A Quadratic Speedup in the Optimization of Noisy Quantum Optical Circuits

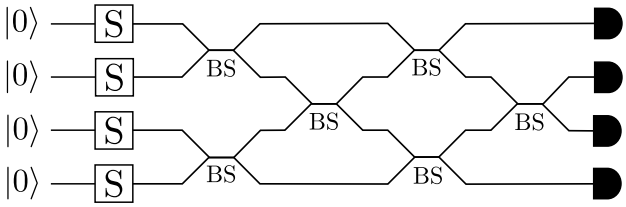
Robbe De Prins[1], Yuan Yao[2], Anuj Apte[3,4], and Filippo M. Miatto[2,3]

[1]Photonics Research Group, INTEC, Ghent University – imec, Sint-Pietersnieuwstraat 41, 9000 Ghent, Belgium
[2]Télécom Paris and Institut Polytechnique de Paris, LTCI, 20 Place Marguerite Perey, 91120 Palaiseau, France
[3]Xanadu, Toronto, ON, M5G 2C8, Canada
[4]Kadanoff Center for Theoretical Physics & Enrico Fermi Institute, Department of Physics, University of Chicago, Chicago, IL 60637

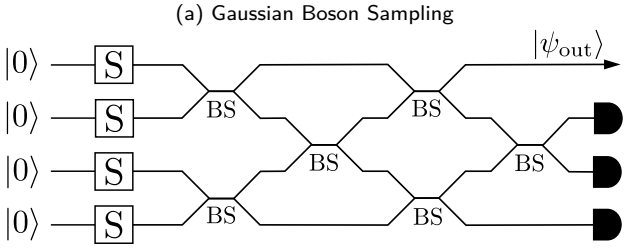
Linear optical quantum circuits with photon number resolving (PNR) detectors are used for both Gaussian Boson Sampling (GBS) and for the preparation of non-Gaussian states such as Gottesman-Kitaev-Preskill (GKP), cat and NOON states. They are crucial in many schemes of quantum computing and quantum metrology. Classically optimizing circuits with PNR detectors is challenging due to their exponentially large Hilbert space, and quadratically more challenging in the presence of decoherence as state vectors are replaced by density matrices. To tackle this problem, we introduce a family of algorithms that calculate detection probabilities, conditional states (as well as their gradients with respect to circuit parametrizations) with a complexity that is comparable to the noiseless case. As a consequence we can simulate and optimize circuits with twice the number of modes as we could before, using the same resources. More precisely, for an M-mode noisy circuit with detected modes D and undetected modes U, the complexity of our algorithm is $O(M^2 \prod_{i\in U} C_i^2 \prod_{i\in D} C_i)$, rather than $O(M^2 \prod_{i\in D\cup U} C_i^2)$, where C_i is the Fock cutoff of mode i. As a particular case, our approach offers a full quadratic speedup for calculating detection probabilities, as in that case all modes are detected. Finally, these algorithms are implemented and ready to use in the open-source photonic optimization library `MrMustard` [29].

1 Introduction

Linear optical quantum circuits with photon number resolving (PNR) detectors are studied because of two main reasons. First of all, they are used to perform Gaussian Boson Sampling (GBS). In GBS, squeezed states are sent through an interferometer and subsequently detected by PNR detectors. An example of such a circuit is depicted in Fig. 1(a).



(a) Gaussian Boson Sampling



(b) Conditional non-Gaussian state generation (e.g. GKP, cat, NOON states, etc.)

Figure 1: Examples of linear optical quantum circuits with PNR detectors. Vacuum states are squeezed and sent through an interferometer. A subset of the modes is measured with PNR detectors.

GBS is a leading approach in pursuing quantum advantage [14, 17]. Moreover, several quantum algorithms based on GBS have been introduced [1–3, 6–8, 15, 16, 24], some of which rely on the ability to train the circuit parameters.

The second (and arguably more useful) application for circuits with PNR detectors is the generation of conditional non-Gaussian states. Examples of such states include Gottesman-Kitaev-Preskill (GKP) states, cat states, bosonic-code states, weak cubic phase states, ON states and NOON states [11, 21, 22, 25–28, 30]. These states are used in a wide range of applications, such as generating bosonic error correction codes, providing resource states for the implementation of non-Gaussian gates and quantum metrology. We emphasize the particular interest of GKP states [13] as they are one of the leading candidates for qubits in optical quantum computation [5].

Robbe De Prins: robbe.deprins@ugent.be
Yuan Yao: yuan.yao@telecom-paris.fr
Anuj Apte: apteanuj@uchicago.edu
Filippo M. Miatto: filippo@xanadu.ai

Fig. 1(b) depicts a circuit that can be used to generate non-Gaussian states. Depending on the PNR detection pattern, a certain state is generated. The probability distribution of all conditional states is governed by the circuit parameters. By training these parameters, we can increase the probability of generating certain non-Gaussian states of interest and their quality.

In this work we address these simulation and optimization tasks using the framework that we introduced in our previous work [18, 31]. This framework allows one to recursively calculate elements of the matrix representation of Gaussian operators in Fock space. Here, it provides us with the matrix elements that define the detection probabilities or the amplitudes of conditional states. Moreover, we can recursively calculate the gradients of these elements with respect to a circuit parametrization, which allows us to find the parameters that minimize a certain cost function using gradient descent.

In realistic settings, decoherence effects such as photon loss affect the output of quantum circuits. Consequently, we need to be able to include these effects into our simulations if we want them to be faithful and useful. This motivates us to carry out simulations using density matrices. Normally, swapping state vectors for density matrices would make tasks quadratically more demanding in terms of both memory and runtime. We will show that we can almost completely get around this quadratic increase by introducing an algorithm that allows us to apply the recurrence relations fewer times while still including the amplitudes of interest. The resulting algorithm works for circuits with in principle an arbitrary number of PNR detectors. We will show that the complexity of our algorithm is comparable to the complexity of the lossless case, as long as the number of detected modes is a large fraction of the total number of modes.

The paper is structured as follows. In Section 2 we recall our simulation and optimization framework [18, 31] and apply it to lossless circuits with PNR detectors (i.e. using state vectors). In Section 3 we extend the framework to density matrices. We do this for GBS circuits (such as Fig. 1(a)) in Section 3.1 and for conditional state generator circuits (such as Fig. 1(b)) in Section 3.2. In Section 4 we discuss the complexity of our algorithms. Section 4.1 gives numerical results for the memory requirements and speed. Section 4.2 gives a comparison with the state-of-the-art classical GBS simulation method.

Note that the construction of good ansätze for GKP generating circuits, as well as the construction of associated cost functions and target states is a separate research question in itself that we do not address in this manuscript.

2 Circuit optimization framework revisited

2.1 Representing Gaussian operators in Fock space

In Reference [18], it was shown that quantum optical circuits can be simulated by using a recurrence relation that calculates elements of the matrix representation of Gaussian operators (i.e. pure Gaussian states, mixed Gaussian states, Gaussian unitary transformations or Gaussian channels) in Fock space. We will denote such a matrix representation by $\boldsymbol{\mathcal{G}}$ and call its elements the 'Fock amplitudes' of a Gaussian operator.

As we are interested in calculating detection probabilities and possible conditional states here, we will consider $\boldsymbol{\mathcal{G}}$ to be the matrix representation of the multi-mode Gaussian state before the detectors. In other words, $\boldsymbol{\mathcal{G}}$ is either a state vector or density matrix in Fock space. We represent $\boldsymbol{\mathcal{G}}$ as a multidimensional array and refer to its total number of dimensions (i.e. indices) as D. Hence, a general Fock amplitude can be written as $\mathcal{G}_{\boldsymbol{k}}$, where $\boldsymbol{k}$ is an integer vector of length D. We will refer to $\boldsymbol{k}$ as a 'Fock index' of $\boldsymbol{\mathcal{G}}$. If $\boldsymbol{\mathcal{G}}$ is a state vector, we use the convention that every element of $\boldsymbol{k}$ corresponds to an optical mode. If $\boldsymbol{\mathcal{G}}$ is a density matrix, every *pair* of consecutive elements in $\boldsymbol{k}$ corresponds to an optical mode. For example, $\boldsymbol{k} = [m, n, p, q]$ is a general Fock index for a density matrix on 2 modes, where the indices m, n and p, q respectively correspond with the first and second mode. The expression for the Fock amplitudes using Dirac notation is $\mathcal{G}_{\boldsymbol{k}} = \mathcal{G}_{mnpq} = \langle m, p| \mathcal{G} |n, q\rangle$. For a general number of modes M, it follows that:

$$D = \begin{cases} M, & \text{if } \boldsymbol{\mathcal{G}} \text{ is a state vector,} \\ 2M, & \text{if } \boldsymbol{\mathcal{G}} \text{ is a density matrix.} \end{cases} \tag{1}$$

Fock amplitudes can now be calculated using the following recurrence relation:

$$\mathcal{G}_{\boldsymbol{k}+\mathbf{1}_i} = \frac{1}{\sqrt{k_i+1}} \left(\mathcal{G}_{\boldsymbol{k}} b_i + \sum_{l=1}^{D} \sqrt{k_l}\, \mathcal{G}_{\boldsymbol{k}-\mathbf{1}_l} A_{il} \right), \tag{2}$$

where $\mathbf{1}_i$ is a vector of all zeroes except for a single 1 in the i^{th} entry. Note that Fock indices that contain at least one negative value correspond to a zero Fock amplitude, as negative photon numbers are nonphysical. Hence, the sum over l may contain less than D terms.

The matrix $\boldsymbol{A}$ and vector $\boldsymbol{b}$ in Eq. (2) are complex-valued parameters (of size $D \times D$ and D respectively) that are easily acquired for a specific circuit as they derive from the parameters of the Gaussian representation. If $\boldsymbol{\mathcal{G}}$ is a density matrix $\boldsymbol{\rho}$ we recall the results derived in Reference [31] that relate $\boldsymbol{A}_{\boldsymbol{\rho}}$ and $\boldsymbol{b}_{\boldsymbol{\rho}}$ to its complex (i.e. in the $a/a^\dagger$ basis) covariance matrix $\boldsymbol{\sigma}$

and displacement vector $\boldsymbol{\mu}$:

$$\boldsymbol{A_\rho} = \boldsymbol{P}_M \boldsymbol{\sigma}_- \boldsymbol{\sigma}_+^{-1}, \tag{3}$$

$$\boldsymbol{b_\rho} = \left(\boldsymbol{\sigma}_+^{-1}\boldsymbol{\mu}\right)^* = \boldsymbol{P}_M \boldsymbol{\sigma}_+^{-1}\boldsymbol{\mu}, \tag{4}$$

where $\boldsymbol{\sigma}_\pm = \boldsymbol{\sigma} \pm \frac{1}{2}\mathbb{1}_{2M}$ and $\boldsymbol{P}_M = \begin{bmatrix} \boldsymbol{0}_M & \mathbb{1}_M \\ \mathbb{1}_M & \boldsymbol{0}_M \end{bmatrix}$. If $\boldsymbol{\mathcal{G}}$ is a state vector $\boldsymbol{\psi}$, then $\boldsymbol{A_\psi}$ and $\boldsymbol{b_\psi}$ can be obtained from:

$$\boldsymbol{A_\rho} = \boldsymbol{A}_{\boldsymbol{\psi}}^* \oplus \boldsymbol{A_\psi}, \tag{5}$$

$$\boldsymbol{b_\rho} = \boldsymbol{b}_{\boldsymbol{\psi}}^* \oplus \boldsymbol{b_\psi}. \tag{6}$$

Let us now define the 'weight' of a Fock index $\boldsymbol{k}$ as:

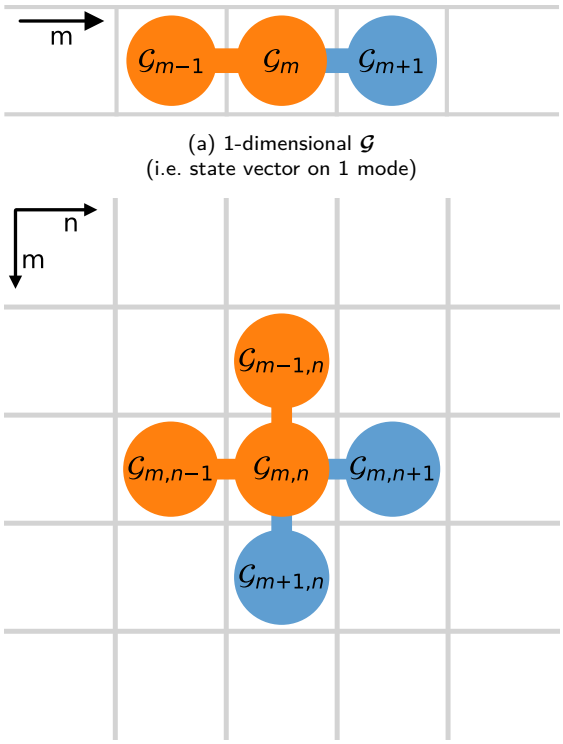
$$w = \sum_{i=1}^{D} k_i \ . \tag{7}$$

We see that Eq. (2) allows us to write D Fock amplitudes of weight $w+1$ as linear combinations of a single Fock amplitude of weight w and D Fock amplitudes of weight $w-1$. In order to refer to these different roles, we call 'read' the group of amplitudes of weight $w-1$ and 'write' the group of amplitudes of weight $w+1$ (to refer to the fact that D amplitudes need to be read from memory so that D new ones can be written to memory), and we refer to the single amplitude of weight w as the 'pivot'. Fig. 2 gives a schematic representation of Eq. (2) for the case where $\boldsymbol{\mathcal{G}}$ is 1-dimensional (i.e. for a state vector on one mode) and 2-dimensional (i.e. for a state vector on two modes or a density matrix on one mode). In this figure, the amplitudes marked in blue (write) are written as linear combinations of the orange ones (read+pivot). In general, a Fock index $\boldsymbol{k}$ marks a position in a D-dimensional 'Fock lattice'. Eq. (2) can thus be interpreted as a relation between $2D+1$ amplitudes that we can draw as a cross (or hypercross for higher dimensions). We can repeatedly reposition the hypercross in $\boldsymbol{\mathcal{G}}$ to calculate new Fock amplitudes under the condition that we already computed the read and pivot amplitudes.

2.2 State vector simulations

Let us now consider how we can apply the recurrence relation (that is, how we can move around the hypercross) to obtain the probabilities of PNR outcomes or the amplitudes of conditional states using the state vector formalism in a noiseless, lossless circuit. As the number of possible measurement results $\boldsymbol{n} = [n_1, n_2, ..., n_M]$ ($n_i \in [0, 1, ..., \infty]$) is in principle infinite, we limit ourselves to calculating the most probable ones such that the required resources for our simulation remain finite. We will consider the Fock amplitudes $\mathcal{G}_{\boldsymbol{k}}$ for all $\boldsymbol{k}$ of length M that satisfy the following boundary conditions:

$$\boldsymbol{0} \leq \boldsymbol{k} < \textit{cutoffs} \ . \tag{8}$$



(a) 1-dimensional $\boldsymbol{\mathcal{G}}$
(i.e. state vector on 1 mode)

(b) 2-dimensional $\boldsymbol{\mathcal{G}}$
(i.e. state vector on 2 modes or density matrix on 1 mode)

Figure 2: Schematic representation of how Eq. (2) can be used to calculate the Fock amplitudes $\mathcal{G}_{\boldsymbol{k}}$ of a Gaussian state. Every Fock index $\boldsymbol{k}$ marks a position in the Fock lattice. Every blue node can be written as a linear combination of the orange nodes.

Here, $\textit{cutoffs} = [C_1, C_2, C_3, ...]$ is the set of upper bounds for the photon numbers in all modes. We assume that they are chosen such that the probability of detecting C_i or more photons in mode i is negligible.

Note that Fock amplitude $\mathcal{G}_{\boldsymbol{0}}$ (where $\boldsymbol{0} = [0, 0, ..., 0]$) is the vacuum component of $\boldsymbol{\mathcal{G}}$. If $\boldsymbol{\mathcal{G}}$ is a density matrix $\boldsymbol{\rho}$, it can be computed as:

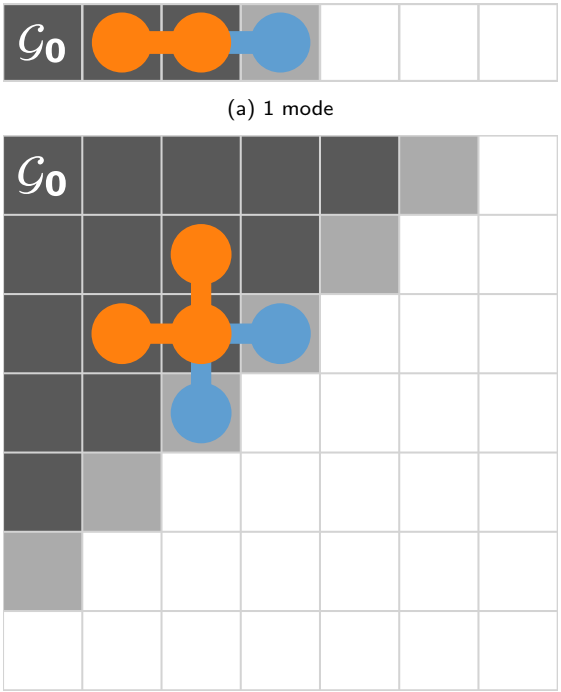
$$\rho_{\boldsymbol{0}} = \frac{\exp\left[-\frac{1}{2}\overline{\boldsymbol{\mu}}^\dagger \boldsymbol{\sigma}_+^{-1}\overline{\boldsymbol{\mu}}\right]}{\sqrt{\det\left(\boldsymbol{\sigma}_+\right)}}, \tag{9}$$

If $\boldsymbol{\mathcal{G}}$ is a state vector $\boldsymbol{\psi}$, ignoring a global phase, it holds that $\psi_{\boldsymbol{0}} = \sqrt{\rho_{\boldsymbol{0}}}$.

Starting from $\mathcal{G}_{\boldsymbol{0}}$, we can calculate all of the amplitudes by applying Eq. (2). We start by placing the pivot of our hypercross at $\boldsymbol{0}$ (for which $w = 0$) and write amplitudes for which $w = 1$. Next, we apply all pivots for which $w = 1$ and write amplitudes for which $w = 2$. By repeatedly increasing w and applying all pivots of that weight, we can calculate the required amplitudes. As the amplitudes we write have a higher weight than the amplitudes we read, we know that the right amplitudes are always calculated before we need to read them.

Fig. 3 shows an intermediate step of this process for circuits that consist of one and two modes. In this

figure, the cutoff values of all modes are chosen to be 7. Dark grey cells depict amplitudes that have already been used as pivots. Light grey cells are amplitudes that have been calculated, but have not yet been used as pivots. At the end of the process all cells in the figure will be calculated.



(a) 1 mode

(b) 2 modes

Figure 3: Intermediate step of state vector simulations for circuits consisting of 1 and 2 modes. Fock amplitudes $\mathcal{G}_{\boldsymbol{k}}$ of the output state vectors are computed recursively. We start at $\mathcal{G}_{\boldsymbol{0}}$ and apply pivots in order of increasing weight until all amplitudes are calculated. At this intermediate step, dark grey cells have been used as pivots. Light grey cells have been written and will be used as pivots in the next step. Animated versions of these figures are included in the Supplementary Materials.

Note that this strategy to calculate Fock amplitudes allows for two types of parallelization. First, given a specific pivot, we can parallelize the calculations of different elements in the 'write' group. Second, since we order pivots according to increasing weight, we can also apply pivots of the same weight simultaneously.

2.3 Alternative cutoff conditions

The boundary conditions of Eq. (8) are useful for simulating circuits for which we know the maximum number of photons that a PNR detectors can measure. The cutoff in the undetected modes can be chosen separately, depending on the required accuracy for calculating the conditional state. However, the recurrence relation also allows one to consider other cutoff conditions.

A first useful example occurs when we want to place an upper bound on the total number of photons that is present in all modes. As the total number operator $\hat{\boldsymbol{n}} = \sum_{i=1}^{M} \hat{n}_i$ commutes with the multi-mode Fock Hamiltonian [12], such an upper bound defines a cutoff on the energy levels of the multi-mode Gaussian state before the detectors. More formally, we can replace Eq. (8) by:

$$0 \leq w(\boldsymbol{k}) < w_{\max} \ , \tag{10}$$

which can be related to an upper bound for the total number of photons $N_{\max}$ in the circuit:

$$w_{\max} = \begin{cases} N_{\max}, & \text{if } \boldsymbol{\mathcal{G}} \text{ is a state vector,} \\ 2N_{\max}, & \text{if } \boldsymbol{\mathcal{G}} \text{ is a density matrix.} \end{cases} \tag{11}$$

Note that for Eq. (10) the number of amplitudes $\mathcal{G}_{\boldsymbol{k}}$ that have the same weight increases binomially with w. For Eq. (8), this number of amplitudes first increases with w, after which it reaches a maximum and decreases. Indeed, once $w \geq \min(\mathit{cutoffs})$, the right inequality of Eq. (8) starts to exclude general Fock indices of weight w. Eventually, when w is raised all the way to $\sum_{i=1}^{D}(C_i - 1)$ the number of allowed indices has decreased back to 1.

Another possible cutoff condition is given by the total sum of the probabilities of PNR outcomes. After each iteration (in which we apply all pivots of weight w), we can evaluate this sum and check whether it is sufficiently close to 1 to stop the process.

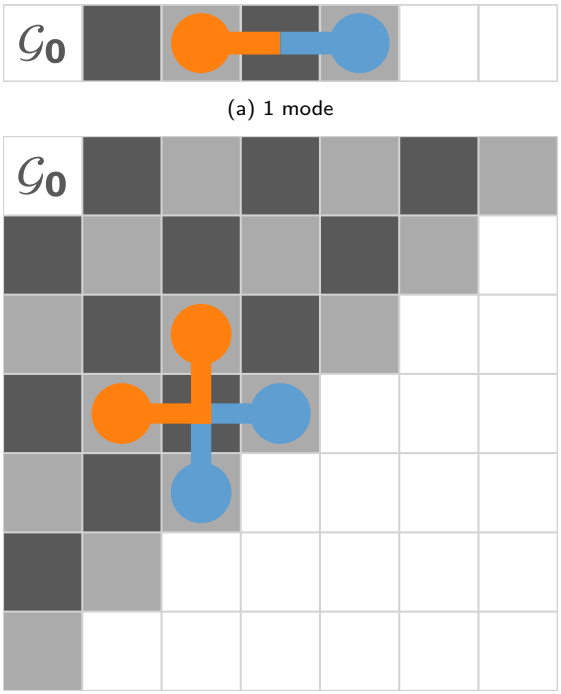
2.4 Circuits without displacement gates

In Reference [31] we showed how to compute the parameters $\boldsymbol{A}$, $\boldsymbol{b}$ and $\mathcal{G}_{\boldsymbol{0}}$ that define a Gaussian operator. More specifically for Gaussian states, we showed how $\boldsymbol{A}$, $\boldsymbol{b}$ and $\mathcal{G}_{\boldsymbol{0}}$ can be calculated from the covariance matrix and means vector. Moreover, it can be shown that for a state with zero displacement vector we have $\boldsymbol{b} = \boldsymbol{0}$. Note that this applies to the states before the detectors in Fig. 1 as these circuits do not contain displacement gates.

In the case that there is no displacement, we can substitute $\boldsymbol{b} = \boldsymbol{0}$ in Eq. (2) such that our recurrence relation turns into:

$$\mathcal{G}_{\boldsymbol{k}+\boldsymbol{1}_i} = \frac{1}{\sqrt{k_i + 1}} \sum_{l=1}^{D} \sqrt{k_l}\, \mathcal{G}_{\boldsymbol{k}-\boldsymbol{1}_l} A_{il} \ . \tag{12}$$

We find that the only Fock amplitudes that differ from zero are the ones which have a Fock index $\boldsymbol{k}$ with even weight. For state vectors, we can alter the strategy described in Fig. 3 by only considering pivots that have odd weight. This leads to the checkered pattern of Fig. 4, where we still apply pivots in order of increasing weight. Note that now we now fill the array twice as fast because we only need to compute half of the amplitudes.



(a) 1 mode

(b) 2 modes

Figure 4: Intermediate step of state vector simulations for circuits that do not contain displacement gates (consisting of 1 and 2 modes). Fock amplitudes $\mathcal{G}_{\boldsymbol{k}}$ are calculated recursively as in Fig. 3, but now they are zero when $\sum_i k_i$ is odd. Consequently, pivots (i.e. the central nodes of the hypercross) do not need to be read. At this intermediate step, dark grey cells have been used as pivots but only for placing the cross (their value remains zero), while light grey cells have been actually written.

2.5 Gradients

In this section, we present how the framework above allows not only to *simulate* but also to *optimize* circuits. Given a loss function L that depends on the probabilities of the PNR outcomes (and the conditionally generated states), we need to calculate the partial derivatives of L with respect to the parameters of the circuit. As explained in Reference [18], the so-called 'down-stream gradient' of L with respect to the conjugate of a complex circuit parameter ξ can be computed using the chain rule as follows:

$$\frac{\partial L}{\partial \xi^*} = \sum_{\boldsymbol{k}} \frac{\partial L}{\partial \mathcal{G}^*_{\boldsymbol{k}}} \frac{\partial \mathcal{G}^*_{\boldsymbol{k}}}{\partial \xi^*} + \frac{\partial L}{\partial \mathcal{G}_{\boldsymbol{k}}} \frac{\partial \mathcal{G}_{\boldsymbol{k}}}{\partial \xi^*} \ . \tag{13}$$

We now consider ξ to be equal to b_m or A_{mn} and note that Eq. (2) does not depend on b^*_m or A^*_{mn}, such that:

$$\frac{\partial L}{\partial b^*_m} = \sum_{\boldsymbol{k}} \frac{\partial L}{\partial \mathcal{G}^*_{\boldsymbol{k}}} \frac{\partial \mathcal{G}^*_{\boldsymbol{k}}}{\partial b^*_m} = \sum_{\boldsymbol{k}} \frac{\partial L}{\partial \mathcal{G}^*_{\boldsymbol{k}}} \left(\frac{\partial \mathcal{G}_{\boldsymbol{k}}}{\partial b_m} \right)^* \ , \tag{14}$$

$$\frac{\partial L}{\partial A^*_{mn}} = \sum_{\boldsymbol{k}} \frac{\partial L}{\partial \mathcal{G}^*_{\boldsymbol{k}}} \frac{\partial \mathcal{G}^*_{\boldsymbol{k}}}{\partial A^*_{mn}} = \sum_{\boldsymbol{k}} \frac{\partial L}{\partial \mathcal{G}^*_{\boldsymbol{k}}} \left(\frac{\partial \mathcal{G}_{\boldsymbol{k}}}{\partial A_{mn}} \right)^* \ . \tag{15}$$

As the upstream gradient tensor $\partial L / \partial \mathcal{G}^*_{\boldsymbol{k}}$ can be provided to us by an automatic differentiation framework such as TensorFlow or PyTorch, we only have to compute the local gradients $\partial \mathcal{G}_{\boldsymbol{k}} / \partial b_m$ and $\partial \mathcal{G}_{\boldsymbol{k}} / \partial A_{mn}$.

From Eq. (2) we now derive:

$$\frac{\partial \mathcal{G}_{\boldsymbol{k}+\mathbf{1}_i}}{\partial b_m} = \frac{1}{\sqrt{k_i+1}} \left(\frac{\partial \mathcal{G}_{\boldsymbol{k}}}{\partial b_m} b_i + \mathcal{G}_{\boldsymbol{k}} \delta_{im} + \sum_{l=1}^{D} \sqrt{k_l} \frac{\partial \mathcal{G}_{\boldsymbol{k}-\mathbf{1}_l}}{\partial b_m} A_{il} \right) \ , \tag{16}$$

$$\frac{\partial \mathcal{G}_{\boldsymbol{k}+\mathbf{1}_i}}{\partial A_{mn}} = \frac{1}{\sqrt{k_i+1}} \left(\frac{\partial \mathcal{G}_{\boldsymbol{k}}}{\partial A_{mn}} b_i + \sum_{l=1}^{D} \sqrt{k_l} \left[\frac{\partial \mathcal{G}_{\boldsymbol{k}-\mathbf{1}_l}}{\partial A_{mn}} A_{il} + \mathcal{G}_{\boldsymbol{k}-\mathbf{1}_l} \delta_{im} \delta_{ln} \right] \right) \ , \tag{17}$$

where δ_{jk} is the Kronecker delta function. Since both Eq. (16) and Eq. (17) are structured in a similar way as Eq. (2), we can implement all three equations simultaneously. We do so by taking a single walk through the Fock lattice, that is, by performing a single iteration over the Fock indices $\boldsymbol{k}$. We still differentiate between the different types of Fock indices 'read' ($\boldsymbol{k} - \mathbf{1}_l$), 'pivot' ($\boldsymbol{k}$) and 'write' ($\boldsymbol{k} + \mathbf{1}_i$), but instead of only manipulating amplitudes $\mathcal{G}_{\boldsymbol{k}}$, we now also process their partial derivatives with respect to b_m and A_{mn}. Note that every $\boldsymbol{k}$ now corresponds with one Fock amplitude $\mathcal{G}_{\boldsymbol{k}}$, D gradients $\partial \mathcal{G}_{\boldsymbol{k}} / \partial b_m$ and D^2 gradients $\partial \mathcal{G}_{\boldsymbol{k}} / \partial A_{mn}$, such that both the memory and time usage of an *optimization* are a factor $1 + D + D^2$ higher than those of a *simulation*.

3 Extension to density matrix simulations

3.1 Algorithm for Gaussian Boson Sampling

Consider a circuit of which all M modes are detected (such as the one in Fig. 1(a)). To capture mixed states (such as can arise in the presence of photon loss) density matrices must be used in place of state vectors. For simplicity, let us assume that the photon number cutoff in each mode is equal to C. To calculate

the probabilities of the C^M possible PNR detection patterns, one could start by following the procedure described in Section 2.2 to calculate all C^{2M} Fock amplitudes of the multi-mode state before the detector. The probability of observing a certain photon number pattern $\boldsymbol{n} = [n_1, n_2, ..., n_M]$ at the detectors is then given by:

$$p(\boldsymbol{n}) = \mathcal{G}_{n_1 n_1 n_2 n_2 ... n_M n_M} \ . \tag{18}$$

However, as we are only interested in the C^M diagonal amplitudes, we can construct a more efficient algorithm that selectively applies the recurrence relation in the Fock lattice. This way we prevent the calculation of irrelevant amplitudes as much as possible. After choosing an adequate set of pivot positions, we can apply them in order of increasing weight.

3.1.1 Single mode

Let us first consider the case where we have a single mode. Here the Fock lattice only has two dimensions (i.e. $\boldsymbol{k} = [m, n]$) and we can use the hypercross of Fig. 2(b). For now also consider the case where the circuit under consideration does not contain displacement gates. As explained in Section 2.4, this implies that the inner 'pivot' node of the hypercross cross does not need to be read. Fig. 5(a) visualizes how Eq. (12) can be applied in order to calculate the required diagonal amplitudes. We have chosen all pivots of the type $[a+1, a]$ that satisfy $[0, 0] \leq [a+1, a] < [C_1, C_1]$. Note that we could have equivalently chosen pivots of the type $[a, a+1]$ instead. We apply the pivots in order of increasing weight, i.e. from the top left to the bottom right. As these pivots only read amplitudes that are previously written by other pivots, the total set of pivots can be said to be 'self-sufficient'.

Fig. 5(b) shows the case where the circuit under consideration does contain displacement gates. Now, we also have to read the value of the pivot node in order to apply the hypercross. These values (at positions $[a+1, a]$) can be provided by introducing extra pivots of the type $[a, a]$. In their turn, the off-diagonal pivots provide the amplitude values of the diagonal pivots. In other words, the total set of the diagonal and off-diagonal pivots is self-sufficient here.

3.1.2 Two modes

We now consider density matrix simulations of GBS circuits with two modes, such that Eq. (2) can be represented by a four dimensional hypercross. However, we still choose to visualize both the hypercross and $\boldsymbol{\mathcal{G}}$ in two dimensions via the Kronecker product. Below, we explain in more detail how such a representation is constructed. The hypercross itself is shown in Fig. 6. Fig. 7 visualizes how this hypercross can be applied to get the diagonal Fock amplitudes in the case where $cutoffs = [4, 4]$.

We write $\boldsymbol{k} = [m, n, p, q]$, where $[m, n]$ and $[p, q]$ are the indices corresponding to the first and second mode respectively. Note now that if $[p, q]$ would be fixed, we are left with a 2D matrix that is only indexed by $[m, n]$, such that it can be visualized in a similar way as Fig. 5. We now combine all such matrices (for all possible values of p and q) in a block matrix. This leads to a 2D 'nested representation'. If $M > 2$, we can recursively apply this process for different index pairs (i.e. constructing block matrices of block matrices), such that we always end up with a 2D image. Note that pivots are no longer applied from top left to bottom right in this representation, as this would not correspond with the order of increasing weight.

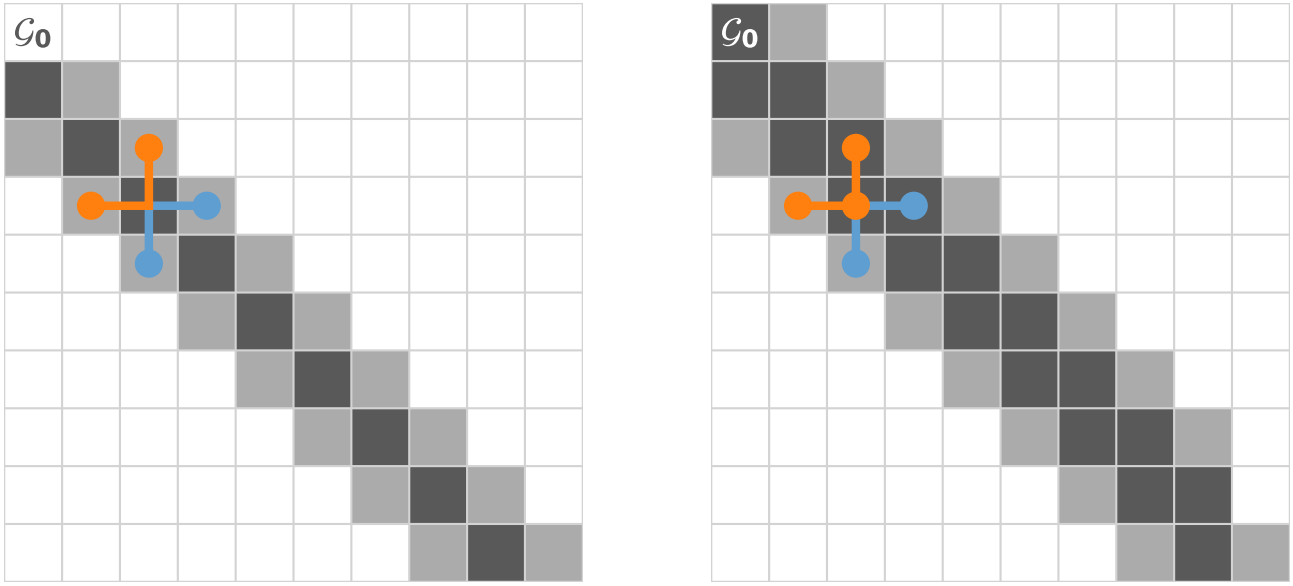
We have to make sure that amplitudes are written before they are read. In other words, the total set of pivots used in Fig. 7 has to be self-sufficient. We can check that this is true by first considering the pivots of the type $[a, a, b, b]$ and $[a+1, a, b, b]$ (i.e. the diagonal cells in Fig. 7 and the cells under those). This set of pivots is *almost* self-sufficient: within each $C_1 \times C_1$ block that lies on the diagonal of Fig. 7 (i.e. within each block containing amplitudes of the type $[m, n, b, b]$), *almost* all of these pivots get their required 'read' and 'pivot' amplitudes from the 'write' amplitudes from another pivot in those blocks. The only amplitudes that are missing to complete the self-sufficiency are the amplitudes of the type $[0, 0, b, b]$ (marked as $\star$). These last amplitudes act like 'seed amplitudes' in the diagonal $C_1 \times C_1$ blocks, similar to how $\mathcal{G}_{\mathbf{0}}$ acts as a seed in Fig. 5. These missing amplitudes can be obtained from the remaining pivots outside of the diagonal $C_1 \times C_1$ blocks: $[0, 0, b+1, b]$ (marked as $\boldsymbol{+}$). These last pivots 'bridge' the gaps between different diagonal $C_1 \times C_1$ blocks by providing the necessary increments of k_i for $i \in \{3, 4, 5, ..., 2M\}$.

3.1.3 General number of modes

The pivot placement strategy of Figs. 5 and 7 can be generalized to a larger number of modes. The strategy for 3 modes is visualized in Appendix A.

Algorithm 1 shows how a GBS circuit with an *arbitrary* number of modes can be simulated in the density matrix formalism. Lines 1 to 5 are used to apply the diagonal pivots $diag = [a, a, b, b, c, c, ...]$ in order of increasing weight. Note again that these pivots are also diagonal in the nested representation, while the order in which we apply them is not necessarily from top left to bottom right (see for example the animated version of Fig. 7 in the Supplementary Materials). In order to apply the diagonal pivots, a variable S is increased stepwise, starting from 0. Each time, we apply all diagonal pivots that satisfy both $a + b + c + ... = S$ and the boundary conditions of Eq. (8).

Lines 6 to 10 are used to apply the off-diagonal pivots $diag + \mathbf{1}_{2K-1}$, where $K \in \{1, 2, ..., M\}$ (i.e. $[a+1, a, b, b, c, c, ...]$, $[a, a, b+1, b, c, c, ...]$,



(a) Circuit without displacement gates

(b) Circuit with displacement gates

Figure 5: Visualisation of how Eq. (2) can be applied to density matrices in order to calculate the detection probabilities $|\mathcal{G}_{n_1 n_1}|^2$ of a single mode circuit. Pivots (dark grey) are applied from top left to bottom right, i.e. in order of increasing weight. Light grey cells are non-pivot amplitudes that are written. White cells do not have to be written, which improves on the naive idea of applying pivots in all cells (as in Fig. 3(b)). In Fig. a, the pivots are not read as there Eq. (2) simplifies to Eq. (12). We chose to upper bound the photon number by 10 in this example. Animated versions of these figures are included in the Supplementary Materials.

$[a, a, b, b, c{+}1, c, ...]$, etc.). For $K{=}1$, the off-diagonal pivots lie in the diagonal $C_1 \times C_1$ blocks. For $K > 1$, the off-diagonal pivots are 'bridge pivots' that provide the 'source amplitudes' $[0, 0, b, b, c, c, ...]$. Note that because of line 8, the number of off-diagonal pivots decreases with K (see both Fig. 7 and Appendix A for reference).

In Appendix B, we show that both the total number of pivots and the total number of written amplitudes that appear in Algorithm 1 scale like $\prod_{i=1}^{M} C_i$, which simplifies to C^M if the cutoffs on all modes are equal.

Note that if the local cutoff conditions of Eq. (8) are replaced by the global cutoff condition of Eq. (10), then the sum of line 1 runs to $N_{\max} = \frac{1}{2} w_{\max}$ instead, while the cutoff conditions in lines 2, 4 and 8 drop out. As shown in Appendix B, the scaling of the algorithm then changes to $(w_{\max})^M$.

3.1.4 Compact storage of the Fock amplitudes

In Appendix B, we show that all amplitudes that are written in Algorithm 1 can be parameterized as $diag + offset$ where $diag$ is a diagonal position in the Fock lattice and $offset$ is an offset vector that only comes in a select number of types. This parametrization helps to store the amplitudes in a unique and compact manner. However, in the case that we detect all modes, we are only interested in the $\prod_{i=1}^{M} C_i$ diagonal amplitudes. The off-diagonal amplitudes do not need long-term storage in memory. It can be shown that all off-diagonal amplitudes are included in the 'read' group of a pivot exactly once. Thus, we can remove off-diagonal 'read' amplitudes from memory once they have been used. We only have to store a buffer of off-diagonal amplitudes that correspond with a select number of weight values. In addition to the animated versions of Figs. 5, 7 and 11, we also include animations in the Supplementary Materials that apply this 'buffer strategy'.

For a circuit consisting of 4 modes (such as the one in Fig. 1(a)), Fig. 8 shows how the number of stored amplitudes evolves as we apply more pivots. We have chosen the photon number cutoff to be 10 in all modes. In contrast to the strategy without buffer (blue curve), the buffer strategy (orange curve) reaches a maximum before the end of the algorithm is reached. This results from the fact that the number of pivots that have an equal weight reaches a maximum at $w = \sum_{i=1}^{M} C_i$ when we apply the local boundary conditions of Eq. (8). For reference, Fig. 8 also shows a horizontal dashed line at $\prod_{i=1}^{M} C_i = C^M = 10^4$. Note that after completing Algorithm 1 using the buffer strategy all off-diagonal amplitudes are removed, such that the orange curve coincides with the dashed curve.

3.2 Algorithm for conditional state generation

Let us now consider circuits where all but one mode are detected, such as the one of Fig. 1(b). Our results can readily be generalized to an arbitrary number of undetected modes. Our goal is now to calculate the

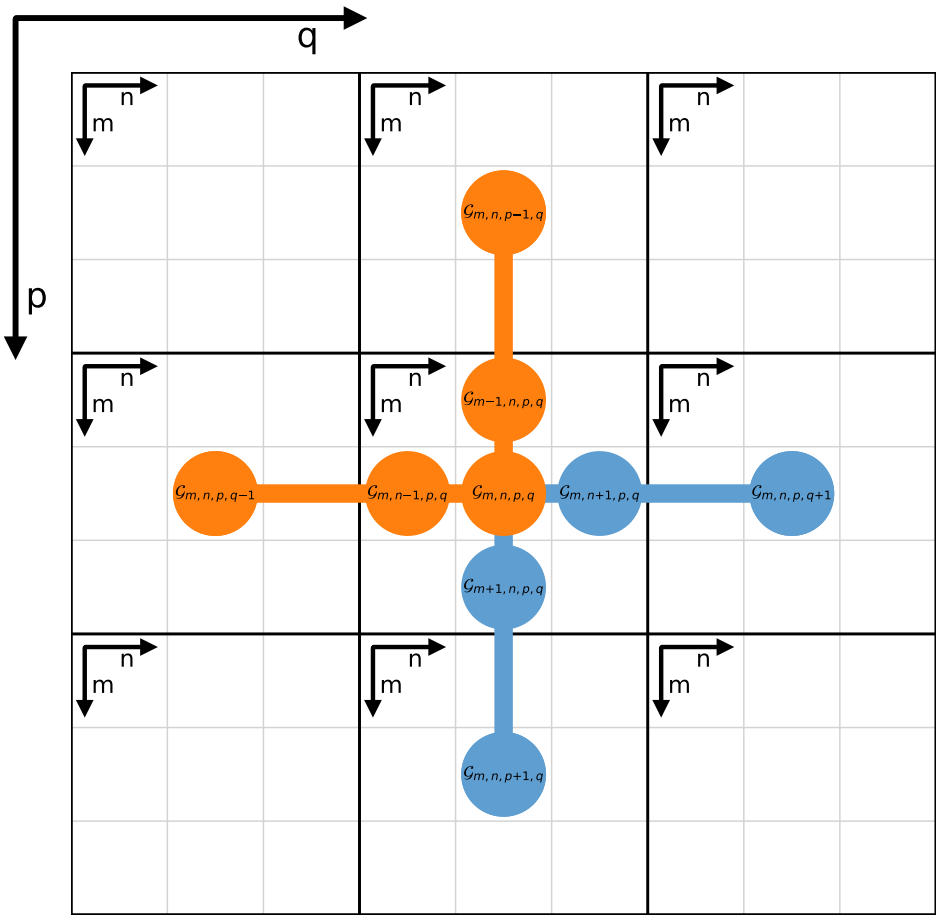


Figure 6: Schematic representation of Eq. (2) where $\mathcal{G}$ is 4-dimensional. The Fock amplitudes $\mathcal{G}_{mnpq}$ are represented via the Kronecker product: all $C_1 \times C_1$ corresponding to different values of p and q are combined in a block matrix.

Algorithm 1 Density matrix simulation of a GBS circuit

for $S \leftarrow 0$ to $(\sum_{i=1}^{M} C_i) - 1$ **do** // Stepwise increase of pivot weight
  calculate the set *diag_set* of all length-$2M$ indices $[a, a, b, b, c, c, ...]$ that satisfy $a + b + c + ... = S$ and $\mathbf{0} \leq [a, b, c, ...] < \mathit{cutoffs}$
  for *diag* in *diag_set* **do**
    if $\mathit{diag}_1 < C_1 - 1$ **then**
      apply *diag* as pivot // Diagonal pivot ($w = 2S$)
  for *diag* in *diag_set* **do**
    for $K \leftarrow 1$ to M **do**
      if the first $2(K-1)$ elements of *diag* are 0 **then**
        if $\mathit{diag}_{2K} < C_K - 1$ **then**
          apply $\mathit{diag} + \mathbf{1}_{2K-1}$ as pivot // Off-diagonal pivot ($w = 2S + 1$)

distribution of states that are generated conditionally on the PNR detection results. As a first example, we consider a circuit with two modes and one detector, such that we can use the nested representation of Fig. 6. In this representation, the targeted distribution is defined by the Fock amplitudes $\mathcal{G}_{mnpq}$ in the diagonal $C_1 \times C_1$ blocks. Each detection outcome corresponds with one such $C_1 \times C_1$ block, which is the unnormalized density matrix of the conditional state.

The targeted blocks can be calculated using the two step process presented in Fig. 9. First, we calculate all Fock amplitudes in the upper left $C_1 \times C_1$ block, which is the density matrix corresponding with detecting zero photons. For this first step, we can use the hypercross of Fig. 6 where we choose only to *increment* indices m and n (not p and q). Note that we also do not have to *decrement* p and q, as these amplitudes would correspond with negative photon numbers. For the second step of our simulation process, we do have to *decrement* all indices, but this time we choose only to *increment* indices p and q (not m and n). Moreover, we choose to apply pivots in blocks of

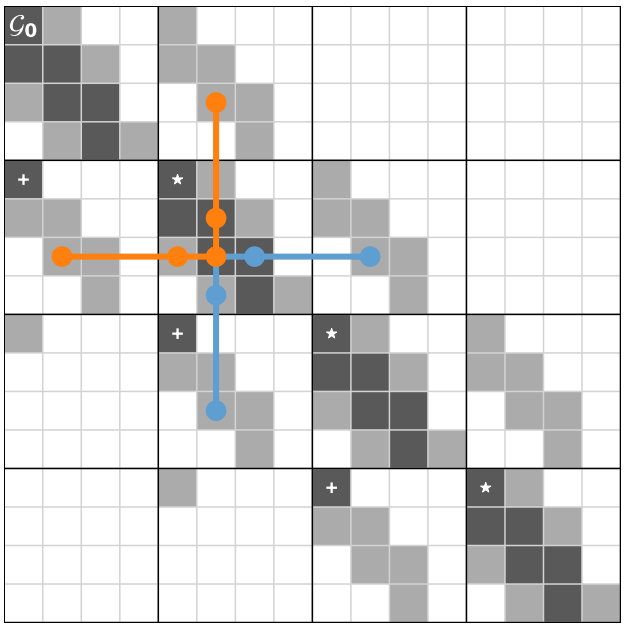


Figure 7: Visualisation of how Eq. (2) (i.e. the hypercross from Fig. 6) can be applied to density matrices in order to calculate the detection probabilities $\mathcal{G}_{n_1 n_1 n_2 n_2}$ of a two mode circuit. The photon number in both modes is upper bound by 4. Dark grey cells represent pivots. Light grey cells represent non-pivot amplitudes that are written. The pivots marked as ✦ write to the pivots marked as ⋆. Similar to $\mathcal{G}_{\mathbf{0}}$, these last pivots (⋆) act as 'seed' amplitudes in their $C_1 \times C_1$ blocks. An animated version of this figure is included in the Supplementary Materials.

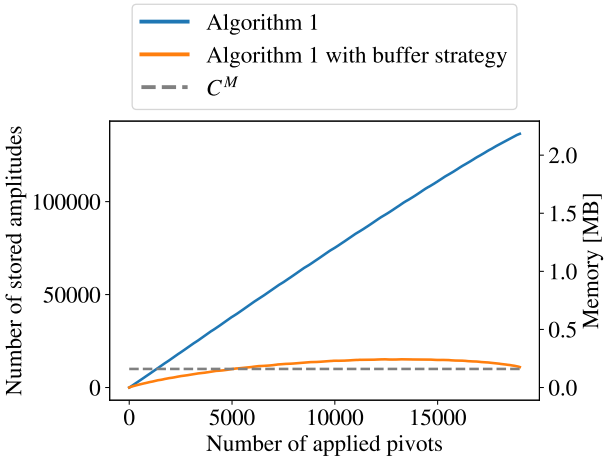


Figure 8: The number of stored amplitudes when using a buffer of off-diagonal amplitudes (orange curve) and without using such a buffer (blue curve) as a function of the number of applied pivots. After all pivots are applied, the buffer is left empty such that the orange curve reaches a value of C^M (dashed curve). This figure is made using 4 modes, all with photon number cutoff of 10. Both the real and complex part of the amplitudes are stored as 64-bit-precision floating-point numbers.

size $C_1 \times C_1$. By doing so, we can apply a coarse-grained version of Algorithm 1 *as if* the circuit under consideration has $M-1$ modes. In this example, $M = 2$ such that we apply a coarse grained version of Fig. 5(b). Within each $C_1 \times C_1$ block of pivots, the individual pivots still need to be applied according to increasing weight, similar to Fig. 3(b).

This simulation process for state generator circuits can be generalized to an arbitrary number of modes M. Algorithm 2 considers all cases where we have 1 undetected mode and $M-1$ detected modes. The extension to an arbitrary number of undetected modes is straightforward. A similar two step process is followed as in Fig. 9. Note that step 2 of Algorithm 2 is indeed a coarse-grained version of Algorithm 1 as we apply $C_1 \times C_1$ blocks of pivots. That is, we apply pivots $[m, n, a, a, b, b, c, c, ...]$ for $m, n \in \{0, 1, ..., C_1 - 1\}$ where $a, b, c, ...$ follow from Algorithm 1 after substituting M by $M-1$ and *cutoffs* by $[C_2, C_3, C_4, ...]$. As Algorithm 1 scales as $\prod_{i=1}^{M} C_i$, it is clear from the above that Algorithm 2 scales as ${C_1}^2 \prod_{i=2}^{M} C_i$. In the case where we choose all modes to have the same cutoff C, these scaling factors are C^M and C^{M+1} respectively.

4 Complexity

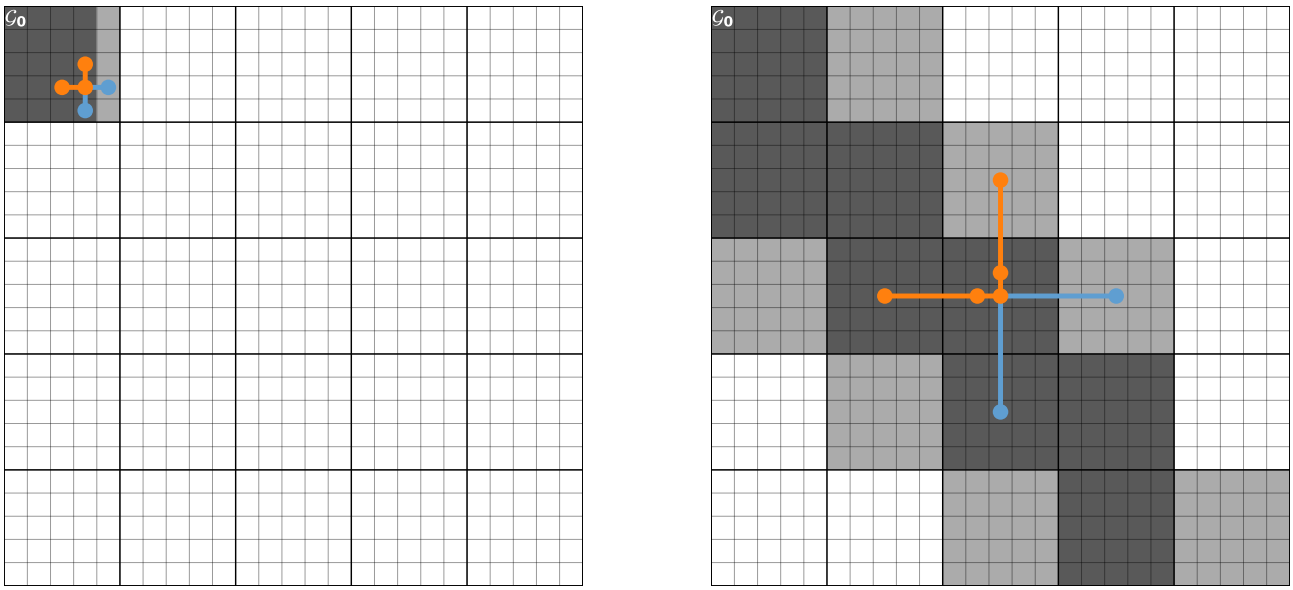
In the case where we use state vectors, Section 2.2 explains how pivots can be applied to calculate all Fock amplitudes that satisfy the cutoff conditions of Eq. (8). The total number of pivots then scales as $O(\prod_{i=1}^{M} C_i)$. In the case where we simulate a GBS circuit using density matrices, we apply Algorithm 1. In Appendix B, we show that the total number of pivots that are used in this algorithm also scales as $O(\prod_{i=1}^{M} C_i)$.

As is clear from Eq. (2), the complexity of applying a single pivot is given by D^2. (Note that Eq. (2) can be rewritten as the sum of a vector and a matrix-vector multiplication by rescaling $\mathcal{G}_{\boldsymbol{k}-\mathbf{1}_l}$ and $\mathcal{G}_{\boldsymbol{k}+\mathbf{1}_i}$ with $\sqrt{k_l}$ and $\sqrt{k_i+1}$ respectively.) From Eq. (1) it follows that both using state vectors and density matrices, our algorithms for GBS simulation scale like $O(M^2 \prod_{i=1}^{M} C_i)$. As is clear from Section 3.2, for the generation of single mode conditional states, this complexity changes to $O(M^2 C_1^2 \prod_{i=2}^{M} C_i)$. Algorithm 2 can readily be extended to account for a general number of undetected modes. By doing so, the complexity changes to:

$$O(M^2 \prod_{i \in I_U} C_i^2 \prod_{i \in I_D} C_i), \tag{19}$$

where I_U and I_D are the sets of indices i that respectively correspond to undetected and detected modes.

In the remainder of this work, we first demonstrate how this scaling behaviour can be observed for circuits with 4 modes. Afterwards, the results for GBS circuits are compared to the state-of-the-art classical simulation method.



(a) Calculate upper left $C_1 \times C_1$ block

(b) Apply coarse-grained version of Algorithm 1

Figure 9: Visualisation of Algorithm 2 for a circuit of two modes, one of which is detected. Dark grey cells represent pivots. Light grey cells represent non-pivot amplitudes that are written. White cells represent amplitudes that are not written. For the pivots $\mathcal{G}_{3300}$ (in Fig. a) and $\mathcal{G}_{2222}$ (in Fig. b) the hypercross is shown, where we only keep two of its blue nodes. For both modes, we chose a photon number cutoff of 5. Animated versions of these figures are included in the Supplementary Materials.

Algorithm 2 Density matrix simulation of a conditional state generator circuit

Step 1 Calculate $C_1 \times C_1$ block that corresponds with zero photon detections

apply all pivots $[m, n, 0, 0, ...]$ for $m \in \{0, 1, ..., C_1 - 1\}$, $n \in \{0, 1, ..., C_1 - 2\}$
▷ **only write** amplitude types $[m+1, n, 0, 0, ...]$ and $[m, n+1, 0, 0, ...]$

Step 2 Coarse-grained version of Algorithm 1

for $S \leftarrow 0$ to $(\sum_{i=2}^{M} C_i)$-1 **do**
  calculate the set *diag_set* of all indices $[a, a, b, b, c, c, ...]$ of length $2(M-1)$ that satisfy $a + b + c + ... = S$ and $\mathbf{0} \leq [a, b, c, ...] < [C_2, C_3, C_4, ...]$
  for *diag* in *diag_set* **do**
    if $diag_2 < C_2 - 1$ **then**
      apply all pivots $[m, n, diag]$ for $m, n \in \{0, 1, ..., C_1 - 1\}$ // Diagonal $C_1 \times C_1$ block
      ▷ **do not write** amplitude types $[m+1, n, 0, 0, ...]$ and $[m, n+1, 0, 0, ...]$
  for *diag* in *diag_set* **do**
    for $K \leftarrow 1$ to $M - 1$ **do**
      if the first $2(K-1)$ elements of *diag* are 0 **then**
        if $diag_{2K} < C_{1+K} - 1$ **then**
          apply all pivots $[m, n, diag] + \mathbf{1}_{2K+1}$ for $m, n \in \{0, 1, ..., C_1 - 1\}$ // Off-diagonal $C_1 \times C_1$ block
          ▷ **do not write** amplitude types $[m+1, n, 0, 0, ...]$ and $[m, n+1, 0, 0, ...]$

4.1 Memory usage and simulation time

Fig. 10 visualizes the memory usage and simulation time for a circuit with 4 modes (such as the circuits in Fig. 1). We have chosen the photon number cutoff C to be equal for all modes. As both the memory usage and simulation time scale with the number of applications of Eq. (2) (i.e. the number of pivots), the trends in Figs. 10(a) and 10(b) are similar.

When using state vectors, we calculate C^M amplitudes to simulate a circuit, regardless of the number of PNR detectors (green line in Fig. 10(a)). When using density matrices, this number would increase to C^{2M} (orange line in Fig. 10(a)) if we naively applied the strategy of Section 2.2. When all modes in the circuit are measured, Algorithm 1 reduces the memory requirements from the orange curve to the solid blue curve. This last curve corresponds with the number of written amplitudes given in Appendix B.3. It can be lowered further to the dashed blue curve when the

buffer strategy of Section 3.1.4 is applied. Note that the memory usage at a cutoff value of 10 corresponds with the maximum of the orange curve in Fig. 8 . From the slopes of these curves we verify that the complexity of Algorithm 1 is equal to the complexity of a state vector simulation, i.e. C^M, as was discussed in Section 3.1.3. When all but one mode of the circuit are detected, we can use Algorithm 2 to improve on the naive strategy without selective pivot placement. As discussed in Section 3.2, the complexity of this last algorithm is C^{M+1}.

When calculating both the required amplitudes (Eq. (2)) and gradients (Eqs. (16) and (17)) to optimize the circuit, we know from Section 2.5 that we can implement all three equations by taking a single walk through the Fock lattice. As a result, the memory usage of an *optimization* is a factor $1 + D + D^2$ higher than the memory usage of a *simulation* (where $D = M$ for state vectors and $D = 2M$ for density matrices). When we would calculate both amplitudes and gradients for Fig. 10 (where $M = 4$), this means that the orange, red and blue curves would shift up on the log scale corresponding with a factor of $1 + 2M + 4M^2 = 73$, while the factor for the green curve would be $1 + M + M^2 = 21$.

Note that when performing an optimization using our technique, the cost function L (which could be chosen to be the fidelity to a target state for example) determines only the complexity of the first step of the chain rule, which consists in calculating $\partial L/\partial \mathcal{G}^*_{\boldsymbol{k}}$ (cf. Eqs. (14) and (15)). This quantity can be provided to us by an automatic differentiation framework such as TensorFlow or PyTorch. The *subsequent steps* of the chain rule, which are given by the gradients that we compute in Eq. (16) and (17) are independent of L and essentially dictate the required computation time and memory resources.

4.2 Comparison with the state-of-the-art GBS algorithm

In this section we focus our attention on the case where all modes are detected. This provides us with a useful reference point for our algorithms, since classical GBS algorithms are well studied [9, 14, 17, 20]. We should note that there exist approximate GBS sampling algorithms such as [19] that vastly outperform approaches where the probabilities are computed exactly in terms of simulation time and memory requirements. These are appropriate for instance in applications where the samples are needed rather than the exact probabilities. Such algorithms are not considered here.

When using state vectors, the state-of-the-art classical GBS algorithm [9] obtains the probability of a single detection pattern $\boldsymbol{n} = [n_1, n_2, ..., n_M]$ with a complexity that is upper bounded by $N^3 2^{N/2}$ (where $N = \sum_i n_i$) and lower bounded by $N^3 \prod_{i=1}^{M} \sqrt{n_i + 1}$. This algorithm is primarily used to *generate samples* from a GBS circuit, i.e. to draw a pattern $\boldsymbol{n}$ from its measurement probability distribution. A popular method for this is 'chain rule sampling', where the photon number in each mode is sampled sequentially, conditioned on the photon numbers in the previous modes. This method only requires the calculation of the conditional probability distributions of the modes instead of the total joint probability distribution.

Instead of *sampling* from a GBS circuit, here we obtain its *joint probability distribution* by calculating the probabilities of all detection outcomes up to a certain photon number cutoff. This is useful to study quantum algorithms based on GBS [4, 8]. Naively, one could apply the algorithm of Reference [9] to all detection patterns up to a certain photon number cutoff. Assuming all probabilities can be obtained at the lower bound of the complexity, we get:

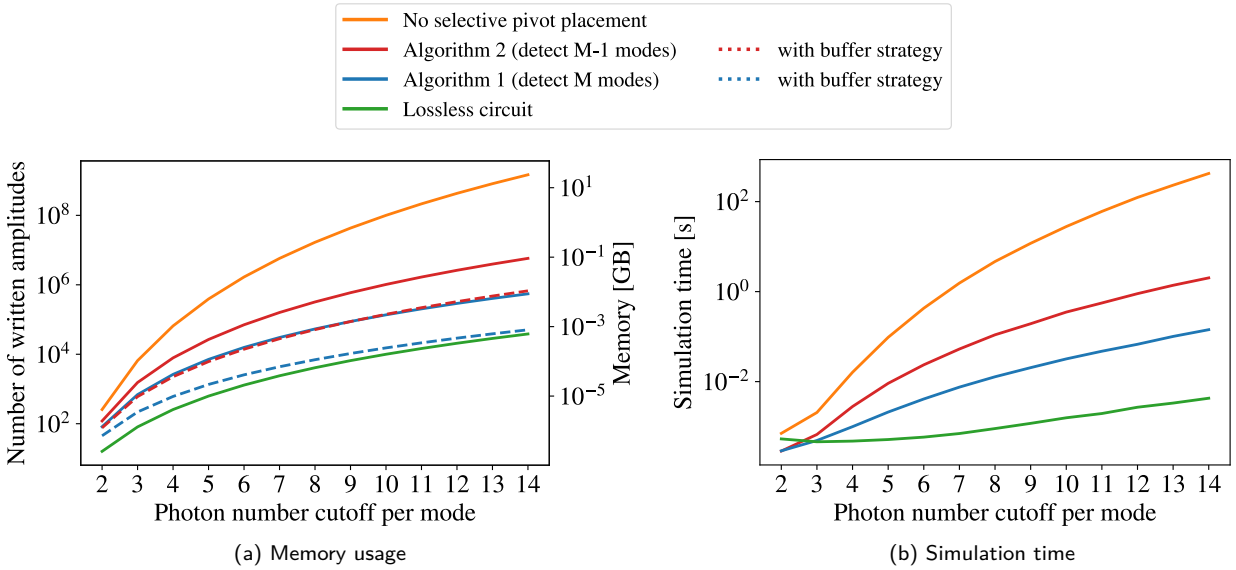
$$\sum_{\boldsymbol{n}=\boldsymbol{0}}^{cutoffs} N^3 \prod_{i=1}^{M} \sqrt{n_i + 1}. \tag{20}$$

In Appendix C it is shown that this is higher than the complexity of our algorithm, which is $M^2 \prod_{i=1}^{M} C_i$. Note that to obtain $p(\boldsymbol{n})$ using Algorithm 1, we need to substitute C_i by $n_i + 1$.

Reference [9] also provides a way to obtain all probabilities $p([n', n_2, ..., n_M])$ (where $n' \in [0, 1, ..., C-1]$ and all other n_i are fixed) at once, with the same complexity of obtaining only $p([C-1, n_2, ..., n_M])$. Nonetheless, Appendix C shows that fixing n_1 to $C_1 - 1$ in Eq. (20) still results in a complexity that is higher than $M^2 \prod_{i=1}^{M} C_i$. Currently, the algorithm in Reference [9] is not extended to include more than one 'batched' mode, and hence our algorithm is faster at obtaining the total joint probability distribution of a GBS circuit. However, if an extension to multiple batched modes were to be made, it might improve on our algorithm when using state vectors. This forms an interesting open research question.

In the case of density matrix simulations, the complexity of our algorithm ($M^2 \prod_{i=1}^{M} C_i$) remains unaltered, while Reference [9] presents a complexity of $N^3 \prod_{i=1}^{M} (n_i + 1)$. Note that, although this last expression is quadratically higher than the *lower bound* of their algorithm for state vectors, it denotes the *actual* complexity to calculate a single probability $p(\boldsymbol{n})$. It follows that for $N^3 > M^2$ (e.g. when $C_i > 1$, $\forall\, i \in [1, 2, ..., M]$), our algorithm scales better, while it also produces the probabilities of all detection patterns with lower photon numbers. Consequently, two regimes can be defined for density matrix simulations. If $N^3 > M^2$, a possible extension of Reference [9] to multiple batched modes would not improve on our algorithm. For $N^3 < M^2$ this question remains open for further study.

Regarding gradients, there exist alternative techniques such as computing gradients analytically or



(a) Memory usage

(b) Simulation time

Figure 10: The memory usage and simulation time of simulating a 4 mode circuit, where the photon number cutoff C is chosen equal in all modes. The complexities of these algorithms are given by the slopes at large C. Memory and time required for Algorithm 1 scales as C^M, which matches the scaling of a state vector simulation (green). This is quadratic improvement over the C^{2M} complexity of the naive strategy without selective pivot placement (orange). Similarly, Algorithm 2 scales as C^{M+1}. The dashed curves show how the memory requirements of Algorithms 1 and 2 are lowered by the buffer strategy of Section 3.1.4. Note that the atypical behavior of simulation time for small cutoff is a result of time incurred during initialization.

using the parameter shift rule. Computing analytical gradients, even with a given formula is usually slower than our technique because typically the analytic formula involves functions that are more complex than the steps of our recurrence relation. For example, the displacement gate entries in Fock representation are given by a combination of Laguerre polynomials, factorials and exponential functions [10] and although they can be derived analytically, the resulting derivative function is more complex than the few multiplications and additions required the recurrent formulation of the gradient of the displacement gate. Parameter shift rules [23] require two forward passes per parameter, but they are not universal in the sense that there exists a parameter shift rule only for specific Gaussian unitaries (displacements, beamsplitters, squeezers etc). The complexity of the parameter shift is analogous to the complexity of our technique, however even though in this paper we have focused on Gaussian states, using recurrence relations for gradients works for any Gaussian object, including Gaussian unitaries and Gaussian channels [31].

5 Conclusions

We have presented an exact procedure to obtain the detection probabilities and conditional states of noisy linear optical quantum circuits with PNR detectors. For a circuit with M modes, we propose an algorithm for which the memory requirements and speed have a complexity of $\mathcal{O}(M^2 \prod_{i=1}^{M} C_i)$, where C_i is the photon number cutoff of mode i. This constitutes a quadratic improvement over previous approaches.

The reduction in complexity applies to measured modes, even when we are after computing marginal states. Moreover, our methods can easily be adapted to obtain the gradients of the detection probabilities and conditional states with respect to a circuit parametrization.

These methods are included in the open-source library `MrMustard` [29]. They are written in pure Python using `Numpy` and are sped up using the just-in-time compiling capabilities of `Numba`. This paves the way to making simulations and optimizations of realistic circuits with PNR detectors. We expect our methods to accelerate the research on both GBS based algorithms and conditional state generation, with a particular emphasis on GKP state generation using ansatze such as the one in Fig. 1(b).

Acknowledgements

Special thanks to Rachel S. Chadwick, Sebastián Duque Mesa, Peter Bienstman and Guy Van der Sande for the valuable discussions. The work of Robbe De Prins was performed in the context of the Flemish FWO project G006020N and the Belgian EOS project G0H1422N. It was also co-funded by the European Union in the Prometheus Horizon Europe project. His international mobility was made possible by the Scientific Research Committee (CWO) of Ghent University. The work of Anuj Apte is supported by Yoichiro Nambu Graduate Fellowship courtesy of Department of Physics, University of Chicago.

References

[1] Juan Miguel Arrazola and Thomas R. Bromley. Using Gaussian boson sampling to find dense subgraphs. *Physical Review Letters*, 121(3), July 2018. DOI: 10.1103/physrevlett.121.030503.

[2] Juan Miguel Arrazola, Thomas R. Bromley, and Patrick Rebentrost. Quantum approximate optimization with Gaussian boson sampling. *Physical Review A*, 98(1), July 2018. DOI: 10.1103/physreva.98.012322.

[3] Leonardo Banchi, Mark Fingerhuth, Tomas Babej, Christopher Ing, and Juan Miguel Arrazola. Molecular docking with Gaussian boson sampling. *Science Advances*, 6(23), June 2020. DOI: 10.1126/sciadv.aax1950.

[4] Leonardo Banchi, Nicolás Quesada, and Juan Miguel Arrazola. Training Gaussian boson sampling distributions. *Physical Review A*, 102(1):012417, 2020. DOI: 10.1103/PhysRevA.102.012417.

[5] J. Eli Bourassa, Rafael N. Alexander, Michael Vasmer, Ashlesha Patil, Ilan Tzitrin, Takaya Matsuura, Daiqin Su, Ben Q. Baragiola, Saikat Guha, Guillaume Dauphinais, et al. Blueprint for a scalable photonic fault-tolerant quantum computer. *Quantum*, 5:392, 2021. DOI: 10.22331/q-2021-02-04-392.

[6] Kamil Brádler, Pierre-Luc Dallaire-Demers, Patrick Rebentrost, Daiqin Su, and Christian Weedbrook. Gaussian boson sampling for perfect matchings of arbitrary graphs. *Physical Review A*, 98(3), September 2018. DOI: 10.1103/physreva.98.032310.

[7] Kamil Brádler, Shmuel Friedland, Josh Izaac, Nathan Killoran, and Daiqin Su. Graph isomorphism and Gaussian boson sampling. *Special Matrices*, 9(1):166–196, January 2021. DOI: 10.1515/spma-2020-0132.

[8] Thomas R. Bromley, Juan Miguel Arrazola, Soran Jahangiri, Josh Izaac, Nicolás Quesada, Alain D. Gran, Maria Schuld, Jeremy Swinarton, Zeid Zabaneh, and Nathan Killoran. Applications of near-term photonic quantum computers: software and algorithms. *Quantum Science and Technology*, 5(3):034010, 2020. DOI: 10.1088/2058-9565/ab8504.

[9] Jacob F. F. Bulmer, Bryn A. Bell, Rachel S. Chadwick, Alex E. Jones, Diana Moise, Alessandro Rigazzi, Jan Thorbecke, Utz-Uwe Haus, Thomas Van Vaerenbergh, Raj B. Patel, et al. The boundary for quantum advantage in Gaussian boson sampling. *Science advances*, 8(4): eabl9236, 2022. DOI: 10.1126/sciadv.abl9236.

[10] Kevin E. Cahill and Roy J. Glauber. Density operators and quasiprobability distributions. *Physical Review*, 177(5):1882, 1969. DOI: 10.1103/PhysRev.177.1882.

[11] Kosuke Fukui, Shuntaro Takeda, Mamoru Endo, Warit Asavanant, Jun-ichi Yoshikawa, Peter van Loock, and Akira Furusawa. Efficient backcasting search for optical quantum state synthesis. *Phys. Rev. Lett.*, 128:240503, June 2022. DOI: 10.1103/PhysRevLett.128.240503.

[12] Christopher C. Gerry and Peter L. Knight. *Introductory quantum optics*. Cambridge university press, 2005.

[13] Daniel Gottesman, Alexei Kitaev, and John Preskill. Encoding a qubit in an oscillator. *Phys. Rev. A*, 64:012310, June 2001. DOI: 10.1103/PhysRevA.64.012310.

[14] Craig S. Hamilton, Regina Kruse, Linda Sansoni, Sonja Barkhofen, Christine Silberhorn, and Igor Jex. Gaussian boson sampling. *Phys. Rev. Lett.*, 119:170501, October 2017. DOI: 10.1103/PhysRevLett.119.170501.

[15] Joonsuk Huh and Man-Hong Yung. Vibronic boson sampling: Generalized Gaussian boson sampling for molecular vibronic spectra at finite temperature. *Scientific Reports*, 7(1), August 2017. DOI: 10.1038/s41598-017-07770-z.

[16] Soran Jahangiri, Juan Miguel Arrazola, Nicolás Quesada, and Nathan Killoran. Point processes with Gaussian boson sampling. *Physical Review E*, 101(2), February 2020. DOI: 10.1103/physreve.101.022134.

[17] Regina Kruse, Craig S. Hamilton, Linda Sansoni, Sonja Barkhofen, Christine Silberhorn, and Igor Jex. Detailed study of Gaussian boson sampling. *Phys. Rev. A*, 100:032326, September 2019. DOI: 10.1103/PhysRevA.100.032326.

[18] Filippo M. Miatto and Nicolás Quesada. Fast optimization of parametrized quantum optical circuits. *Quantum*, 4:366, 2020. DOI: 10.22331/q-2020-11-30-366.

[19] Changhun Oh, Minzhao Liu, Yuri Alexeev, Bill Fefferman, and Liang Jiang. Tensor network algorithm for simulating experimental Gaussian boson sampling. *arXiv preprint arXiv:2306.03709*, 2023. DOI: 10.48550/arXiv.2306.03709.

[20] Nicolás Quesada. Franck-Condon factors by counting perfect matchings of graphs with loops. *The Journal of chemical physics*, 150(16):164113, 2019. DOI: 10.1063/1.5086387.

[21] Nicolás Quesada, Luke G. Helt, Josh Izaac, Juan Miguel Arrazola, Reihaneh Shahrokhshahi, Casey R. Myers, and Krishna K. Sabapathy. Simulating realistic non-Gaussian state preparation. *Phys. Rev. A*, 100:022341, August 2019. DOI: 10.1103/PhysRevA.100.022341.

[22] Krishna K. Sabapathy, Haoyu Qi, Josh Izaac, and Christian Weedbrook. Production of photonic universal quantum gates enhanced by machine learning. *Phys. Rev. A*, 100:012326, July 2019. DOI: 10.1103/PhysRevA.100.012326.

[23] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *Phys. Rev. A*, 99(3):032331, 2019. DOI: 10.1103/PhysRevA.99.032331.

[24] Maria Schuld, Kamil Brádler, Robert Israel, Daiqin Su, and Brajesh Gupt. Measuring the similarity of graphs with a Gaussian boson sampler. *Physical Review A*, 101(3), March 2020. DOI: 10.1103/physreva.101.032314.

[25] Daiqin Su, Casey R. Myers, and Krishna K. Sabapathy. Conversion of Gaussian states to non-Gaussian states using photon-number-resolving detectors. *Phys. Rev. A*, 100:052301, November 2019. DOI: 10.1103/PhysRevA.100.052301.

[26] Daiqin Su, Casey R. Myers, and Krishna K. Sabapathy. Generation of photonic non-Gaussian states by measuring multimode Gaussian states. *arXiv preprint arXiv:1902.02331*, 2019. DOI: 10.48550/arXiv.1902.02331.

[27] Kan Takase, Jun-ichi Yoshikawa, Warit Asavanant, Mamoru Endo, and Akira Furusawa. Generation of optical Schrödinger cat states by generalized photon subtraction. *Phys. Rev. A*, 103:013710, January 2021. DOI: 10.1103/PhysRevA.103.013710.

[28] Kan Takase, Kosuke Fukui, Akito Kawasaki, Warit Asavanant, Mamoru Endo, Jun-ichi Yoshikawa, Peter van Loock, and Akira Furusawa. Gaussian breeding for encoding a qubit in propagating light. *arXiv preprint arXiv:2212.05436*, 2022. DOI: 10.48550/arXiv.2212.05436.

[29] Xanadu Quantum Technologies. MrMustard. `https://github.com/XanaduAI/MrMustard`, 2022.

[30] Ilan Tzitrin, J. Eli Bourassa, Nicolas C. Menicucci, and Krishna K. Sabapathy. Progress towards practical qubit computation using approximate Gottesman-Kitaev-Preskill codes. *Phys. Rev. A*, 101:032315, March 2020. DOI: 10.1103/PhysRevA.101.032315.

[31] Yuan Yao, Filippo M. Miatto, and Nicolás Quesada. The recursive representation of Gaussian quantum mechanics. *arXiv preprint arXiv:2209.06069*, 2022. DOI: 10.48550/arXiv.2209.06069.

A Density matrix simulation of a 3 mode GBS circuit

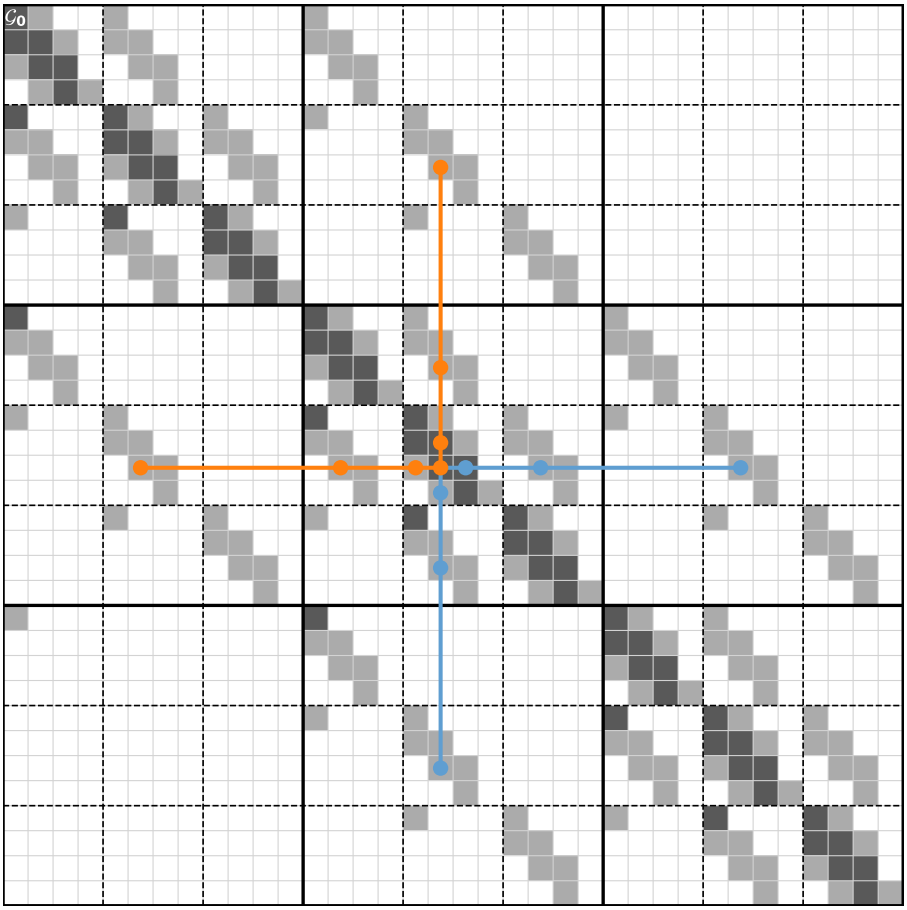


Figure 11: Visual representation of Algorithm 1 for 3 modes, with *cutoffs* $= (4, 3, 3)$. A generalized version of Fig. 6 is used to visualize the Fock amplitudes $\mathcal{G}_{mnpqrs}$. Dark grey cells represent pivots. Light grey cells represent non-pivot amplitudes that are written. White cells represent amplitudes that are not computed. An animated version of this figure is included in the Supplementary Materials.

B Further analysis of Algorithm 1

B.1 Total number of pivots

In this section we derive the total number of pivots that are used in Algorithm 1. First consider the diagonal pivots $diag = [a, a, b, b, c, c, ...]$. We know there are $\prod_{i=1}^{M} C_i$ such amplitudes that satisfy $\mathbf{0} \leq diag < cutoffs$, but we do not have to use all of them as pivots in order to obtain all diagonal amplitudes. As is clear from Fig. 11, we do not use the bottom right diagonal amplitude in the diagonal $C_1 \times C_1$ blocks. In other words, we only need $(C_1 - 1)\prod_{i=2}^{M} C_i$ diagonal pivots. In a similar way, it can be seen that we use $(C_1 - 1)\prod_{i=2}^{M} C_i$ pivots of the type $diag + \mathbf{1}_1$, $(C_2 - 1)\prod_{i=3}^{M} C_i$ pivots of the type $diag + \mathbf{1}_3$, $(C_3 - 1)\prod_{i=4}^{M} C_i$ pivots of the type $diag + \mathbf{1}_5$, etc. In general, the number of pivots is equal to:

$$(C_1 - 1)\prod_{i=2}^{M} C_i + \sum_{K=1}^{M}\left[(C_K - 1)\prod_{i=K+1}^{M} C_i\right] . \tag{21}$$

If the photon number cutoff is equal for all modes, this simplifies to:

$$(C - 1)\left[C^{M-1} + \sum_{K=1}^{M} C^{M-K}\right] = 2C^M - C^{M-1} - 1, \tag{22}$$

which scales as C^M.

If we use the global cutoff condition of Eq. (10) instead, the number of pivots is equal to:

$$\begin{aligned}
&\sum_{N=0}^{N_{\max}-1}\left[\frac{(N+M-1)!}{(M-1)!N!}+\sum_{K=1}^{M}\frac{(N+M-K)!}{(M-K)!N!}\right] \\
&=\binom{N_{\max}+M-1}{N_{\max}}+\binom{N_{\max}+M}{N_{\max}}-1 \\
&\leq 2(N_{\max}+M)^{\min(N_{\max},M)}-1\ .
\end{aligned}$$

Assuming $M \ll N_{\max}$, this upper bound scales as $(N_{\max})^M$.

B.2 Types of (off-)diagonal amplitudes

Let us consider the amplitudes that are read and written when applying a pivot in Algorithm 1. We subdivide all pivots in two types: diagonal pivots ($diag = [a, a, b, b, c, c, ...]$) and off-diagonal pivots ($diag + \mathbf{1}_K$ where $K \in \{1, 3, 5, ..., 2M-1\}$).

A diagonal pivot $diag$ reads amplitudes $diag - \mathbf{1}_i$ and writes amplitudes $diag + \mathbf{1}_i$, where $i \in \{1, 2, 3, ..., 2M\}$. Each amplitude of the type $diag - \mathbf{1}_i$ can always be rewritten as $diag' + \mathbf{1}_{i'}$, where $i' = i + (-1)^{i+1}$ and $diag'$ is obtained by lowering $diag_i$ and $diag_{i'}$ by 1. We conclude that a diagonal pivot reads and writes amplitudes of the type $diag + offset_1$, where $offset_1 = \mathbf{1}_i$ ($i \in \{1, 2, 3, ..., 2M\}$).

An off-diagonal pivot $diag + \mathbf{1}_K$ ($K \in \{1, 3, 5, ..., 2M-1\}$) reads amplitudes $diag + \mathbf{1}_K - \mathbf{1}_i$ and writes amplitudes $diag + \mathbf{1}_K + \mathbf{1}_i$, where $i \in \{1, 2, 3, ..., 2M\}$. In a similar way it can be shown that an off-diagonal pivot only reads and writes pivots of the type $diag + offset$, where $offset$ is one of the following types:

- $offset_0 = \mathbf{0}$
- $offset_2 = 2 \cdot \mathbf{1}_K$ ($K \in \{1, 3, 5, ..., 2M-1\}$)
- $offset_{1010} = \mathbf{1}_K + \mathbf{1}_i$ ($K \in \{1, 3, 5, ..., 2M-1\}$ and $i \in \{K+2, K+4, ..., 2M-1\}$)
- $offset_{1001} = \mathbf{1}_K + \mathbf{1}_i$ ($K \in \{1, 3, 5, ..., 2M-1\}$ and $i \in \{K+3, K+5, ..., 2M\}$)

Note that $offset_{0110} = \mathbf{1}_K + \mathbf{1}_i$ ($K \in \{1, 3, 5, ..., 2M-1\}$ and $i \in \{2, 4, 6, ..., K-1\}$) does not occur. As is clear from line 8 in Algorithm 1, off-diagonal pivots $diag + \mathbf{1}_K$ ($K \in \{1, 3, 5, ..., 2M-1\}$) always satisfy $diag_i = 0$ for $i \in \{1, 2, ..., 2(K-1)\}$. When reading the required amplitudes for an off-diagonal pivot, indices $k_1, k_2, ..., k_{2(K-1)}$ therefore do not need to be lowered.

This parametrization allows all calculated amplitudes to be stored in a structured way, without storing zero values for amplitudes that do not occur in Algorithm 1. It can be shown that each amplitude in this structure is written exactly once. In other words, the structure is fully dense and there are no two pivots writing to the same position in the Fock lattice. As is explained in Section 3.1.4, it can also be shown that every off-diagonal amplitude is included in the 'read' group of a pivot exactly once, after which it can be removed from memory.

B.3 Total number of written amplitudes

From Algorithm 1, it is clear that a diagonal pivot writes at most $2M$ values, while an off-diagonal pivot $diag + \mathbf{1}_K$ writes at most $2(M-K)$ amplitudes. The actual number of written amplitudes is determined by invoking the boundary condition $\mathbf{k} < cutoffs$. Assuming the cutoffs in all modes to be equal to C, a deeper analysis shows that Algorithm 1 writes the following number of amplitudes:

- $(C-1)C^{M-1} + (C-2)C^{M-1} + (2M-2)(C-1)^2C^{M-2}$ of the type $diag + offset_1$
- C^M of the type $diag + offset_0$ ($= diag$)
- $(C-2)\sum_{K=0}^{M-1} C^{M-K-1}$ of the type $diag + offset_2$
- $(C-1)^2\sum_{K=0}^{M-1}(M-K-1)C^{M-K-2}$ of the type $diag + offset_{1010}$
- $(C-1)^2\sum_{K=0}^{M-1}(M-K-1)C^{M-K-2}$ of the type $diag + offset_{1001}$

As is also shown in Fig. 10(a), the total number of amplitudes scales as C^M. We therefore drastically reduce the memory requirements of density matrix simulations compared to the naive strategy of calculating all C^{2M} Fock amplitudes.

C Scaling behaviour of Equation (20)

In this section we show that our state vector algorithm is faster than the algorithm of Reference [9] at calculating the probabilities of all PNR outcomes (up to a certain cutoff) of a GBS circuit. The complexity of our algorithm is given by $M^2 \prod_{i=1}^{M} C_i$. If we apply the algorithm of Reference [9] to all detection patterns (without using the batched strategy that was discussed in Section 4.2), then its complexity is lower bound by Eq. (20). We assume the cutoff conditions to be constant for all modes and put $\widetilde{C} = C-1$ for ease of notation. Eq. (20) is then further lower bound by:

$$\sum_{\boldsymbol{n}=\boldsymbol{0}}^{cutoffs} N^3 = \sum_{N=0}^{\widetilde{C}M} \binom{N+M-1}{N} N^3 \tag{23}$$

$$= \frac{\widetilde{C}M(\widetilde{C}M+1)(\widetilde{C}^2M^4+3\widetilde{C}^2M^3+\widetilde{C}(2\widetilde{C}+3)M^2+(3\widetilde{C}-1)M+1)\binom{\widetilde{C}M+M}{\widetilde{C}M+1}}{(M+1)(M+2)(M+3)} \tag{24}$$

$$\geq \frac{\widetilde{C}^4M^6\binom{\widetilde{C}M+M}{\widetilde{C}M+1}}{(M+3)^3} \geq \frac{M^6\widetilde{C}^{M+3}}{(M+3)^3} \propto M^3C^{M+3} \ . \tag{25}$$

If we do apply the batched strategy for one of the modes, Eq. (20) can be lowered by fixing the value of n_1 to $\widetilde{C}_1 = C_1 - 1$. Consequently, the left part of Eq. (23) becomes:

$$\sum_{\boldsymbol{n}=\boldsymbol{0}}^{[\widetilde{C}_2,\widetilde{C}_3,\ldots,\widetilde{C}_M]} (\widetilde{C}_1+N')^3. \tag{26}$$

Assuming the cutoffs to be constant in all modes, we find in a similar way that this is equal to:

$$\sum_{N'=0}^{\widetilde{C}(M-1)} \binom{N'+M-2}{N'} (N')^3 \geq \frac{(M-1)^6\widetilde{C}^{M+2}}{(M+2)^3} \propto M^3C^{M+2} \ , \tag{27}$$

where $N' = \sum_{i=2}^{M} n_i$.