrature is also affected, evidenced by a *smoothing* over time. The bottom plot of Figure 3 further illustrates this effect by displaying the time evolution of the squeezing degree, defined as the ratio of the squeezed to anti-squeezed field quadrature variances. The squeezing degree starts near zero for a squeezed state, and gradually evolves to unity as squeezing is degraded by the amplitude-damping dynamics. We numerically verify that in the steady state, the squeezing degree approaches 1.

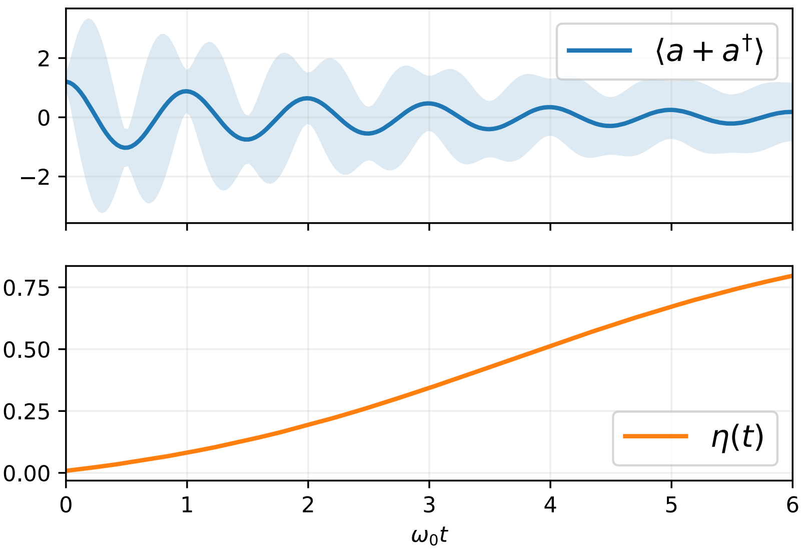

Figure 3: Damped time evolution of a squeezed state visualized in terms of time-dependent field quadrature $\langle X(t) \rangle = \langle a^\dagger + a \rangle$ (Top) and squeezing degree $\eta(t)$ (Bottom). Initial squeezed state is a squeezed-coherent state with $\alpha = 2$ and $r = 1.2$. Parameters for the dynamics are: $\omega_0 = 2\pi$ and $\gamma = 0.1 \times \omega_0$.

4.3 Mode entanglement and displacement detection

A number of experiments in quantum networks use the so-called sources of heralded single photon entanglement [17, 18], which consists in producing single photon states and subsequently delocalizing these over different spatial modes using beam-splitters (BS). Detection of single photon entanglement and more generally of *mode-entanglement* can be achieved through the use of displacement-based detection schemes [19], which allow for violations of Bell inequalities [20] as well as quantum communication protocols [21]. Displacement-based detectors are Gaussian operations and as such QuGIT can be used to

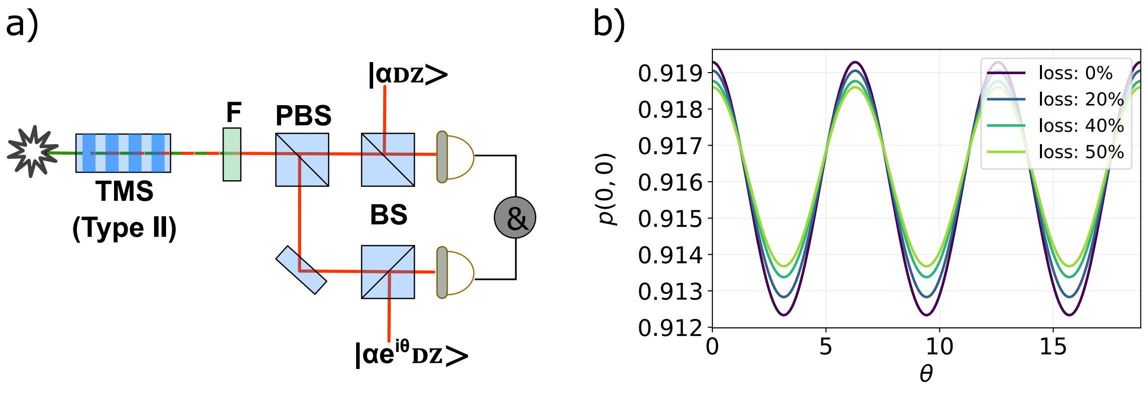

Figure 4: a) Typical setup from single-photon entanglement experiments. b) Oscillations in photon-counting statistics after a displacement-based detection scheme, calculated using QuGIT. Losses are modelled through a BS interaction with ancillary vacuum modes; legend indicates transmission of the setup.

explore the physics of these quantum communication experiments.

Figure 4a) shows a typical setup for a mode-entanglement experiment. A coherent source pumps a second-order nonlinear crystal (which can be placed inside a cavity as in the case of an optical parametric oscillator), producing Type II downconverted photon pairs. The statistics of the generated state is that of a two-mode squeezed state (TMS). The pump laser is filtered (F), and the modes in the TMS are spatially split using a polarizing beam-splitter (PBS). Each mode is subsequently input into unbalanced beam-splitters and mixed with local oscillators with a relative phase reference of $e^{i\theta}$. The joint probability of photon counts is detected using photodiodes and coincidence logic. In this setup, the joint photon number probability after displacement-based detection can undergo oscillations as a function of the relative phase θ. In particular, the probability of detecting zero photons at each of the photodiodes is given by,

$$p(0,0) = \mathrm{Tr}\left(D_a(\alpha)D_b(\alpha e^{i\theta})|\Psi_{TMS}\rangle\langle\Psi_{TMS}|D_a^\dagger(\alpha)D_b^\dagger(\alpha e^{i\theta})|0\rangle\langle 0|\right) \tag{13}$$

where $|\Psi_{TMS}\rangle$ is the TMS state and D_a, D_b are displacement operators in modes a and b, respectively. Figure 4b) displays a numerical plot of $p(0,0)$ calculated using QuGIT, taking into account losses in the interferometer through BS interactions with ancilla modes. This simple example demonstrates the power of the toolbox in simulating quantum interference experiments performed using Gaussian operations plus photon-counting, such as required in boson sampling [8] and photonic quantum computing [9].

4.4 Conditional dynamics

QuGIT is capable of solving general stochastic conditional dynamics. To demonstrate this capability, we consider the example of an Optical Parametric Oscillator (OPO) generating single mode squeezing via the Hamiltonian outlined in Section 6.1 of Ref [13], given by

$$\hat{H} = \frac{\chi}{2}\left(\hat{x}\hat{p} + \hat{p}\hat{x}\right) , \tag{14}$$

where χ is the squeezing rate.

The OPO mode can interact with an environment, modelled as a thermal bath with density matrix $V_B = (2n_{th}+1)\mathbb{1}$. For simplicity, we will consider $n_{th} = 0$. Interaction between the bath and the OPO mode is modelled by the interaction matrix $C = \sqrt{\gamma}\mathbb{1}$, where gamma is a damping constant. The associated drift and diffusion matrices are respectively given by [13],

$$A = \begin{bmatrix} -\chi - \gamma/2 & 0 \\ 0 & \chi - \gamma/2 \end{bmatrix} \ , \quad D = \gamma(2n_{th} + 1)\mathbb{1} \, . \tag{15}$$

This defines a stable unconditional dynamics whenever $\chi < \gamma/2$, and one can show that the steady state squeezing degree (defined as the ratio of squeezed to anti-squeezed quadratures) reads [13]

$$\eta_{\text{uncond}} = \frac{1 - 2\chi/\gamma}{1 + 2\chi/\gamma} \, . \tag{16}$$

We now consider the OPO mode is continuously monitored via projections onto the covariance matrix $V_m = \text{diag}(s, 1/s)$. We take the case of homodyne measurement of the $\hat{x}$ quadrature ($s \to \infty$). In this case, the steady state acquires a squeezing degree given by

$$\eta_{\text{cond.}} = \left(\frac{\gamma - 2\chi}{\gamma} \right)^2 \tag{17}$$

Figure 5 shows the time evolution of the squeezing degree for $\chi = \gamma/3$ for both the unconditional and conditional dynamics, together with their corresponding values in the steady state predicted by the theory. Observe that continuously monitoring the system enhances the steady state squeezing of the OPO mode.

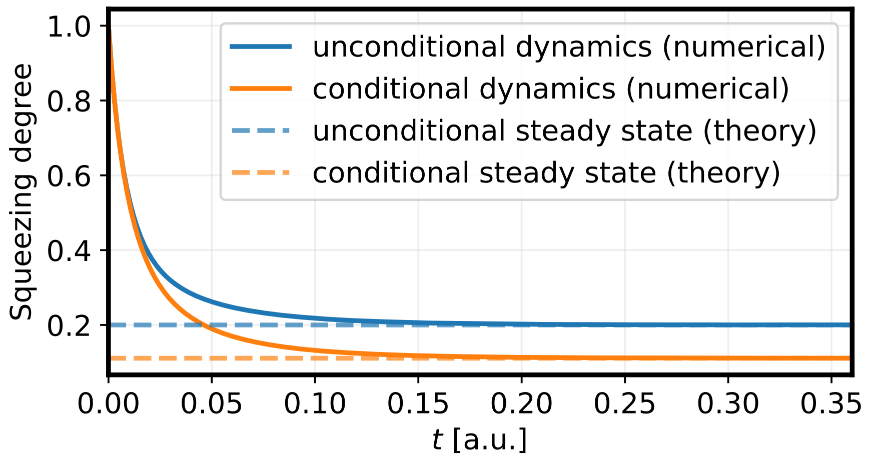

Figure 5: Unconditional and conditional dynamics of the squeezing degree of an OPO. In the conditional case, the OPO mode is subject to continuous. homodyne detection of the $\hat{x}$ quadrature. $N_{traj} = 100$ quantum trajectories were considered for the quantum Monte-Carlo simulation.

This example illustrates the use of QuGIT to the study of conditional stochastic quantum evolution. We close this example highlighting that the above-described stochastic evolution tools can be further generalized to include feedback and optimal control according to the needs of the studied system, as for example in [22].

4.5 Random Gaussian circuits

We now turn to the problem of random quantum circuits, for which the dynamics of entanglement growth has recently gained increased attention [23, 24]. As we next demonstrate, QuGIT can efficiently generate and simulate random unitary circuits and quantify entanglement growth under increasing numbers of gates and modes.

We generate circuits by randomly selecting a gate for each mode/pair of modes of a N-mode Gaussian state for T turns, winding up with a quantum circuit made of NT elementary gates. Elementary gates are taken from a list containing the identity, rotation, displacement, squeezing, two-mode squeezing, and beam splitter operations. When necessary, the parameter associated to each gate (such as the rotation angle for the rotation operator) are uniformly chosen from a pre-established range. The random circuit is then applied to a tensor product of N vacuum states, typically producing a highly entangled random state.

We think of the modes as organized in a discrete 1D lattice with each mode located at integer values. Define the spatial-dependent von Neumann entropy entropy $S(x)$ as the entropy of a bipartition of the system at mode x. $S(x)$ is used as a spatial-sensitive measure of entanglement, and the dynmanics of $S(x)$ under varying number of gates quantifies entanglement growth. Given large numbers of modes and elementary gates, QuGIT has a solid advantage in simulating examples of this type with respect to QuTIP, where trucation of Hilbert space would severely limit the simulation.

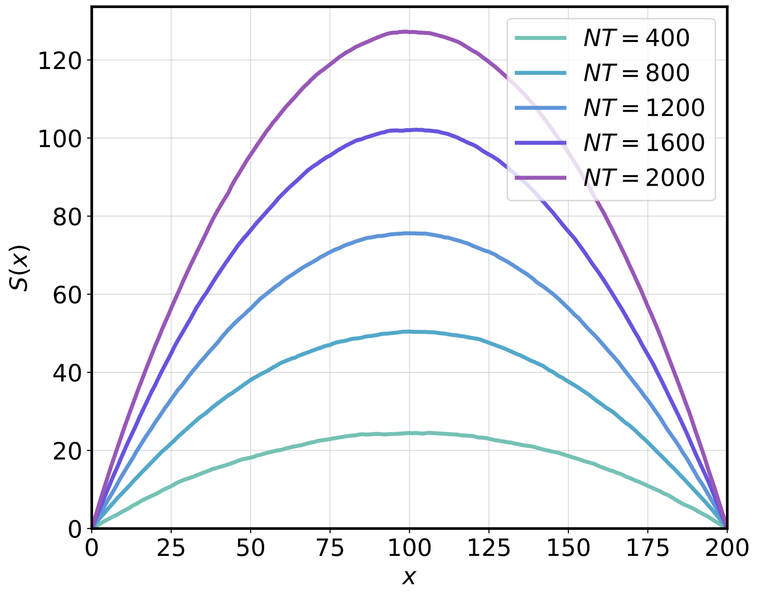

Figure 6: The spatial von Neumann entropy $S(x)$ for random Gaussian states comprised of $N = 200$ modes, obtained by applying random unitary quantum circuits to an initial vacuum state. Each circuit contains NT gates. For each curve shown we average over an ensemble of 100 realizations.

Figure 6 displays the simulation result for $N = 200$ modes with five increasing values for T. Due to the statistical nature of this example ensembles of 100 realizations for each value of T were considered, with each curve shown expressing the mean value of $S(x)$ over all realizations. The high value of N was selected to experiment with the performance of the toolbox. In order to produce the plot in Figure 6, 500 random circuits were simulated, with an overall number of $600,000$ gates. The von Neumann entropy had to be calculated approximately $100,000$ times. The final result was obtained in 3 hours with a simple average notebook, 1.8 GHz dual-core 8 GB laptop. The outcome is consistent with simulations obtained with qubits and quantum circuits composed of Clifford gates [23] providing a continuous-variable analogue for known results of entanglement growth under random unitaries acting on finite-dimensional Hilbert spaces [25].

5 Performance

We now turn our attention to the performance of QuGIT. In general, the number of modes for the system in question heavily determines computation time. This can be readily observed when considering the time evolution of systems with a large numbers of interacting modes. Consider the problem of calculating the open quantum dynamics of a system of N modes in which all modes interact with independent thermal baths and with one another such that all mode-quadratures are coupled, i.e. the matrix $H(t)$ in Equation (4) has no vanishing elements. In this case the unitary dynamics is governed by the Hamiltonian

$$\hat{H} = \hbar \sum_{j=1}^{N} \omega_j \hat{b}_j^\dagger \hat{b}_j + \sum_{\substack{j=1 \\ k\neq j}}^{N} \alpha_{jk}\hat{x}_j\hat{x}_k + \beta_{jk}\hat{p}_j\hat{p}_k + \gamma_{jk}\hat{x}_j\hat{p}_k + \delta_{jk}\hat{p}_j\hat{x}_k \,, \tag{18}$$

while interaction with the environment is modelled as an independent thermal bath at finite temperatures for each particle, inducing quantum Brownian motion on each mode. Here, we refer to Ref [26] for the equations of motion dictating this Gaussian-preserving open system dynamics.

Figure 7 shows the computation time needed to calculate the time evolution of the system using QuGIT as a function of the number of interacting modes, averaged over 50 realizations. Note the polynomial scaling with the number of modes. It is instructive to compare computation times for QuTiP versus QuGIT. While QuGIT could exactly simulate a 50-mode Gaussian dynamics in 8 s, QuTiP takes 25 s to simulate a similar 4-mode Gaussian dynamics with a Hilbert space of dimension of 5. It is expected that in the ideal case the toolbox performance scales as N^2, following the growth in size of the covariance matrix for Gaussian states. Figure 7 includes a quadratic fit to the computation time displaying good agreement with the data for $N \leq 50$.

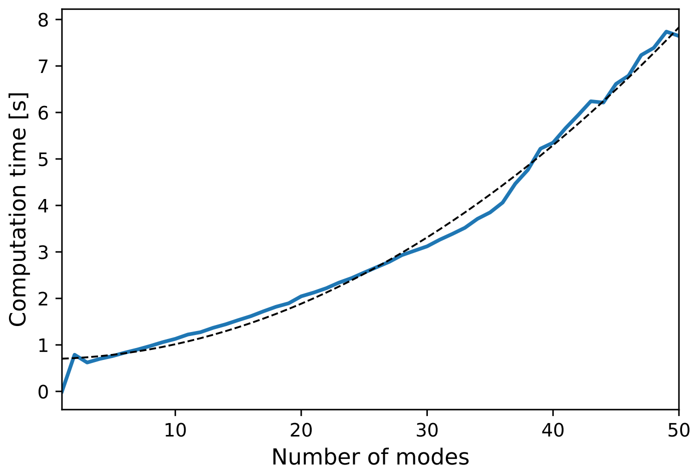

Figure 7: Blue: Computation time for a generic unconditional dynamics, averaged over 50 realizations, as a function of the number of modes. Each simulation consisted of a time evolution of $10,000$ time steps spanning 5 complete cycles of the harmonic oscillators. Black dashed line: quadratic fit. Parameters used were $\omega_j = 2\pi \times 305\,\mathrm{kHz}$, and $\alpha_{jk} = \beta_{jk} = \gamma_{jk} = \delta_{jk} \sim \omega_j/3$.

As a second example of performance, we revisit Section 4.5 and study the computation time of QuGIT for random circuits. The total computation time including state initialization, application of the random quantum circuit and calculation of the spatial-dependent von Neumann entropy is shown in Figure 8 as a function of number of modes and gates.

Observe that the number of gates per mode has a more significant impact on the performance over the number of modes. For the most challenging case of a total of $NT = 720$ gates applied to $N = 40$ modes the simulation completes in less than a third of a second on the average laptop computer. All performance tests were carried out on a 2.80 GHz quad-core, 16 GB laptop running Windows.

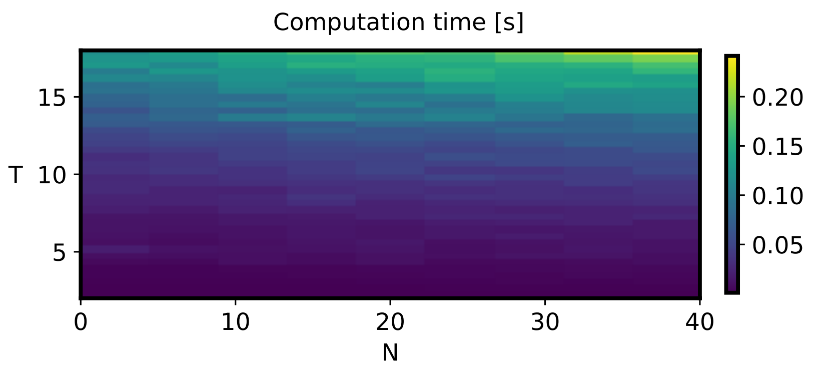

Figure 8: Computation time for random Gaussian circuits as a function of the number of modes N and number of gates per mode T.

6 Conclusion

In this work, we report an open-source numerical Python toolbox to simulate Gaussian quantum states and operations. By directly using the symplectic representation of Gaussian states, the toolbox can exactly simulate multi-mode systems without the need for truncated Hilbert spaces or other approximations. The resources needed to store and manipulate quantum states in QuGIT is greatly reduced prompting the use of the toolbox to simulate systems with many constituents.

Various numerical examples were carried out to exhibit the toolbox versatility and robustness for a variety of Gaussian quantum systems and dynamics. The performance of the toolbox was considered. In conclusion, while the quantum information community benefits from excellent packages for simulating quantum systems such as QuTiP, we hope QuGIT will add to that list, providing complementary solutions when it comes to the simulation of Gaussian continuous variable systems.

Citation guideline

If you make use of QuGIT in your research please add a citation to this paper and acknowledge using:

This work makes use of the QuGIT toolbox.

Acknowledgements

We thank Pedro Paraguassú, Bruno Suassuna and Igor Califrer for helpful discussions and testing of the toolbox. I.B. thanks Dr. Dan Marchesin for the numerous discussions on efficient programming. This work was financed in part by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq). I. B. thanks the support received by the FAPERJ Scholarship No. E-26/200.270/2020 and CNPq Scholarship No. 140279/2021-0. T.G. thanks the support received by the FAPERJ Scholarship No. E-26/202.830/2019. D. T. thanks the support received by the CNPq Scholarship No. 132606/2020-8.

References

[1] J.R. Johansson, P.D. Nation, and Franco Nori. QuTiP: An open-source python framework for the dynamics of open quantum systems. *Computer Physics Communications*, 183(8):1760–1772, August 2012.

[2] Sebastian Krämer, David Plankensteiner, Laurin Ostermann, and Helmut Ritsch. QuantumOptics.jl: A julia framework for simulating open quantum systems. *Computer Physics Communications*, 227:109–116, June 2018.

[3] Sze M Tan. A computational toolbox for quantum and atomic optics. *Journal of Optics B: Quantum and Semiclassical Optics*, 1(4):424–432, August 1999.

[4] J.R. Johansson, P.D. Nation, and Franco Nori. QuTiP 2: A python framework for the dynamics of open quantum systems. *Computer Physics Communications*, 184(4):1234–1240, April 2013.

[5] Markus Aspelmeyer, Tobias J. Kippenberg, and Florian Marquardt. Cavity optomechanics. *Rev. Mod. Phys.*, 86:1391–1452, Dec 2014.

[6] Uros Delic. *Cavity cooling by coherent scattering of a levitated nanosphere in vacuum.* PhD thesis, uniwien, 2019.

[7] Andrés de los Ríos Sommer, Nadine Meyer, and Romain Quidant. Strong optomechanical coupling at room temperature by coherent scattering. *Nature Communications*, 12(1), January 2021.

[8] Han-Sen Zhong, Hui Wang, Yu-Hao Deng, Ming-Cheng Chen, Li-Chao Peng, Yi-Han Luo, Jian Qin, Dian Wu, Xing Ding, Yi Hu, Peng Hu, Xiao-Yan Yang, Wei-Jun Zhang, Hao Li, Yuxuan Li, Xiao Jiang, Lin Gan, Guangwen Yang, Lixing You, Zhen Wang, Li Li, Nai-Le Liu, Chao-Yang Lu, and Jian-Wei Pan. Quantum computational advantage using photons. *Science*, 370(6523):1460–1463, December 2020.

[9] Sara Bartolucci, Patrick M. Birchall, Mercedes Gimeno-Segovia, Eric Johnston, Konrad Kieling, Mihir Pant, Terry Rudolph, Jake Smith, Chris Sparrow, and Mihai D. Vidrighin. Creation of entangled photonic states using linear optics, 2021.

[10] Christian Weedbrook, Stefano Pirandola, Raúl García-Patrón, Nicolas J. Cerf, Timothy C. Ralph, Jeffrey H. Shapiro, and Seth Lloyd. Gaussian quantum information. *Rev. Mod. Phys.*, 84:621–669, May 2012.

[11] Alessio Serafini. *Quantum Continuous Variables: A Primer of Theoretical Methods.* CRC Press, jun 2017.

[12] Daniel Grimmer, Eric Brown, Achim Kempf, Robert B Mann, and Eduardo Martín-Martínez. A classification of open gaussian dynamics. *Journal of Physics A: Mathematical and Theoretical*, 51(24):245301, May 2018.

[13] Marco G. Genoni, Ludovico Lami, and Alessio Serafini. Conditional and unconditional gaussian quantum dynamics. *Contemporary Physics*, 57(3):331–349, 2016.

[14] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[15] Marlan O. Scully and M. Suhail Zubairy. *Quantum Optics*. Cambridge University Press, September 1997.

[16] A. I. Lvovsky. Squeezed light, 2016.

[17] F. Monteiro, V. Caprara Vivoli, T. Guerreiro, A. Martin, J.-D. Bancal, H. Zbinden, R. T. Thew, and N. Sangouard. Revealing genuine optical-path entanglement. *Physical Review Letters*, 114(17), May 2015.

[18] Peter C. Humphreys, Norbert Kalb, Jaco P. J. Morits, Raymond N. Schouten, Raymond F. L. Vermeulen, Daniel J. Twitchen, Matthew Markham, and Ronald Hanson. Deterministic delivery of remote entanglement on a quantum network. *Nature*, 558(7709):268–273, June 2018.

[19] N. Bruno, A. Martin, P. Sekatski, N. Sangouard, R. T. Thew, and N. Gisin. Displacement of entanglement back and forth between the micro and macro domains. *Nature Physics*, 9(9):545–548, July 2013.

[20] T. Guerreiro, F. Monteiro, A. Martin, J. B. Brask, T. Vértesi, B. Korzh, M. Caloz, F. Bussières, V. B. Verma, A. E. Lita, R. P. Mirin, S. W. Nam, F. Marsilli, M. D. Shaw, N. Gisin, N. Brunner, H. Zbinden, and R. T. Thew. Demonstration of einstein-podolsky-rosen steering using single-photon path entanglement and displacement-based detection. *Physical Review Letters*, 117(7), August 2016.

[21] F Monteiro, E Verbanis, V Caprara Vivoli, A Martin, N Gisin, H Zbinden, and R T Thew. Heralded amplification of path entangled quantum states. *Quantum Science and Technology*, 2(2):024008, May 2017.

[22] Lorenzo Magrini, Philipp Rosenzweig, Constanze Bach, Andreas Deutschmann-Olek, Sebastian G. Hofer, Sungkun Hong, Nikolai Kiesel, Andreas Kugi, and Markus Aspelmeyer. Real-time optimal quantum control of mechanical motion at room temperature, 2021.

[23] A. Nahum, J. Ruhman, S. Vijay, and J. Haah. Quantum entanglement growth under random unitary dynamics. *Physical Review X*, 7(3), jul 2017.

[24] Yaodong Li and Matthew P. A. Fisher. Statistical mechanics of quantum error correcting codes. *Physical Review B*, 103(10), Mar 2021.

[25] Don N. Page. Average entropy of a subsystem. *Physical Review Letters*, 71(9):1291–1294, Aug 1993.

[26] Vittorio Giovannetti and David Vitali. Phase-noise measurement in a cavity with a movable mirror undergoing quantum brownian motion. *Phys. Rev. A*, 63:023812, Jan 2001.

[27] V. V. Dodonov, O. V. Man'ko, and V. I. Man'ko. Multidimensional hermite polynomials and photon distribution for polymode mixed light. *Phys. Rev. A*, 50:813–817, Jul 1994.

[28] Ond řej Černotík and Radim Filip. Strong mechanical squeezing for a levitated particle by coherent scattering. *Phys. Rev. Research*, 2:013052, Jan 2020.

[29] Leonardo Banchi, Samuel L. Braunstein, and Stefano Pirandola. Quantum fidelity for arbitrary gaussian states. *Phys. Rev. Lett.*, 115:260501, Dec 2015.

[30] Jianwei Xu. Quantifying coherence of gaussian states. *Phys. Rev. A*, 93:032111, Mar 2016.

[31] Giuseppe Vallone, Gianfranco Cariolaro, and Gianfranco Pierobon. Means and covariances of photon numbers in multimode gaussian states. *Phys. Rev. A*, 99:023817, Feb 2019.

[32] Jinglei Zhang. *Quantum measurement and preparation of Gaussian states.* PhD thesis, Department of Physics and Astronomy, Aarhus University, 2018.

A Toolbox methods

Table 2: gaussian_state class' methods

Method	Description	Reference
displace	Applies a displacement operator on a single mode Gaussian state	[10]
squeeze	Applies a squeezing operator on a single mode Gaussian state	[10]
rotate/phase	Applies a rotation operator on a single mode Gaussian state	[10]
beam_splitter	Applies a beam splitter operator on a two mode Gaussian state	[10]
two_mode_squeezing	Applies a two mode squeezing operator on a two mode Gaussian state	[10]
apply_unitary	Applies a generic unitary operator given its symplectic representation	[10]
loss_ancilla	Applies a beam splitter operator between a desired mode and an ancilla	—
tensor_product	Tensor product of two Gaussian states	[11]
partial_trace	Partial trace over some modes	[11]
only_modes	Partial trace over all but some modes	[11]
matrix_element_coherent_basis	Calculates the density matrix elements on coherent state basis	[27]
matrix_element_number_basis	Calculates the density matrix elements on number states basis	[27]
purity	Purity	[10]
symplectic_eigenvalues	Symplectic eigenvalues of the covariance matrix	[10]
von_Neumann_Entropy	von Neumann entropy	[10]
mutual_information	Mutual information	—
squeezing_degree	Ratio of the variance of the squeezed and antisqueezed quadratures	[28]
fidelity	Quantum Fidelity between the two Gaussian states	[29]
coherence	Coherence of a multipartite Gaussian state	[30]
occupation_number	Occupation number for each mode of the Gaussian state	—
number_operator_moments	Calculates means vector and covariance matrix of number operator	[31]
number_statistics	Calculates the number distribution of the Gaussian state	[27]
wigner	Wigner function over a 2D grid for a single mode Gaussian state	[10]
q_function	Hussimi Q-function over a 2D grid for a single mode Gaussian state	[27]
logarithmic_negativity	Logarithmic negativity for a bipartition of a Gaussian state	[10]
measurement_general	Conditional state after a partial Gaussian measurement	[32]
measurement_homodyne	Conditional state after a partial homodyne measurement	[32]
measurement_general	Conditional state after a partial heterodyne measurement	[32]
print	Prints the Gaussian state on the console log	—
copy	Creates an identical copy	—

Table 3: Methods of the gaussian_dynamics class

Method	Description
unconditional_dynamics	Calculates the time evolution of an initial state following an unconditional dynamics
conditional_dynamics	Calculates the time evolution of an initial state following a conditional dynamics
steady_state	Calculates the steady state of an unconditional dynamics
semi_classical	Calculates the semi-classical time evolution of the mean quadratures, Monte Carlo method

B QuGIT example codes

We now present the codes used on the examples of this work to illustrate the capabilities of the toolbox. For simplicity, the lines of code associated with plotting have been omitted.

B.1 Unitary field quadrature dynamics

```
import numpy as np
import quantum_gaussian_toolbox as qgt

##### Parameters for the dynamics
omega    = 2*np.pi                        # Natural frequency [Hz]
t = np.linspace(0, 2/omega, int(200))  # Timestamps for simulation

A = np.array([[    0   ,  +omega ],
              [ -omega ,     0   ]])   # Drift matrix
D = np.diag([0, 0])                     # Diffusion matrix
N = np.zeros((2,1))                     # Driving vector

##### Simulating coherent state time evolution
initial_0 = qgt.coherent(alpha=2)      # Initial coherent state
simulation_0 = qgt.gaussian_dynamics(A, D, N, initial_0)
states_0 = simulation_0.unconditional_dynamics(t) # Simulate

##### Simulating coherent squeezed state time evolution
initial_1 = initial_0.copy()          # Copy coherent state
initial_1.squeeze(r=1.2)              # Apply squeezing operator

simulation_1 = qgt.gaussian_dynamics(A, D, N, initial_1)
states_1 = simulation_1.unconditional_dynamics(t) # Simulate

##### Retrieve information from time evolved states
mean_x_1 = np.zeros(len(t))  # List to store mean quadrature, 1st simulation
var_x_1  = np.zeros(len(t))  # List to store variance,        1st simulation

mean_x_0 = np.zeros(len(t))  # List to store mean quadrature, 2nd simulation
var_x_0  = np.zeros(len(t))  # List to store variance,        2nd simulation

for i in range(len(t)):                # Loop through time-evolved states
    mean_x_0[i] = states_0[i].R[0]
    var_x_0[i]  = states_0[i].V[0,0]

    mean_x_1[i] = states_1[i].R[0]
    var_x_1[i]  = states_1[i].V[0,0]
```

B.2 Damped harmonic oscillator code

Number operator moments and Wigner function dynamics for coherent state

```
import numpy as np
import quantum_gaussian_toolbox as qgt

##### Parameters
omega = 2*np.pi;                     # Particle natural frequency
gamma = 2*np.pi*0.3;                 # Damping constant
t = np.linspace(0, 3.5*2*np.pi/omega, int(200))  # Timestamps for simulation

x = np.linspace(-6,6,200)
p = np.linspace(-6,6,200)
X, P = np.meshgrid(x, p);            # Meshgrid for phase-space

##### Matrices defining the dynamics
A = np.array([[-gamma/2,  +omega],
              [-omega  ,-gamma/2]]) # Drift matrix
D = np.diag([gamma, gamma]);         # Diffusion matrix
N = np.zeros((2,1));                 # Driving vector

##### Simulation
initial = qgt.coherent(alpha=2)      # Initial state

simulation = qgt.gaussian_dynamics(A, D, N, initial) # Time evolution instance
states = simulation.unconditional_dynamics(t)        # Simulate

##### Retrive information from time evolved states
n_bar = np.zeros(len(t))             # List to store occupation numbers
n_var = np.zeros(len(t))             # List to store variance of number operator
W = []                               # List to store Wigner functions

for i in range(len(t)):              # Loop through time-evolved states
    n_bar[i], n_var[i] = states[i].number_operator_moments()
    W.append(states[i].wigner(X, P))

ss = simulation.steady_state()       # Steady state of the system
F = qgt.fidelity(ss, qgt.vacuum())   # Fidelity with vacuum state
```

Quadrature and squeezing degree dynamics for coherent-squeezed state

```
import numpy as np
import quantum_gaussian_toolbox as qgt

##### Parameters
omega = 2*np.pi;                   # Particle natural frequency
gamma = 2*np.pi*0.1;               # Damping constant
t = np.linspace(0, 6, int(200))    # Timestamps for simulation

##### Matrices defining the dynamics
A = np.array([[-gamma/2,  +omega],
              [-omega   ,-gamma/2]]) # Drift matrix
D = np.diag([gamma, gamma]);       # Diffusion matrix
N = np.zeros((2,1));               # Driving vector

##### Simulation
initial = qgt.coherent(alpha=2)    # Initial state
initial.squeeze(r=1.2)

simulation = qgt.gaussian_dynamics(A, D, N, initial) # Time evolution instance
states = simulation.unconditional_dynamics(t)        # Simulate

##### Retrive information from time evolved states
squeezing_number = np.zeros(len(t)) # List to store occupation numbers

mean_x = np.zeros(len(t))          # List to store mean quadrature
var_x = np.zeros(len(t))           # List to store quadrature variance

for i in range(len(t)):            # Loop through time-evolved states
    squeezing_number[i] = states[i].squeezing_degree()[0]
    mean_x[i] = states[i].R[0]
    var_x[i] = states[i].V[0,0]

ss = simulation.steady_state()     # Steady state of the system
print(ss.squezzing_degree()[0])  # Fidelity with vacuum state
```

B.3 Mode entanglement and displacement detection

```
import numpy as np
import quantum_gaussian_toolbox as qgt

phi = np.linspace(0, 6*np.pi, int(200)) # Relative phases
tau_list = np.array([1, 0.8, 0.6, 0.5])

for j in range(len(tau_list)):

    bipartite = qgt.vacuum(N=2)         # Initial state
    bipartite.two_mode_squeezing(r=0.4) # Apply two-mode squeezing operator

    bipartite.loss_ancilla(0, tau_list[j]) # Apply BS with ancilla mode

    Fidelity = np.zeros(len(phi))          # List of fidelities

    for i in range(len(phi)):
        coherent1  = qgt.coherent(alpha = 0.1)
        coherent2  = qgt.coherent(alpha = 0.1 * np.exp(1j*phi[i]))
        coherent12 = qgt.tensor_product([coherent1, coherent2])

        Fidelity[i] = qgt.fidelity(coherent12, bipartite)
```

B.4 Conditional dynamics

```
##### Paramters
gamma = 2*np.pi*10
nbar_env = 0
chi    = gamma/3

A = np.block([[ -chi - gamma/2 ,       0         ],
              [       0         ,  chi - gamma/2 ]]) # Drift matrix

D = gamma*(2*nbar_env+1)*np.eye(2)                    # Diffusion matrix
N = np.zeros((2,1))                                   # Driving vector

initial_state = qgt.coherent(alpha=3) # Initial state
t = np.linspace(0, 0.36, 2000)        # Timestamps for simulation

##### Unconditional dynamics
simulation = qgt.gaussian_dynamics(A, D, N, initial_state)  # Simulation instance

unconditional_states = simulation.unconditional_dynamics(t) # Simulate

unconditional_sq = np.zeros(len(t))   # Calculate time evolved squeezing degree
for i in range(N_time):
    unconditional_sq_temp = qgt.squeezing_degree(unconditional_states[i])
    unconditional_sq[i] = unconditional_sq_temp[0]

###### Conditional dynamics
C = np.diag([np.sqrt(gamma), np.sqrt(gamma)]) # System-bath interaction
rho_b =  qgt.thermal(nbar_env)                # Bath's state

conditional_states = simulation.conditional_dynamics(t, N_ensemble=100,
C_int = C, rho_bath =  rho_b, s_list=[1e-5], phi_list=[np.pi/2])

conditional_sq = np.zeros(len(t))
for i in range(N_time):
    conditinal_sq_temp = qgt.squeezing_degree(conditional_states[i])
    conditional_sq[i] = conditinal_sq_temp[0]

###### Analytical predictions
a = 1 / (1 + chi/(gamma/2))
b = 1 / (1 - chi/(gamma/2))
steady_unconditional = (a/b) * np.ones(N_time) # Steady state unconditional

##### Analytical prediction for conditional dynamics
c = (gamma - 2*chi) / gamma
d = 1/c
steady_conditional = (c/d) * np.ones(N_time) # Steady state conditional
```

B.5 Random Gaussian circuits

For this example, we focus on the toolbox usage and do not show the code that chooses a random Gaussian circuit and applies it to an initial state. The circuits are composed of: displacement, single-mode squeezing, rotation, beam-splitter and two-mode squeezing operators, whose parameters are chosen at random. The choice of the circuit is done by the method `random_circuit` and its application on a tensor product of vacuum states is carried out by `apply_circuit`. The code for these methods can be found at:
https://github.com/IgorBrandao42/Quantum-Gaussian-Information-Toolbox.

```
import numpy as np
import random

import gaussian_toolbox as qgt

N = 200            # Number of modes for the initial state
mean_alpha = 0.1   # Mean real parameter for the displacement operator
std_alpha  = 0.01  # Standard deviation for the displacement operator's parameter
T = [2,4,6,8,10]   # Number of gates per mode to apply to the initial state
loops = 100        # Number of iterations to find the mean entropy

s_x = []           # List to store each mean von Neumann entropy
for i in range(len(T)):      # For each gate

    s_x_mean = np.zeros(N+1) # Mean entropy for this number of gates applied

    for j in range(loops):   # Repeat to find the average entropy
        # Generate random Gaussian circuit
        circuit = random_circuit(N,mean_alpha,std_alpha,T[i])

        initial_state = qgt.vacuum(N)  # Initial state

        # Apply random circuit to initial state
        final_state = apply_circuit(initial_state,circuit)

        s_x_temp = np.zeros(N+1)   # Temporary variable

        for k in range(1,N+1):     # Loop through each mode to generate bipartitions

            modes = list(range(k)) # Get indexes to first k modes

            # Get bipartition up to k-th mode
            partition = qgt.only_modes(final_state,modes)

            s_x_temp[k] = qgt.von_Neumann_Entropy(partition) # Calculate its entropy

        s_x_mean = s_x_mean+s_x_temp # Add current entropy to the sum

    s_x_mean = s_x_mean/loops        # Get average entropy
    s_x.append(s_x_mean)             # Append it to the list of mean entropies
```