

# Simulating a Boson Sampler

Ezequiel E. Carrasco

Department of Physics and Astronomy  
University of New Mexico

April 26, 2014

## Abstract

The ideal boson sampler is a linear interferometer that demonstrates interesting nonclassical optical phenomenon and multi-photon interference. This quantum behavior cannot be simulated using a traditional random sampling approach, and thus it is shown that detecting the output from a such an interferometer samples from probability distribution in a way that no classical computer can efficiently simulate. While not a universal quantum computer, such a device would demonstrate that machines acting under the laws of quantum mechanics have potential computation power beyond a classical Turing machine. A case is presented to show why sampling from the probability distribution generated by a boson sampler is difficult and ultimately intractable for a classical computer.

## 1 Introduction

Computers today are classical machines that obey the deterministic physics of everyday life. However, if a computer could be constructed that exploits the effects of quantum mechanics, there is potential that such a device could perform some computations more efficiently than a classical computer.[8] Unfortunately, quantum devices are very delicate, sensitive to error and environment, and their construction is very demanding. As a stepping stone to a legitimate universal computer, the ideal boson sampler has the potential to demonstrate a “quantum advantage” while not performing universal quantum computation.

An ideal boson sampler as proposed by Aaronson[1] is an interferometer of lossless, linear optical elements arranged such that photons entering the device are scattered in ways that demonstrate interesting and nonintuitive quantum effects. The relationship of such an array of beam splitters and phase shifters to a unitary transformation on photon modes is developed as described by Reck. [9]

The boson sampler scatters a nonclassical state of light known as a Fock state with multiple but definite numbers of photons in multiple modes into a superposition of possible output states, each with their own complex amplitude. If the boson sampler is considered ideal and the photons are indistinguishable, the probabilities of observed output states are equal to the absolute square of these complex amplitudes of multi-particle states. The photons entering the boson sampler are considered to be indistinguishable, and hence the probability of observed outputs are the result of multi-particle interference. Due to this multi-photon interference we can only define the amplitudes of a coherent, multi-particle states and not the individual photon's amplitudes. This complication hinders the ability to run a simulation that is available for distinguishable photons and classical analogs of the boson sampler.

A computer model is proposed that takes arbitrary input and generates pseudorandom output that is taken from the calculated probability distribution of an ideal boson sampler. As the probability distribution of the coherent quantum states is inherently nonclassical, it is difficult for a classical computer to simulate efficiently. Each probability involves considering the interfering amplitudes of all permutations of the photons within the device, which can be compactly represented by the permanent of a complex matrix [11].

As the number of photons and modes increase, the Fock space increases exponentially, and the difficulty of using the permanent to calculate each statistical weight of these output states make such a classical simulation intractable. Moreover, the permanent itself cannot be calculated or efficiently approximated as the number of photons grow [13]. The boson sampler, as an optical interferometer, naturally obeys the laws of quantum mechanics. The boson sampler can thus be considered an analog machine that allows sampling from this immensely complex probability distribution. As simulating the boson sampler can use exorbitant computer resources, the sampling of a distribution taken from a physical boson sampler is an example of an analog quantum machine that can perform a specific but limited task virtually instantaneously and with few resources that a classical computer cannot efficiently mimic.

A discussion of the finite-dimensional subspace of the Hilbert space relevant to the boson sampler, the Fock space, is examined and its complexity analyzed. In addition, we present the processes by which photons prepared in a pure Fock state are coherently scattered by the boson sampler and describe method to calculate the amplitude of each pure Fock state in the superposition. Using these tools, we propose a computer simulation that will generate random output from an ideal boson sampler, and its efficiency and scaling is explored.

## 2 Foundations

In this section we will discuss how the input states of a boson sampler are prepared, and how the system of beam splitters and phase shifters act together to scatter the photon modes. Using these tools, the nonclassical probability distribution that defines the boson sampler

is derived. Many of the key ideas were reviewed by Tichy[12] whose work serves as the foundation for the ideas presented.

## 2.1 Scattering Matrix Transformation of Modes

Consider  $N > 1$  sources of photons and  $N$  detectors. Each source can simultaneously create definite, known numbers of indistinguishable, noninteracting photons in modes identical to each other in every way (frequency, temporal shape, polarization, and so forth), except for the path the photons take upon leaving the source. Further, each detector is considered a perfect counter of photons, only sensitive to the numbers of photons in the mode, and does not need to be sensitive to any other aspect of the state. We define  $j = 1, 2, \dots, N$ , where the  $j^{\text{th}}$  source creates  $n_j$  photons traveling towards the  $j^{\text{th}}$  detector.

Let  $n$  be the total number of photons that all the sources create. As the initial multi-mode, multi-particle state for the system has a definite number of photons, we can consider a finite subspace of the Hilbert space, the Fock space. For a given  $N$  and  $n$ , there are  $A$  orthogonal Fock states that form a basis for the Hilbert space all the possible initial input states  $|\Psi^{(\alpha)}\rangle = |n_1^{(\alpha)}, n_2^{(\alpha)}, \dots, n_N^{(\alpha)}\rangle$  of the system, where  $1 \leq \alpha \leq A$  and, for any  $\alpha$ ,

$$\sum_{j=1}^N n_j^{(\alpha)} = n. \quad (1)$$

$A$  is equal to number of different ways  $n$  indistinguishable photons can be sorted into  $N$  distinguishable modes, which is given by Feller[2],

$$A = \binom{N+n-1}{n} = \binom{N+n-1}{N-1} = \frac{(N+n-1)!}{n!(N-1)!}. \quad (2)$$

$A$  is also equal to the dimensionality of the Fock space that is spanned by all  $A$  state vectors. If the system is lossless and all  $n$  photons arrive at the  $N$  detectors, then any observed output state  $|\Psi^{(\beta)}\rangle$  is also a member of the same Fock space as the input state  $|\Psi^{(\alpha)}\rangle$ .

We consider sending photons through a linear interferometer as shown in Fig. 1. The distance of each of the  $N$  paths from each source to the corresponding detector is the same. All  $N$  sources are assumed to have created their photons at the same instant, and thus all  $n$  photons arrive at the detectors at the same time. A lossless mirror is placed in the paths of the photons so that the reflected paths cross at least one other. The  $p^{\text{th}}$  path crosses the  $q^{\text{th}}$  path, where  $p < q$ , at a point  $\Omega^{(p,q)}$  as shown in Fig. 1. Beam splitters and phase shifters are placed at the points  $\Omega^{(p,q)}$  as shown in Fig. 2. Lastly, another set of  $N$  phase shifters is placed after the points  $\Omega^{(p,N)}$  that act on the modes before they enter the detectors as seen in Fig. 3. All these beam splitters and phase shifters scatter and transform the  $N$  modes.

Let  $\hat{a}_j^\dagger$  be the creation operator for one photon that leaves the the  $j^{\text{th}}$  source in the  $j^{\text{th}}$  mode. Then we have,

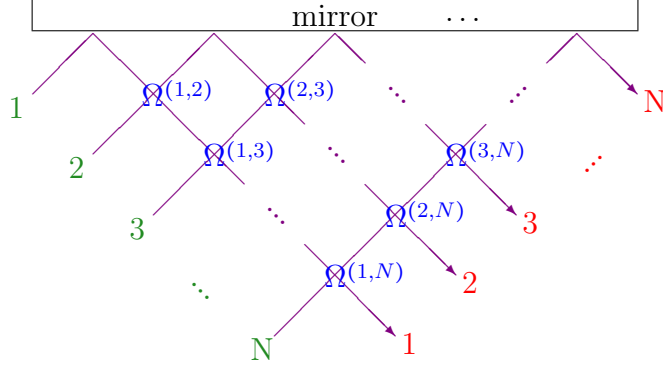


Figure 1: A depiction of the photons' paths and their crossing points  $\Omega^{(p,q)}$  shown in blue. Photon sources are shown in green, detectors are shown in red and the paths are shown in violet.

$$|\Psi^{(\alpha)}\rangle = |n_1^{(\alpha)}, n_2^{(\alpha)}, \dots, n_N^{(\alpha)}\rangle = \left( \prod_{j=1}^N \frac{(\hat{a}_j^\dagger)^{n_j^{(\alpha)}}}{\sqrt{n_j^{(\alpha)}!}} \right) |0\rangle, \quad (3)$$

where  $|0\rangle$  is an  $A$ -dimensional subspace of the vacuum state.

The beam splitter and phase shifter arrangements placed at the points  $\Omega^{(p,q)}$  and the phase shifters placed after all  $\Omega^{(p,q)}$  collectively perform a single lossless, unitary scattering transformation  $\hat{\mathbb{S}}$  on  $|\Psi^{(\alpha)}\rangle$ . A matrix representation of  $\hat{\mathbb{S}}$  has elements  $\mathbb{S}_{\beta,\alpha}$  given by

$$\mathbb{S}_{\beta,\alpha} = \langle \Psi^{(\beta)} | \hat{\mathbb{S}} | \Psi^{(\alpha)} \rangle. \quad (4)$$

As  $N$  and  $n$  grow,  $A$  can become quite large and creating an  $A \times A$  matrix representation of  $\hat{\mathbb{S}}$  can be intractable. Instead, we can look at a unitary transformation of the  $N$ -input modes  $\hat{a}_j^\dagger$ . We represent this new transformation as an  $N \times N$  complex unitary matrix  $\Lambda$  that acts on the  $N$ -dimensional space of modes.

Because the photons are non interacting and the interferometer is *linear* we can immediately write a solution to the scattering evolution. We let  $k = 1, 2, \dots, N$  as for  $j$  and define the elements of the unitary matrix  $\Lambda$  in the following fashion,

$$\hat{\mathbb{S}} \hat{a}_j^\dagger \hat{\mathbb{S}}^\dagger = \sum_{k=1}^N \Lambda_{k,j} \hat{a}_k^\dagger. \quad (5)$$

Thus the input modes  $\hat{a}_j^\dagger$  evolve into a superposition of all modes with complex amplitudes  $\Lambda_{j,k}$ . We can now see how  $|\Psi^{(\alpha)}\rangle, |\Psi^{(\beta)}\rangle, \hat{a}_j^\dagger, \hat{\mathbb{S}}$ , and  $\Lambda$  are all related. We can define the coherent output state  $|\Psi_{out}^{(\alpha)}\rangle$  as,

$$|\Psi_{out}^{(\alpha)}\rangle = \hat{S} |\Psi^{(\alpha)}\rangle = \sum_{\beta=1}^A \mathbb{S}_{\beta,\alpha} |\Psi^{(\beta)}\rangle = \left( \prod_{j=1}^N \frac{(\hat{S} \hat{a}_j^\dagger \hat{S}^\dagger)^{n_j^{(\alpha)}}}{\sqrt{n_j^{(\alpha)}!}} \right) |0\rangle = \left( \prod_{j=1}^N \frac{1}{\sqrt{n_j^{(\alpha)}!}} \left( \sum_{k=1}^N \Lambda_{j,k} \hat{a}_k^\dagger \right)^{n_j^{(\alpha)}} \right) |0\rangle. \quad (6)$$

After  $|\Psi^{(\alpha)}\rangle$  has evolved under  $\hat{S}$ , the output state is in a superposition of all states within the Fock space. The probability of observing a particular  $|\Psi^{(\beta)}\rangle$  given by  $|\mathbb{S}_{\beta,\alpha}|^2$  is ultimately related to the matrix  $\Lambda$  and the input state  $|\Psi^{(\alpha)}\rangle$ .

## 2.2 Transformations at the Crossing Points

Now we will look at the transformations that occur at the crossing points  $\Omega^{(p,q)}$  as presented by Reck[9]. To index all crossing points without overcounting, we let  $1 \leq p \leq N-1$  and  $p+1 \leq q \leq N$ . There are  $Z$  total crossing points, where

$$Z = \sum_{p=1}^{N-1} \sum_{q=p+1}^N 1 = \binom{N}{2} = \frac{N(N-1)}{2}. \quad (7)$$

At each point  $\Omega^{(p,q)}$ , a lossless beam splitter with transmittance  $T^{(p,q)}$  and reflectivity  $R^{(p,q)}$  is placed, where  $\{T^{(p,q)}, R^{(p,q)}\} \in \mathbb{R}$  and  $T^{(p,q)} + R^{(p,q)} = 1$ . In addition, a lossless phase shifter with associated phase shift  $\varphi^{(p,q)}$  where  $0 \leq \varphi^{(p,q)} < 2\pi$  is placed along the  $p^{th}$  path before the crossing point. Thus as photons approach and pass through  $\Omega^{(p,q)}$ , the mode associated with the  $p^{th}$  path acquires a phase shift equal to  $\varphi^{(p,q)}$ , and photons from both paths superpose and acquire a phase shift associated with the beam splitter.

The system found at  $\Omega^{(p,q)}$  can be regarded as a 2x2 subsystem of the whole with modes entering and exiting the system. Let  $\hat{a}_p^\dagger$  be the creation operator for the input mode approaching  $\Omega^{(p,q)}$  from the and leaving by the  $p^{th}$  path, and  $\hat{a}_q^\dagger$  be the creation operation for the input mode on the  $q^{th}$  path. A depiction of the system found at  $\Omega^{(p,q)}$  is shown in Fig. 2.

As the modes pass through  $\Omega^{(p,q)}$ , they are transformed. To examine this, we consider a two-element column vector  $\begin{pmatrix} \hat{a}_p^\dagger \\ \hat{a}_q^\dagger \end{pmatrix}$  valued with the input modes and apply a 2x2 matrix to this mode-valued vector. The first transformation to apply will be the lossless phase shifter acting on mode  $\hat{a}_p^\dagger$  while preserving mode  $\hat{a}_q^\dagger$ . This action can be represented by the 2x2 unitary matrix  $\Phi^{(p,q)}$ ,

$$\Phi^{(p,q)} = \begin{pmatrix} e^{i(\varphi^{(p,q)})} & 0 \\ 0 & 1 \end{pmatrix}. \quad (8)$$

Next, the action of the beam splitter on both modes is applied and operates after the phase shift. As the beam splitter is assumed to be lossless, the matrix  $\Theta^{(p,q)}$  associated with the action of the beam splitter will also be unitary. Ignoring the first transformation of the

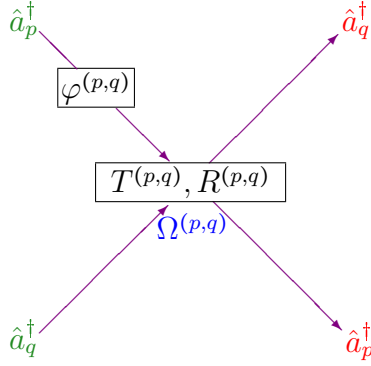


Figure 2: A depiction of the placement of the phase shifter with phase shift  $\varphi^{(p,q)}$  and symmetric beam splitter with transmittance  $T^{(p,q)}$  and reflectivity  $R^{(p,q)}$  placed at the crossing point  $\Omega^{(p,q)}$ . The **input** modes are shown in green and the **output** modes are shown in red.

phase shifter on the input modes for now (i.e. let  $\varphi^{(p,q)} = 0$  and therefore  $\Phi^{(p,q)} = \mathbb{I}$ ), the transformation of  $\Theta^{(p,q)}$  on the input modes will take the form,

$$\Theta^{(p,q)} \begin{pmatrix} \hat{a}_p^\dagger \\ \hat{a}_q^\dagger \end{pmatrix} = \begin{pmatrix} t_{pp} & r_{pq} \\ r_{qp} & t_{qq} \end{pmatrix} \begin{pmatrix} \hat{a}_p^\dagger \\ \hat{a}_q^\dagger \end{pmatrix} = \begin{pmatrix} t_{pp}\hat{a}_p^\dagger + r_{pq}\hat{a}_q^\dagger \\ r_{qp}\hat{a}_p^\dagger + t_{qq}\hat{a}_q^\dagger \end{pmatrix} \quad (9)$$

where  $\{t_{pp}, r_{pq}, r_{qp}, t_{qq}\} \in \mathbb{C}$ . Eq. 9 shows how the action of the beam splitter defines the output modes as a superposition of the input modes with complex amplitudes. As  $\Theta^{(p,q)}$  is unitary, these amplitudes are normalized (there are no losses due to absorption, etc.). The amplitude  $t_{pp}$  can be seen as the amplitude of mode  $\hat{a}_p^\dagger$  that is transmitted and  $r_{pq}$  as its reflected amplitude. Similarly,  $t_{qq}$  is the transmitted amplitude of mode  $\hat{a}_q^\dagger$  and  $r_{qp}$  its reflected amplitude.

Using the definition of unitary matrix,  $(\Theta^{(p,q)})^\dagger \Theta^{(p,q)} = \mathbb{I}$ , and also using that for a complex number  $z$ ,  $z^*z = zz^* = |z|^2$ , we demand that,

$$\begin{pmatrix} t_{pp}^* & r_{qp}^* \\ r_{pq}^* & t_{qq}^* \end{pmatrix} \begin{pmatrix} t_{pp} & r_{pq} \\ r_{qp} & t_{qq} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \Rightarrow \begin{cases} |t_{pp}|^2 + |r_{pq}|^2 = |t_{qq}|^2 + |r_{qp}|^2 = 1 \\ t_{pp}r_{qp}^* + t_{qp}^*r_{pq} = t_{pp}^*r_{qp} + t_{qq}r_{pq}^* = 0 \end{cases} \quad (10)$$

If we allow  $t_{pp} = t_{qq} = \sqrt{T^{(p,q)}}$  and  $r_{pq} = \sqrt{R^{(p,q)}}$ ,  $r_{qp} = -\sqrt{R^{(p,q)}}$ , then these relationships in Eq. 10 are satisfied. There are many solutions to force the matrix to be unitary, and this is only one choice. Further, note that while individual photons approaching on either the  $p^{th}$  or the  $q^{th}$  path have the same chance of transmission or reflection, a photon on the  $q^{th}$  path that is reflected to the  $p^{th}$  path acquires a  $\pi$  phase shift, thus the beam splitter does not affect photons on the  $p^{th}$  and the  $q^{th}$  path in quite the same fashion.

In order to simplify, let  $T^{(p,q)} = \cos^2(\theta^{(p,q)})$  and  $R^{(p,q)} = \sin^2(\theta^{(p,q)})$ , where  $0 \leq \theta^{(p,q)} < 2\pi$ . Then we have,

$$\Theta^{(p,q)} = \begin{pmatrix} \sqrt{T^{(p,q)}} & \sqrt{R^{(p,q)}} \\ -\sqrt{R^{(p,q)}} & \sqrt{T^{(p,q)}} \end{pmatrix} = \begin{pmatrix} \cos(\theta^{(p,q)}) & \sin(\theta^{(p,q)}) \\ -\sin(\theta^{(p,q)}) & \cos(\theta^{(p,q)}) \end{pmatrix}. \quad (11)$$

Let  $\chi^{(p,q)}$  be the 2x2 matrix representation of the ordered action of the phase shifter and then the beam splitter operating on the input modes. As the product of two unitary matrices,  $\chi^{(p,q)}$  will also be unitary. The matrix  $\chi^{(p,q)}$  is represented as,

$$\chi^{(p,q)} = \Theta^{(p,q)}\Phi^{(p,q)} = \begin{pmatrix} \cos(\theta^{(p,q)}) & \sin(\theta^{(p,q)}) \\ -\sin(\theta^{(p,q)}) & \cos(\theta^{(p,q)}) \end{pmatrix} \begin{pmatrix} e^{i\varphi^{(p,q)}} & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} e^{i\varphi^{(p,q)}} \cos(\theta^{(p,q)}) & \sin(\theta^{(p,q)}) \\ -e^{i\varphi^{(p,q)}} \sin(\theta^{(p,q)}) & \cos(\theta^{(p,q)}) \end{pmatrix}. \quad (12)$$

We can ultimately see that the complete transformation  $\chi^{(p,q)}$  operating on the input modes at  $\Omega^{(p,q)}$  can be described with only two parameters,  $\theta^{(p,q)}$  and  $\varphi^{(p,q)}$ , and its action on the modes can be seen as a rotation by two angles in the complex space of modes. Each  $\chi^{(p,q)}$  is therefore a member of the rotation group  $U(2)$ .

We can now describe the action of a beam splitter and phase shifter arrangement at  $\Omega^{(p,q)}$  on all  $N$  modes. Let  $\lambda^{(p,q)}$  be an  $N \times N$  matrix that acts on  $N$  modes. To construct  $\lambda^{(p,q)}$ , we first let  $\lambda^{(p,q)} = \mathbb{I}$ , then we replace four elements of this identity matrix with the four elements of  $\chi^{(p,q)}$  in the following fashion,

$$\begin{cases} \lambda_{pp}^{(p,q)} = \chi_{1,1}^{(p,q)} \\ \lambda_{pq}^{(p,q)} = \chi_{1,2}^{(p,q)} \\ \lambda_{qp}^{(p,q)} = \chi_{2,1}^{(p,q)} \\ \lambda_{qq}^{(p,q)} = \chi_{2,2}^{(p,q)} \end{cases}. \quad (13)$$

This construction ensures that the matrix  $\lambda^{(p,q)}$  acts only on the  $p^{th}$  and  $q^{th}$  modes while preserving all the other  $N - 2$  modes.

### 2.3 Transformations After Crossing Points

After all the crossing points  $\Omega^{(p,q)}$ , another set of  $N$  phase shifters is placed before each detector, each with phase shift  $\psi_j$ . A diagram of the placement of these phase shifters is shown in Fig. 3. These phase shifters ensure that we can implement an arbitrary linear scattering matrix represented by  $\Lambda$  in Eq. 6 as described in the Appendix A.

The final operation on the  $N$  modes after the operations of  $Z$  matrices  $\lambda^{(p,q)}$  is the final set of phase shifters with phase shift  $\psi_j$ , where  $0 \leq \psi_j < 2\pi$ . This can be represented by a diagonal  $N \times N$  matrix  $\mathbb{Y}$  that has the form,

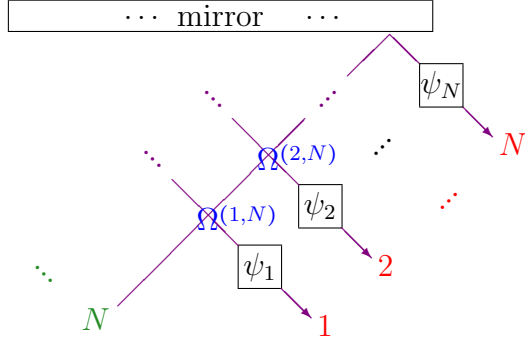


Figure 3: The placement of the set of  $N$  phase shifters with phase shift  $\psi_j$ .

$$\mathbb{Y} = \begin{pmatrix} e^{i\psi_1} & 0 & \dots & 0 \\ 0 & e^{i\psi_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & e^{i\psi_N} \end{pmatrix}. \quad (14)$$

The structure of this matrix ensures that the phase shift  $\psi_j$  only acts on the  $j^{\text{th}}$  mode. The action of  $\mathbb{Y}$  can be seen as a rotation in the complex space of modes by an angle  $\psi_j$ . This action can be reversed and is given by  $\mathbb{Y}^{-1} = \mathbb{Y}^\dagger$ , where  $\psi_j \rightarrow -\psi_j$ .

## 2.4 Complete Transformations on All Modes

The combined, ordered operation of all  $\lambda^{(p,a)}$  matrices and the final  $\mathbb{Y}$  transformation serve to describe the single matrix  $\Lambda$  that acts on the initial, original input modes as presented in Eq. 5. The  $N \times N$  matrix  $\Lambda$  becomes,

$$\Lambda = \mathbb{Y}[\lambda^{(N-1,N)}][\lambda^{(N-2,N)}\lambda^{(N-2,N-1)}] \dots [\lambda^{(1,N)}\lambda^{(1,N-1)} \dots \lambda^{(1,3)}\lambda^{(1,2)}], \quad (15)$$

where the brackets indicate grouping by the  $p^{\text{th}}$  modes. To further emphasize this grouping, let the matrix  $\mathbb{L}^{(p)}$  be defined as,

$$\mathbb{L}^{(p)} = \lambda^{(p,N)}\lambda^{(p,N-1)} \dots \lambda^{(p,p+2)}\lambda^{(p,p+1)}. \quad (16)$$

The matrix  $\mathbb{L}^{(p)}$  can be seen as a rotation upon a single vector as discussed in the Appendix A.

## 2.5 Interference Between Two Indistinguishable Photons

The unique behavior seen within the boson sampler comes from the interference among the indistinguishable photons. To explore this, we consider the simplest example of multi-photon interference with only two modes and one photon in each mode and a 50-50 beam splitter that scatters both photons as they interact with it at the same time. This situation is illustrated in Fig. 4.

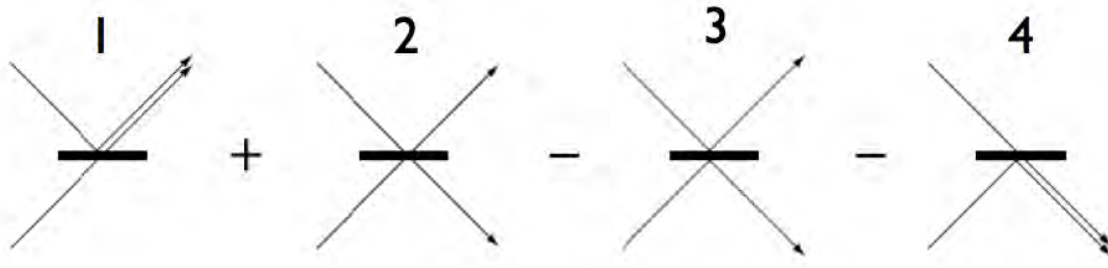


Figure 4: Depiction of the four scattering possibilities of two photons each in their own mode encountering a beam splitter. This image was created by Kok.[6]

The meanings of the signs seen in the Fig. 4 will be discussed later. First consider the case of distinguishable photons, which give classical results. There are four possibilities to consider. The possibility labeled 1 where the photons bunch and head towards the top detector is equal to  $0.5 \times 0.5 = 0.25$ , as there is a 50% chance of the top photon reflecting, and an independent 50% chance of the bottom photon transmitting. There is a 25% chance both photons transmit as in possibility 2, and 25% chance both reflect as in possibility 3. Note that these are not the same outcomes, as each photon is distinguishable and therefore it is possible to tell which photon went to which detector. However, the chance that each detector counts one photon each is  $0.25 + 0.25 = 0.5$ , and is the most commonly seen outcome. Lastly, there is a 25% chance to see possibility 4, where the photons bunch and land in the bottom detector.

Now we send in indistinguishable photons, and the phases associated with these possibilities are important. Let us define the scattering matrix  $\Lambda$  for this situation as,

$$\Lambda = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}. \quad (17)$$

For possibility 1, the top photon reflects with a positive amplitude  $\Lambda_{1,2} = \frac{1}{\sqrt{2}}$ , and the bottom acquires  $\Lambda_{2,2} = \frac{1}{\sqrt{2}}$ , and so the overall phase is positive. For possibility 2, both transmit, and the photons receive the phases on the diagonal, and so the product of the photons' phases is positive. For possibility 3, the photons receive the phases of the off-diagonal elements, and therefore the phase for this possibility is negative. Lastly, for possibility 4, the top photon receives amplitude  $\Lambda_{1,1} = \frac{1}{\sqrt{2}}$  and the bottom obtains  $\Lambda_{2,1} = -\frac{1}{\sqrt{2}}$ , and the phase for this

bunching outcome is negative.

If the photons are indistinguishable and each detector counts one photon each, *there is no way to determine if possibility 2 or 3 was detected*. In this regard, possibilities 2 and 3 are members of the same state, and thus their amplitudes sum. In the case of a 50-50 beam splitter, this indicates that the amplitudes, one positive and one negative, completely cancel. Therefore, in this finely-tuned case, possibilities 2 and 3 are never encountered. This complete suppression of this output state is known as the Hong-Ou-Mandel effect[4].

To analyze the mathematics of this phenomenon in a more general sense, we let  $N = n = 2$ , use no phase shifter, and use the transmittivity  $T$  and reflectivity  $R$  as the elements of the  $2 \times 2$  matrix  $\Lambda$ ,

$$\Lambda = \begin{pmatrix} \sqrt{T} & \sqrt{R} \\ -\sqrt{R} & \sqrt{T} \end{pmatrix} \quad (18)$$

The input state is given by,

$$|\Psi^{(\alpha)}\rangle = \hat{a}_1^\dagger \hat{a}_2^\dagger |0\rangle = |1, 1\rangle. \quad (19)$$

Applying the scattering operator  $\hat{\mathbb{S}}$  and using the identity  $\hat{\mathbb{S}}^\dagger \hat{\mathbb{S}} = \mathbb{I}$ , we have,

$$\hat{\mathbb{S}} |1, 1\rangle = \hat{\mathbb{S}} \hat{a}_1^\dagger \hat{a}_2^\dagger |0\rangle = \hat{\mathbb{S}} \hat{a}_1^\dagger \hat{\mathbb{S}}^\dagger \hat{\mathbb{S}} \hat{a}_2^\dagger \hat{\mathbb{S}} |0\rangle. \quad (20)$$

Noting that  $\hat{\mathbb{S}} |0\rangle = |0\rangle$  and using Eq. 5, we can rewrite Eq. 20 as,

$$\begin{aligned} (\hat{\mathbb{S}} \hat{a}_1^\dagger \hat{\mathbb{S}}^\dagger) (\hat{\mathbb{S}} \hat{a}_2^\dagger \hat{\mathbb{S}}^\dagger) |0\rangle &= \left( \sqrt{T} \hat{a}_1^\dagger - \sqrt{R} \hat{a}_2^\dagger \right) \left( \sqrt{R} \hat{a}_1^\dagger + \sqrt{T} \hat{a}_2^\dagger \right) |0\rangle \\ &= \sqrt{TR} (\hat{a}_1^\dagger)^2 |0\rangle - \sqrt{TR} (\hat{a}_2^\dagger)^2 |0\rangle + T \hat{a}_1^\dagger \hat{a}_2^\dagger |0\rangle - R \hat{a}_2^\dagger \hat{a}_1^\dagger |0\rangle. \end{aligned} \quad (21)$$

In the case of distinguishable photons, the operators acting in the order  $\hat{a}_1^\dagger \hat{a}_2^\dagger$  with amplitude  $T$  create a state  $|1, 1\rangle_T$  that is distinguishable from  $|1, 1\rangle_R$  created by the operators  $\hat{a}_2^\dagger \hat{a}_1^\dagger$  with amplitude  $-R$ . This stems from the fact that as the photons are distinguishable, it is possible to tell if both photons transmitted or both photons reflected when  $|1, 1\rangle$  is detected, as mentioned previously. For example, if the top photon is “red” and the bottom is “blue,” then if the top detector sees red and the bottom blue, then we know both photons reflected. The coherent output state for distinguishable photons  $|1, 1\rangle_{dis}$  is then given by,

$$\hat{\mathbb{S}} |1, 1\rangle_{dis} = \sqrt{TR} |2, 0\rangle - \sqrt{TR} |0, 2\rangle + T |1, 1\rangle_T - R |1, 1\rangle_R. \quad (22)$$

The probabilities of seeing each output state is given by the absolute square of the amplitudes,

$$\begin{cases} |\langle 2, 0 | \hat{\mathbb{S}} |1, 1\rangle_{dis} |^2 = TR \\ |\langle 0, 2 | \hat{\mathbb{S}} |1, 1\rangle_{dis} |^2 = TR \\ |\langle 1, 1 | \hat{\mathbb{S}} |1, 1\rangle_{dis} |^2 = T^2 + R^2 \end{cases} \quad (23)$$

However, if the photons are indistinguishable, then there is only one  $|1, 1\rangle$  state, and the final probabilities are different. The amplitudes  $T$  and  $-R$  interfere with each other and are grouped and summed together. If  $\hat{S}|1, 1\rangle_{ind}$  is the coherent output state for indistinguishable photons, then,

$$\hat{S}|1, 1\rangle_{ind} = \sqrt{2TR}|2, 0\rangle - \sqrt{2TR}|0, 2\rangle + (T - R)|1, 1\rangle \quad (24)$$

The probabilities for the indistinguishable case are given by,

$$\begin{cases} |\langle 2, 0 | \hat{S} | 1, 1 \rangle_{ind}|^2 = 2TR \\ |\langle 0, 2 | \hat{S} | 1, 1 \rangle_{ind}|^2 = 2TR \\ |\langle 1, 1 | \hat{S} | 1, 1 \rangle_{ind}|^2 = (T - R)^2 \end{cases} \quad (25)$$

The interference between photons that changes the probability distributions from the distinguishable case to the indistinguishable as seen by comparing the two sets give rise to the interesting behavior of photons traveling through a boson sampler. As discussed above, if a 50-50 beam splitter is used for indistinguishable photons, where  $T = R = \frac{1}{2}$ , then it can be seen that the amplitude for the  $|1, 1\rangle$  state goes to zero for the indistinguishable photons and thus is never observed.

To see the Hong-Ou-Mandel effect, the photons must arrive at the beam splitter at the same time, and their respective wave functions must be identical except for the paths they follow. However, we can assume that they arrive at slightly different times, and interesting effects can be seen. If two photons are slightly displaced in time, there is a chance that either the distinguishable or the indistinguishable probability distributions will be observed. The farther apart in time the photons are separated, the more chance there is of not seeing the Hong-Ou Mandel effect. This can be graphically illustrated in a plot of the ‘‘HOM Dip.’’

Fig. 5 shows the continuum between the case where there is zero chance of seeing state  $|1, 1\rangle$  in the indistinguishable case and the 50% chance of seeing  $|1, 1\rangle$  as with distinguishable photons. When the difference of arrival times is around zero, the photons are more likely to be observed as indistinguishable, and the output state  $|1, 1\rangle$  is suppressed. As the photons arrive more and more farther apart in time, then the chance of both detectors counting a photon becomes the most likely outcome. The HOM Dip demonstrates the need for tight control of all aspects of the photons prepared and sent through the interferometer. As the photons become more distinguishable, the probability distribution changes.

## 2.6 Output State Probabilities

This section details how to calculate the probability weight of possible output states for distinguishable and indistinguishable photons being scattered by a boson sampler, and how these calculations compare to those of a classical example.

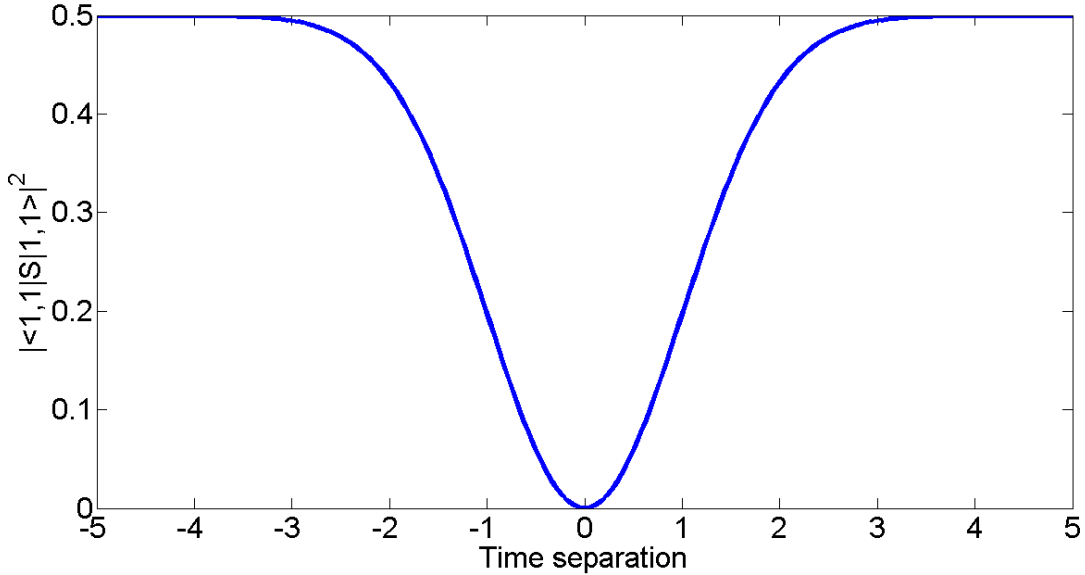


Figure 5: Plot of the HOM Dip. The time separation axis is shown in units of  $\Delta t$  of the spread of the Gaussian amplitude of the photons’ temporal shapes.

### 2.6.1 The Classical Analog, Distinguishable Photons and the Permanent

To demonstrate how the probabilities of observing possible output states of a boson sampler compare to other distributions, first consider a crude classical analog. Imagine a pinball-like machine that takes  $n$  total ball bearings sorted into  $N$  launchers and sends the ball bearings, one at a time, along paths that cross each other. At the crossing points  $\Omega^{(p,q)}$ , a computer-controlled gate-like device randomly “transmits” a ball bearing, keeping it on the same path, or “reflects” it, sending it from the  $p^{th}$  path to the  $q^{th}$  path or vice versa. There are no analogs for the phase shifters as complex phase has no meaning in this context. Further, these ball bearings are considered to be non-interacting, for instance they do not bounce off each other, and if two or more ball bearing enter a gate at the same time, it does not affect how the gates affect their individual paths. At the end of the balls’ paths are bins that collect the balls for counting.

Like for the boson sampler, there is a number-space of ball bearings that includes the input number state of  $n$  ball bearings spread out among  $N$  launchers, and  $A$  possible output number states that is also equal to the number of different ways of sorting  $n$  balls into  $N$  bins as in Eq. 2.

A method for determining the probability of a classical ball bearing starting from the  $k^{th}$  launcher and being collected in the  $j^{th}$  bin can be found by constructing a matrix that is the analog to the modal transformation matrix  $\Lambda$ , which we will call  $\mathbb{C}$ . As with the photons, we construct a matrix similar to  $\lambda^{(p,q)}$  from Eq. 13, renamed  $\lambda_c^{(p,q)}$ , that does not contain complex phases but the non-negative probabilities of a ball bearing changing paths from the  $p^{th}$  path to the  $q^{th}$  path and vice versa. The  $Z$  matrices  $\lambda_c^{(p,q)}$  multiplied together in the

order specified by Eq. 15 constructs  $\mathbb{C}$ . Note that this matrix does not act on an equivalent space of modes as there is no superposition in this classical analog. The matrix serves as a tool to find the ultimate probability of a ball bearing starting on the  $k^{th}$  path and being collected in the  $j^{th}$  bin, given as the  $(j, k)$  element of the  $\mathbb{C}$  matrix.

Using the pinball machine analog and the constructed matrix  $\mathbb{C}$ , we can find the probability of seeing a particular output state given an input state. For the case where the number of balls equal the number of bins and one ball is sent out from each launcher, the probability of finding an output state is given by,

$$P_{\beta, \alpha} = \sum_{\sigma \in S_{\beta, \alpha}} \prod_{j=1}^N \mathbb{C}_{j, \sigma(j)}, \quad (26)$$

where the set  $S_{\beta, \alpha}$  is all permutations of the indices  $(1, 2, \dots, N)$  that assigns the ball bearings from the  $j^{th}$  launcher to the desired collection bin. Depending on the input and output states, the number terms in the sum of Eq. 26 can range from 1 to  $N!$ .

Of particular interest is the case of the probability of observing one ball landing in each bin. The set  $S_{per}$  associated with this setup includes all  $N!$  permutations of the set of numbers  $(1, 2, \dots, N)$  as there are  $N!$  different ways to arrange  $N$  distinguishable balls into  $N$  bins. The probability  $P_c$  associated with this special case is,

$$P_c = \sum_{\sigma \in S_{per}} \prod_{j=1}^N \mathbb{C}_{j, \sigma(j)} = Per(\mathbb{C}), \quad (27)$$

where  $Per(\mathbb{C})$  is the permanent of the matrix  $\mathbb{C}$  and is defined by Eq. 27. The permanent is defined in the same way as the more familiar determinant except there is no alternating sign change between the terms of products. However, unlike the determinant, the permanent is much more difficult to calculate or approximate.

There is a similar equation for the probability of finding  $n$  distinguishable photons in  $N$  modes where  $n = N$ , and where the input state and the output state are both  $|1, 1, \dots, 1\rangle$ . We define a new matrix  $\mathbb{D}$  which has components equal to the absolute square of the components of  $\Lambda$  as  $\mathbb{D}_{j, k} = |\Lambda_{j, k}|^2$ , where the probability of a distinguishable photon from the  $j^{th}$  mode reaching the  $k^{th}$  detector is given by the absolute square of the amplitude associated with this transition. Note that the matrix  $\mathbb{D}$  may not be unitary and does not act on the space of modes. Like  $\mathbb{C}$ , it is a tool to calculate probabilities, and from a mathematical perspective, is identical to  $\mathbb{C}$ .

The probability  $P_d$  related to distinguishable photons in this special regime is given by,

$$P_d = |\langle 1, 1, \dots, 1 | \hat{S} | 1, 1, \dots, 1 \rangle|^2 = Per(\mathbb{D}). \quad (28)$$

## 2.6.2 Indistinguishable Bosons and the Permanent

For indistinguishable bosons, the situation is more complicated. In addition to looking at the case where  $n = N$  and  $|\Psi^{(\alpha)}\rangle = |\Psi^{(\beta)}\rangle = |1, 1, \dots, 1\rangle$ , we will elaborate into a more general regime where  $n$  and  $N$  can vary, and where the input and output states of interest are not restricted.

First we construct a new set of  $N \times N$  matrices,  $\mathbb{N}^{(\alpha, \beta)}$ , with elements  $n_{j,k}^{(\alpha, \beta)}$ . These matrices are constructed so that the following is satisfied,

$$\begin{cases} \sum_{k=1}^N n_{j,k}^{(\alpha, \beta)} = n_j^{(\alpha)} \\ \sum_{j=1}^N n_{j,k}^{(\alpha, \beta)} = n_k^{(\beta)}. \end{cases} \quad (29)$$

For a given  $|\Psi^{(\alpha)}\rangle$  and  $|\Psi^{(\beta)}\rangle$ , the number of  $\mathbb{N}^{(\alpha, \beta)}$  matrices in the associated set will differ. For instance, for  $n = N$ ,  $|\Psi^{(\alpha)}\rangle = |1, 1, \dots, 1\rangle$  and  $|\Psi^{(\beta)}\rangle = |N, 0, 0, \dots, 0\rangle$ , there is only one matrix in the set corresponding with these input and output states, one where the  $N$  elements of the first column of the matrix are all equal to 1 and the rest of the elements of the matrix are 0. In contrast, for  $n = N$  and  $|\Psi^{(\alpha)}\rangle = |\Psi^{(\beta)}\rangle = |1, 1, \dots, 1\rangle$ , there are  $N!$  matrices in the set, one for each different way to sort the rows of an  $N \times N$  identity matrix. The number of matrices  $\mathbb{N}^{(\alpha, \beta)}$  in a set reflect the number of different paths photons in the input state  $|\Psi^{(\alpha)}\rangle$  can take that results in the desired output state  $|\Psi^{(\beta)}\rangle$ .

We now look again at Eq. 6. Concentrating on a portion of Eq. 6 and applying the multinomial theorem, for a given fixed  $j$ ,

$$\left( \sum_{k=1}^N (\Lambda_{k,j}) \hat{a}_j^\dagger \right)^{n_j^{(\alpha)}} = n_j^{(\alpha)}! \sum_{\mathbb{N}^{(\alpha, \beta)} \forall \beta} \prod_{k=1}^N \frac{1}{n_{j,k}^{(\alpha, \beta)}!} (\Lambda_{k,j}) \hat{a}_k^\dagger)^{n_{j,k}^{(\alpha, \beta)}}, \quad (30)$$

where the sum is taken over each member of the sets containing all  $\mathbb{N}^{(\alpha, \beta)}$  for all  $\beta$ . This equation changes the product of sums into sums of products. Substituting Eq. 30 into Eq. 6, we have

$$\hat{\mathbb{S}} |\Psi^{(\alpha)}\rangle = \left( \prod_{j=1}^N \sqrt{n_j^{(\alpha)}!} \sum_{\mathbb{N}^{(\alpha, \beta)} \forall \beta} \prod_{k=1}^N \frac{1}{n_{j,k}^{(\alpha, \beta)}!} (\Lambda_{k,j}) \hat{a}_k^\dagger)^{n_{j,k}^{(\alpha, \beta)}} \right) |0\rangle. \quad (31)$$

We can rewrite Eq. 31 by defining the coefficient  $K^{(\alpha, \beta)}$  related to each  $\mathbb{N}^{(\alpha, \beta)}$  as

$$K^{(\alpha, \beta)} = \frac{\prod_{\ell=1}^N \sqrt{n_\ell^{(\alpha)}!}}{\prod_{\ell=1}^N \prod_{\ell'=1}^N n_{\ell, \ell'}^{(\alpha, \beta)}!}. \quad (32)$$

Rearranging Eq. 31 and substituting in  $K^{(\alpha,\beta)}$  for compactness, we have

$$\hat{\mathbb{S}}|\Psi^{(\alpha)}\rangle = \sum_{\mathbb{N}^{(\alpha,\beta)} \forall \beta} K^{(\alpha,\beta)} \prod_{j=1}^N \prod_{k=1}^N (\Lambda_{k,j} \hat{a}_k^\dagger)^{n_{j,k}^{(\alpha,\beta)}} |0\rangle. \quad (33)$$

Now we look at Eq. 33 and any given term from the sum and see how all of the creation operators act on the vacuum state. Extracting and rearranging,

$$K^{(\alpha,\beta)} \prod_{j=1}^N \prod_{k=1}^N (\Lambda_{k,j} \hat{a}_k^\dagger)^{n_{j,k}^{(\alpha,\beta)}} |0\rangle = K^{(\alpha,\beta)} \left( \prod_{j=1}^N \prod_{k=1}^N (\Lambda_{k,j})^{n_{j,k}^{(\alpha,\beta)}} \right) \left( \prod_{j=1}^N \prod_{k=1}^N (\hat{a}_k^\dagger)^{n_{j,k}^{(\alpha,\beta)}} |0\rangle \right). \quad (34)$$

Concentrating on the double product with the creation operators  $\hat{a}_k^\dagger$  that act on  $|0\rangle$ , we can see that,

$$\prod_{j=1}^N \prod_{k=1}^N (\hat{a}_k^\dagger)^{n_{j,k}^{(\alpha,\beta)}} |0\rangle = \prod_{k=1}^N \left( (\hat{a}_k^\dagger)^{n_{1,k}^{(\alpha,\beta)}} (\hat{a}_k^\dagger)^{n_{2,k}^{(\alpha,\beta)}} \dots (\hat{a}_k^\dagger)^{n_{N,k}^{(\alpha,\beta)}} \right) |0\rangle = \prod_{k=1}^N (\hat{a}_k^\dagger)^{n_k^{(\beta)}} |0\rangle. \quad (35)$$

Here we can see how the  $N$  creation operators  $\hat{a}_k^\dagger$  act to place  $n_k^{(\beta)}$  photons in the  $k^{\text{th}}$  mode of the output state  $|\Psi^{(\beta)}\rangle$ . After applying all the  $\hat{a}_k^\dagger$  operators to the vacuum state, normalizing, and substituting the result into the term shown in Eq. 34, we have,

$$K^{(\alpha,\beta)} \prod_{j=1}^N \prod_{k=1}^N (\Lambda_{k,j} \hat{a}_k^\dagger)^{n_{j,k}^{(\alpha,\beta)}} |0\rangle = K^{(\alpha,\beta)} \left( \prod_{j=1}^N \prod_{k=1}^N (\Lambda_{k,j})^{n_{j,k}^{(\alpha,\beta)}} \right) \left( \prod_{\ell=1}^N \sqrt{n_\ell^{(\beta)}} \right) |\Psi^{(\beta)}\rangle. \quad (36)$$

Using this result, substituting it into Eq. 33, using the definition of  $K^{(\alpha,\beta)}$ , and rearranging,

$$\hat{\mathbb{S}}|\Psi^{(\alpha)}\rangle = \sum_{\mathbb{N}^{(\alpha,\beta)} \forall \beta} \prod_{j=1}^N \prod_{k=1}^N \sqrt{n_j^{(\alpha)}! n_j^{(\beta)}!} \frac{(\Lambda_{k,j})^{n_{j,k}^{(\alpha,\beta)}}}{n_{j,k}^{(\alpha,\beta)}!} |\Psi^{(\beta)}\rangle. \quad (37)$$

We can now look at the amplitude and hence the probability related to a particular  $|\Psi^{(\beta)}\rangle$ ,

$$\langle \Psi^{(\beta)} | \hat{\mathbb{S}} | \Psi^{(\alpha)} \rangle = \sum_{\mathbb{N}^{(\alpha,\beta)}} \prod_{j=1}^N \prod_{k=1}^N \frac{\sqrt{n_j^{(\alpha)}! n_k^{(\beta)}!}}{n_{j,k}^{(\alpha,\beta)}!} (\Lambda_{k,j})^{n_{j,k}^{(\alpha,\beta)}}, \quad (38)$$

where we can restrict the sum over only the matrices  $\mathbb{N}^{(\alpha,\beta)}$  with the same  $\beta$  as the output state  $\langle \Psi^{(\beta)} |$ .

If  $n = N$  and  $|\Psi^{(\alpha)}\rangle = |\Psi^{(\beta)}\rangle = |1, 1, \dots, 1\rangle$ , we can look at a particular case of Eq. 38. As discussed before, the number of  $\mathbb{N}^{(\alpha,\beta)}$  matrices and hence number of terms in the sum is equal to  $N!$ ,

$$\langle 1, 1, \dots, 1 | \hat{\mathbb{S}} | 1, 1, \dots, 1 \rangle = \sum_{\mathbb{N}(\alpha, \beta)} \prod_{j=1}^N \prod_{k=1}^N \Lambda_{k,j}^{n_{j,k}^{(\alpha, \beta)}} \quad (39)$$

In Eq. 39, there are only  $N$  values of the exponent  $n_{j,k}^{(\alpha, \beta)}$  that are equal to 1, and the other  $N^2 - N$  values are 0. As all  $\mathbb{N}(\alpha, \beta)$  represent the  $N!$  ways to permute the different rows of an  $N \times N$  identity matrix, this allows us to view Eq. 39 as a sum of products of all elements of  $\Lambda$  under all permutations of indices. This makes Eq. 39 equivalent to,

$$\langle 1, 1, \dots, 1 | \hat{\mathbb{S}} | 1, 1, \dots, 1 \rangle = \sum_{\mathbb{N}(\alpha, \beta)} \prod_{j=1}^N \prod_{k=1}^N \Lambda_{k,j}^{n_{j,k}^{(\alpha, \beta)}} = \sum_{\sigma \in S_{per}} \prod_{j=1}^N \Lambda_{j, \sigma(j)}, \quad (40)$$

and finally, using the definition of the permanent,

$$|\langle 1, 1, \dots, 1 | \hat{\mathbb{S}} | 1, 1, \dots, 1 \rangle|^2 = |Per(\Lambda)|^2 \quad (41)$$

As with the classical analog and the case of distinguishable photons, the permanent can be used to describe the probabilities related to the input state and the output state equal to  $|1, 1, \dots, 1\rangle$  for situations involving indistinguishable photons. However, unlike in the case of distinguishable particles, due to the complex elements of  $\Lambda$ , approximating the permanent of  $\Lambda$  is intractable as  $N$  increases as discussed in Sec. 3.2.

It is important to note that if the indistinguishable particles in question were fermions, there would be a sign change upon the exchange of two particles in the output modes, which would result not in the use of the permanent but instead the determinant. However, as the absolute value of the determinant of any unitary matrix is always 1, this demonstrates that a “fermion sampler” would only detect output states of  $|1, 1, \dots, 1\rangle$  and never anything else, as to be expected given the exclusion principle and the antisymmetric nature of fermion wavefunctions. Such a device would be quite boring and trivial to simulate.

We can look at situations when neither  $|\Psi^{(\alpha)}\rangle$  nor  $|\Psi^{(\beta)}\rangle$  are equal to  $|1, 1, \dots, 1\rangle$ . Returning to Eq. 38, we can simplify this complex amplitude using the permanent of a matrix with elements taken from  $\Lambda$ . We first define this new  $n \times n$  matrix  $\Lambda^{(\alpha, \beta)}$  that has elements from  $\Lambda$  as described by Scheel[11]. This matrix is constructed so that when its permanent is taken, the amplitudes for the  $n$  photons beginning in  $|\Psi^{(\alpha)}\rangle$  are permuted, multiplied and summed in all the ways to produce  $|\Psi^{(\beta)}\rangle$  with the proper interference effects. A MATLAB function `getScheelMatrix.m` that creates this matrix is shown in the Appendix C.1.

The construction of  $\Lambda^{(\alpha, \beta)}$  is best illustrated with an example. Let  $|\Psi^{(\alpha)}\rangle = |3, 0, 1\rangle$  and  $|\Psi^{(\beta)}\rangle = |1, 2, 1\rangle$ . In the input state, we will have three amplitudes for three photons  $\Lambda_{1,k}$  and one amplitude  $\Lambda_{3,k}$  for the last photon in the last mode. Therefore, within  $\Lambda^{(\alpha, \beta)}$ , we will have three elements from  $\Lambda$  with the row index 1 and one element with row index 3. In the output state, we will have one amplitude  $\Lambda_{j,1}$ , two with  $\Lambda_{j,2}$  and one with  $\Lambda_{j,3}$ . The elements in  $\Lambda^{(\alpha, \beta)}$  have elements from  $\Lambda$  with repeated row indices as defined by the number

of photons in each mode of input state and repeated column indices as indicated by the output state. Putting this all together, we have,

$$\left. \begin{array}{l} |\Psi^{(\alpha)}\rangle = |3, 0, 1\rangle \\ |\Psi^{(\beta)}\rangle = |1, 2, 1\rangle \end{array} \right\} \Rightarrow \Lambda^{(\alpha,\beta)} = \begin{pmatrix} \Lambda_{1,1} & \Lambda_{1,2} & \Lambda_{1,2} & \Lambda_{1,3} \\ \Lambda_{1,1} & \Lambda_{1,2} & \Lambda_{1,2} & \Lambda_{1,3} \\ \Lambda_{1,1} & \Lambda_{1,2} & \Lambda_{1,2} & \Lambda_{1,3} \\ \Lambda_{3,1} & \Lambda_{3,2} & \Lambda_{3,2} & \Lambda_{3,3} \end{pmatrix} \quad (42)$$

If we take the permanent of  $\Lambda^{(\alpha,\beta)}$ , due to its construction of repeated elements taken from  $\Lambda$ , we will have more products than indicated by the double product in Eq. 38, which has only  $n$  products not equal to 1. In particular,

$$Per(\Lambda^{(\alpha,\beta)}) = \sum_{\mathbb{N}^{(\alpha,\beta)}} \prod_{j=1}^N \prod_{k=1}^N \frac{n_{j,k}^{(\alpha,\beta)}!}{n_j^{(\alpha)}! n_k^{(\beta)}!} (\Lambda_{k,j})^{n_{j,k}^{(\alpha,\beta)}} \quad (43)$$

Substituting the relationship of Eq. 43 into Eq. 38, we have the elegant expression given by Scheel [11],

$$\left| \langle \Psi^{(\beta)} | \hat{\mathbb{S}} | \Psi^{(\alpha)} \rangle \right|^2 = \frac{|Per(\Lambda^{(\alpha,\beta)})|^2}{\prod_{j=1}^N n_j^{(\alpha)}! n_j^{(\beta)}!} \quad (44)$$

Again, the permanent can be used to present a compact expression for the probabilities related to different output states.

## 3 Predicting Output Possibilities

### 3.1 Simulations

A boson sampler will very quickly detect an output state when photons prepared in an input Fock state are put through the interferometer. A possible detected output state  $|\Psi^{(\beta)}\rangle$  will be detected with probability  $|\langle \Psi^{(\beta)} | \hat{\mathbb{S}} | \Psi^{(\alpha)} \rangle|^2$ , and an ideal boson sampler naturally “chooses” a  $|\Psi^{(\beta)}\rangle$  out of the  $A$  possibilities each time it is run. The challenge is instructing a classical computer to do the same within the same probability distribution.

Consider first the case where the bosons are distinguishable. While in this case the photons do not interfere with each other, a single photon still interferes *with itself* as in the single-photon, double-slit experiment. One method to determine an output state from the probability of a boson sampler scattering distinguishable particles is to calculate the chance of seeing each  $A$  possible output states and sampling a random number to choose  $|\Psi^{(\beta)}\rangle$  from the  $A$  choices. However,  $A$  grows dramatically with  $N$  and  $n$ , and by taking advantage of computerized random sampling, generating an output state from distinguishable photons does not require calculating the probability of each possible output state.

We can, however, use a computer simulation for any given input state that randomly chooses a final detected mode for each single photon one at a time if the photons are *distinguishable*. In this case, the probability of a photon created in the  $j^{\text{th}}$  mode being detected by the  $k^{\text{th}}$  detector is merely the absolute square of the complex amplitude  $\Lambda_{j,k}$ . The actual path a photon could have been regarded to take is unknown as any premature observation would decohere the system. All that can be known is the probability of a single photon arriving at a single detector.

Using this method, for each of the  $n$  photons, we can sample normally-distributed pseudorandom numbers generated by a computer that determine where an individual photon ultimately lands. Thus by sampling  $n$  random numbers, we can pick out a randomly generated output state that will be taken from the distribution generated by a boson sampler with distinguishable particles.

To determine a photon's ending point, each random number  $\rho$  is taken between 0 and 1. If we are looking at a photon that began in the  $j^{\text{th}}$  mode, and  $0 \leq \rho \leq |\Lambda_{j,1}|^2$ , then we put the photon in the first bin. If  $|\Lambda_{j,1}|^2 < \rho \leq |\Lambda_{j,2}|^2 + |\Lambda_{j,1}|^2$ , then the photon lands in the second bin, and so on, carving up the region between 0 and 1 into  $N$  pieces each with length  $|\Lambda_{j,k}|^2$  for a given  $j$  to randomly determine the fate of a distinguishable photon that begins in the  $j^{\text{th}}$  mode. This procedure is repeated with newly generated  $\rho$  for each of the  $n$  photons. After  $n$  random numbers are sampled, the photons are counted in their bins and an output state is generated. This takes much less resources than considering all  $A$  possible output states. Essentially the same procedure can be applied to the classical analog where the transition probabilities are encoded in the matrix  $\mathbb{C}$ . The only use of quantum mechanics in the case of distinguishable bosons is that the probabilities are defined by the absolute square of the amplitudes  $\Lambda_{j,k}$ .

In stark contrast, if the photons are *indistinguishable*, the probability of a single photon arriving at a particular detector no longer has meaning. Indistinguishable photons not only interfere with themselves but also with each other, in the sense that multi-photon processes interfere with other process. The probabilities are coupled and there is no way to simulate photons traveling through the device one at a time.

To further illustrate the differences between simulating distinguishable and indistinguishable photons, consider the example discussed in Sec. 2.5, where  $n = N = 2$  and  $T = R = \frac{1}{2}$ . If we were to sample two random numbers and use these to determine whether the top and bottom photons reflect or transmit, we would obtain the correct probability distribution of the output states for distinguishable photons. However, sampling two random numbers to determine individual photon paths does not yield the correct probabilities for the case of two indistinguishable photons as the  $|1, 1\rangle$  output state is impossible to observe.

For indistinguishable photons, to randomly determine an output state given an input state and a  $\Lambda$  matrix, the complex amplitude of each member of the Fock space must be calculated. Using these amplitudes, the region between 0 and 1 can be carved up into  $A$  pieces, and one and only one random number is sampled to determine an output taken from

the correct probability distribution. The problem becomes one of brute force calculation and can no longer rely on the advantages of computerized random sampling. A correct method using random sampling or another technique would be preferred, but appears to be impossible.

### 3.2 Using the Permanent

Calculating the possibilities associated with possible output states uses the permanent for special cases of distinguishable photons and the classical analog as well as for all cases of indistinguishable photons. If calculating the  $A$  possibilities within the Fock space is desired, a task that ultimately scales undesirably with  $n$  and  $N$ , calculating the permanent for each  $A$  is used, a task by itself that scales just as poorly.

As can be seen from the definition in Eq. 27, there are  $N!$  terms in the permanent, or in the case of the  $\Lambda^{(\alpha,\beta)}$  matrices,  $n!$  terms. In any case, calculating this probability does not scale well for either  $N$  or  $n$ . It has been proven by Valiant [13] that the problem of calculating the permanent cannot be calculated in *polynomial time*, where resources scale as some polynomial function of the size of the matrix. There are options, however, that reduce the number of calculations on the order of  $N!$  as demanded in Eq. 27.

The elements of the matrices  $\mathbb{C}$  and  $\mathbb{D}$  are all non-negative and real, and there exists an algorithm using random sampling to accurately estimate the permanent of such a matrix in polynomial time as developed by Jerrum et al [5]. In this particular case, a classical computer can efficiently approximate the probabilities  $P_c$  and  $P_d$ . The elements of  $\Lambda$  are complex, and therefore can not make use of this approximation. This difference is the key to the importance of the boson sampler. For a matrix with complex entries, the permanent is provably *inefficient* to calculate or approximate, lying in a complexity class  $\#P$ -complete[13]. Due to the effects of identical interfering photons, the probability of any outcome cannot be efficiently calculated as the number of photons and modes increases, as opposed to the case of distinguishable photons. The boson sampler is therefore potentially a machine that has access to complexity classes unreachable to a computer that operates in the classical regime.

The Ryser formula [10] provides a faster way to calculate the permanent of a matrix that can't use Jerrum's method as in the case of indistinguishable photons. However, this only reduces the number of calculations to the order of  $2^n n^2$  for the permanents of the  $\Lambda^{(\alpha,\beta)}$  matrices. While this is an improvement over  $n!$ , the scaling is still not ideal for a classical computer. An elegant and simple MATLAB function **ryserPermanent.m** written by Winslow [14] provides a way to calculate the permanent of a matrix without hefty memory demands.

Using the Ryser formula to compute the permanents of all  $A$  matrices  $\Lambda^{(\alpha,\beta)}$  to calculate the probability distribution across the entire Fock space, a least  $G$  calculations are needed where,

$$G = A(2^n n^2) = 2^n n^2 \frac{(N + n - 1)!}{n!(N - 1)!} \quad (45)$$

The scaling nature of  $G$  shows that using a classical computer to randomly determine an output state by calculating all possibilities is intractable for large  $n$  and  $N$ .

## 4 Results

### 4.1 Computation Times and Probability Distributions

Fig. 6 shows the average computation times taken by a typical desktop computer to calculate the entire probability distribution over the Fock space. As can be seen, the computation time rises quite dramatically as  $n = N$  increases. The method proposed is too cumbersome to scale to high numbers of modes and photons. Indeed, the problem cannot be made much easier for a classical computer to handle.

Fig. 7 shows the probability distributions for  $n = N = 5$ . Even with such small numbers of modes and photons, the expanse of the Fock space is impressive.

The Appendix C.2 details the MATLAB function `getBiOutputAmps.m` that calculates the probability distribution of a virtual boson sampler given a  $\Lambda$  and  $|\Psi^{(\alpha)}\rangle$ , and the function `getBosonSample.m` generates random output from this distribution. These functions can serve as a method to simulate a boson sampler on a computer, albeit with severe demands of computation time.

### 4.2 Discussion

While an ideal boson sampler that scatters indistinguishable photons is not a universal quantum computer, it does offer a way to sample from a probability distribution that is very difficult for a classical computer to compute and impossible to simulate using random sampling for each photon. Unlike classical regimes, or even that of distinguishable photons, the state must be considered as a whole, and the myriad of ways the photon amplitudes interfere with each other provide a huge obstacle towards simulating the boson sampler on a classical computer.

The solution proposed, of using brute force calculation to generate the probability distribution over the entire Fock space and randomly sample from it, is extremely limited. As the number of modes and photons increase, the computational difficulty of directly calculating the probability distribution soars exponentially. While calculating the permanent that provides the probability of a single possible output does not scale well with the number of photons, the Fock space itself scales even more poorly and provides the greatest challenge to simulating a boson sampler using the direct calculation method.

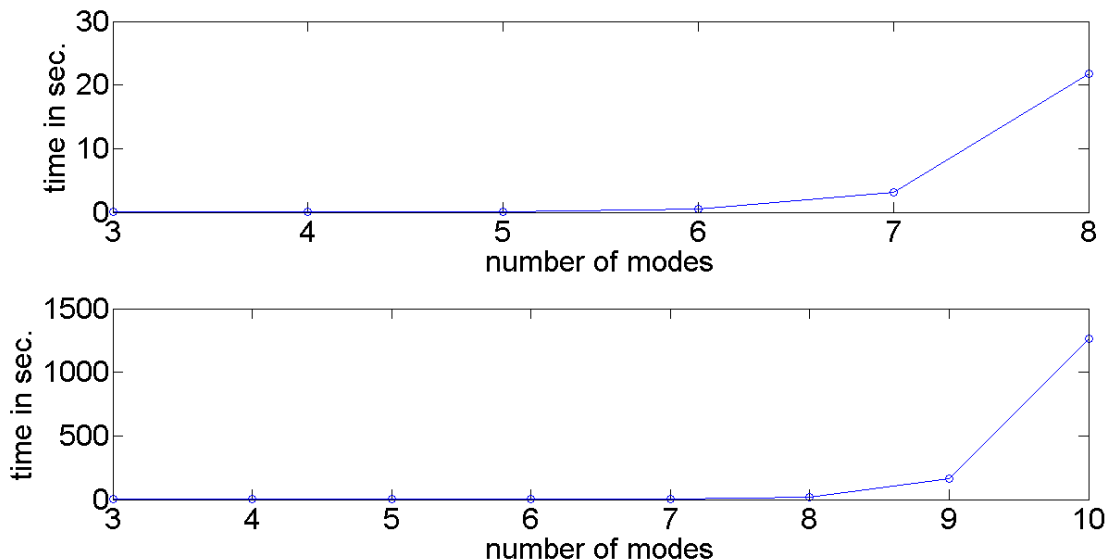


Figure 6: Plots of number of modes versus average computation time taken to compute the amplitudes of all members of a Fock space where  $n = N$  and the input state was given to be  $|1, 1, \dots, 1\rangle$ . Specifically, the times taken for the MATLAB function `getBiOutputAmps.m` to run were recorded. This function can be seen in Appendix C.2. For each mode, nine trials were averaged and a random unitary  $N \times N$   $\Lambda$  matrix (see Appendix C.4) was used for each trial. The top plot shows modes 3 to 8, and the second shows modes 3 to 10 in order to emphasize the increase of computation time as  $N$  increases. The calculation times were gathered on a computer using 32-bit Windows XP SP3 with Intel Core 2 CPU E6420 @ 2.13 GHz running MATLAB R2014a. While the computer has two processors, only one was used to compute the amplitudes.

Because of the difficulties faced by a classical computer, the ideal boson sampler, within its limited regime, offers an example of a device that exploits quantum mechanics to generate output that is intractable for a classical computer to emulate.

## 5 Outlook

The issue of the increasing size of the Fock space and the complexity of the permanent ultimately limit the method of calculating the probability distribution. If a method of using random samples to simulate a boson sampler for indistinguishable particles can be developed to surmount the obstacle of the Fock space and the permanent, the simulation may be improved, but ultimately the nature of the problem as it stands is out of reach of a classical computer. Unless certain foundations of classical computational complexity theory collapse, implementation of an ideal boson sampler would be a demonstration of a quantum information processor that can outperform a classical Turing machine.

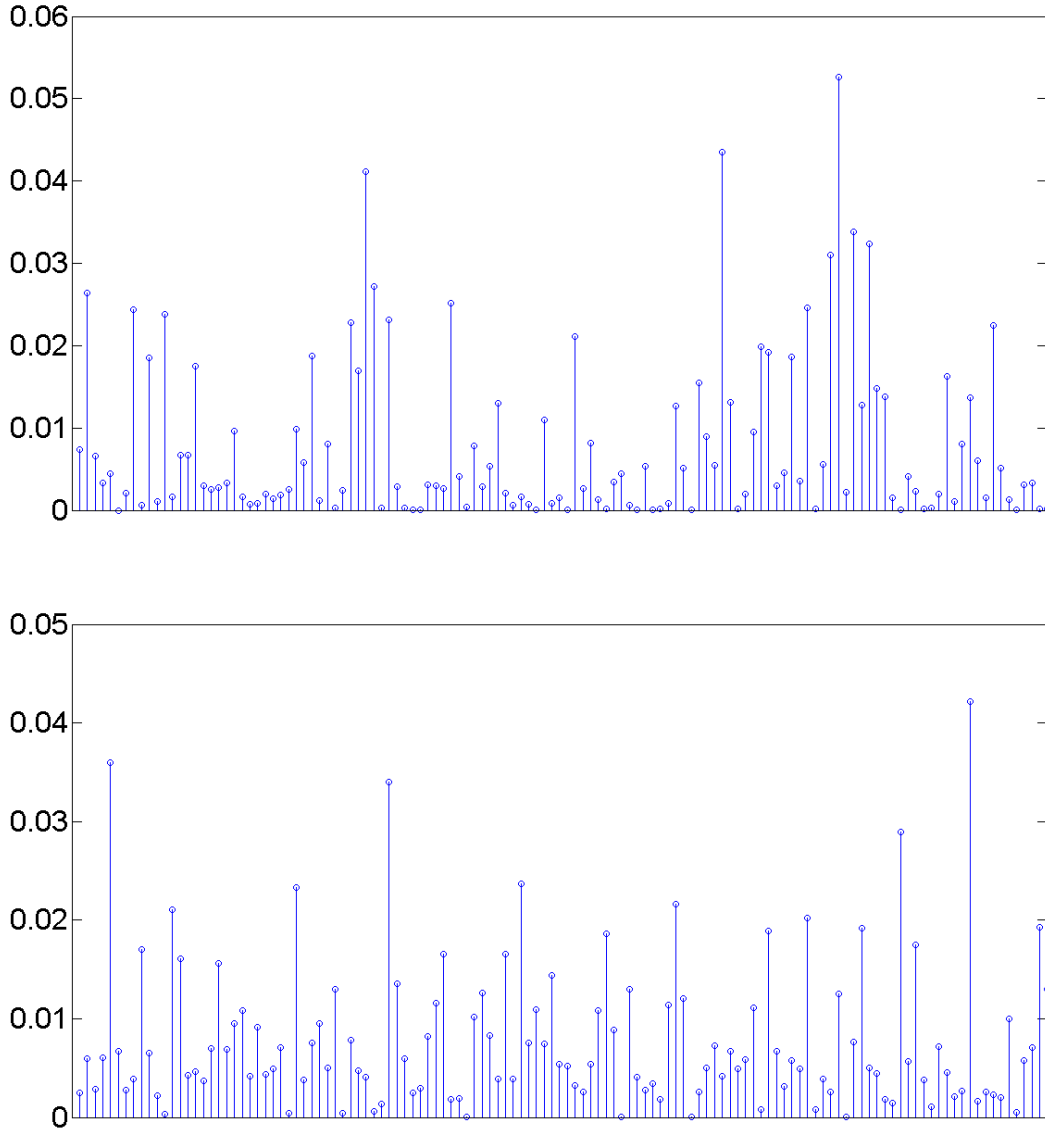


Figure 7: Probability distributions for  $n = N = 5$  and  $|\Psi^{(\alpha)}\rangle = |1, 1, 1, 1, 1\rangle$ . The MATLAB function `getBiOutputAmps.m` was used to generate these probability distributions and can be found in Appendix C.2. The top plot shows the results generated with all 50-50 beam splitters and no beam splitters. The bottom plot was generated with a random unitary matrix for  $\Lambda$  (see Appendix C.4). The state labels along the horizontal axis has been omitted for clarity. The probabilities are ordered by state left to right using ascending binary order as described in Appendix B.

However, the ideal boson sampler assumes many perfections, but in reality it will suffer from a multitude of errors. Each photon has to be identical and within a pure Fock state, the properties of the beam splitters and phase shifters exact, the detectors have to count the photons without failure, and so on. The issue of timing errors for more than two modes and two photons, in which the photons do not leave their sources at the same time, are currently being explored both analytically and within a computer simulation. It is hoped that if the timing errors are not extreme, then the probability distribution of such a flawed boson sampler will approach that of the ideal. Studying timing errors in a multi-mode, multi-photon boson sampler may give insight on how well the device must be controlled in order to ensure that samples from the interferometer are taken from these probability distributions that are inefficient for a classical computer to generate. While the boson sampler has potential as a quantum device, it is not a digital quantum computer and has no form of error correction. It is hoped that this work will help further the exploration of the computational power of analog quantum machines and beyond.

## Acknowledgments

I would like to express my immense gratitude to Dr. Ivan H. Deutsch and Tyler “Bob” Keating for their contributions, support, patience and friendship. Thank you for the opportunity to work with you, the knowledge you have shared with me about such intriguing mysteries, and the confidence you have instilled in me as I go forward. I would also like to thank my poor outdated classical computer for burning through countless calculations over and over and over again without complaining.

## References

- [1] Aaronson, Scott and Arkhipov, Alex. “The Computational Complexity of Linear Optics.” arXiv:1011.3245 [quant-ph]
- [2] Feller, William. *An Introduction to Probability Theory*. New York: John Wiley & Sons Inc., 3rd Edition, 1957.
- [3] van der Geest, Jos. 2007. *PERMPOS* MATLAB Central File Exchange. Link to webpage retrieved April 26, 2014
- [4] Hong, Ou and Mandell. “Measurement of Subpicosecond Time Intervals Between Two photons by Interference.” *Physical Review Letters*, Vol. 59 No. 18, 1987. pp. 2044-2046
- [5] Jerrum, Sinclair, and Vigoda. “A Polynomial-Time Approximation Algorithm for the Permanent of a Matrix with Nonnegative Entries.” *Journal of the ACM*, Vol. 51, No. 4, July 2004, pp. 671-697 Link to PDF retrieved April 26, 2014

- [6] Kok, Pieter. Image taken from the Wikipedia article Hong-Ou-Mandel effect retrieved April 26, 2014. Image was unmodified and used under the Creative Commons license.
- [7] Medrazzi, Francesco. “How to Generate Random Matrices from the Classical Compact Groups.” arXiv:math-ph/0609050v2
- [8] Nielsen, Michael A. and Chuang, Issac L. *Quantum Computation and Quantum information* Cambridge University Press, 2000
- [9] Reck, Zeilinger, Bernstein and Bertani. “Experimental Realization of Any Discrete Unitary Operator.” *Physical Review Letters*, Vol. 73, No. 1, July 1994, pp 58-62
- [10] Ryser, Herbert John. *Combinatorial Mathematics*. The Carus Mathematical Monographs, The Mathematical Association of America. 1963.
- [11] Scheel, Stefan. “Permanents in Linear Optical Networks.” arXiv:quant-ph/0406127
- [12] Tichy, Malte C. “Interference of Identical Particles from Entanglement to Boson-Sampling.” arXiv:1312.4266v1 [quant-ph]
- [13] Valiant, L. G. “The Complexity of Computing the Permanent.” *Theoretical Computer Science* Vol. 8, 1979, pp. 189-201. Link to PDF retrieved April 26, 2014
- [14] Winslow, Luke. 2012. *Matrix Permanent using Ryser Algorithm*. MATLAB Central File Exchange. Link to webpage retrieved April 26, 2014

## A Finding Parameters for an Arbitrary Unitary Matrix

The matrix  $\Lambda$  is the product of unitary matrices, and thus is unitary itself. By systematically adjusting the values for all  $\theta^{(p,q)}$ ,  $\varphi^{(p,q)}$ , and  $\psi_k$ , any arbitrary  $N \times N$  unitary matrix may be constructed. There are  $2Z + N = N^2$  independent parameters, enough to describe any arbitrary  $N \times N$  unitary matrix in this particular basis.

Given an arbitrary unitary transformation  $\hat{S}$  that acts on the Hilbert space of  $|\Psi^{(\alpha)}\rangle$  such that  $\hat{S}|\Psi^{(\alpha)}\rangle = |\Psi_{out}^{(\alpha)}\rangle$ , the matrix  $\Lambda$  can be found that satisfies Eq. 6. Once  $\Lambda$  is defined, the  $N^2$  number of parameters of all  $\theta^{(p,q)}$ ,  $\varphi^{(p,q)}$ , and  $\psi_k$  can be found that ultimately describe  $\Lambda$ . Ideally, a system of finely-tuned and carefully placed photon sources, detectors, beam splitters and phase shifters can serve as an analog machine whose output is taken from the unique probability distribution of the ideal boson sampler.

To find all  $N^2$  rotation angles that define  $\Lambda$ , the following equation taken from the relationships of Sec. 2.4 can be useful,

$$\begin{aligned}\mathbb{Y}^\dagger &= [\lambda^{(N-1,N)}][\lambda^{(N-2,N)}\lambda^{(N-2,N-1)}]\dots[\lambda^{(1,N)}\lambda^{(1,N-1)}\dots\lambda^{(1,3)}\lambda^{(1,2)}]\Lambda^\dagger \\ &= (\mathbb{L}^{(N-1)}\mathbb{L}^{(N-2)}\dots\mathbb{L}^{(2)}\mathbb{L}^{(1)})\Lambda^\dagger.\end{aligned}\quad (46)$$

The result of each  $\lambda^{(p,q)}$  acting on  $\Lambda^\dagger$  in the prescribed order of Eq. (46) results in the diagonal unitary matrix  $\mathbb{Y}^\dagger$ . For  $p = 1$ , the  $N - 1$  number of  $\lambda^{(q,1)}$  matrices given by  $\mathbb{L}^{(1)}$  act on  $\Lambda^\dagger$  first, and only affect the first row (or column) of  $\Lambda^\dagger$ . None of the remaining  $\lambda^{(p\neq 1,q)}$  in  $\mathbb{L}^{(p\neq 1)}$  will act upon the other  $N - 1$  rows or columns (as  $q$  never equals one). Thus after  $\mathbb{L}^{(1)}$  has been applied in to  $\Lambda^\dagger$ , the first row and the first column of the resultant matrix is identical to the first row and first column of  $\mathbb{Y}^\dagger$ .

The unitary matrices  $\Lambda^\dagger$  and  $\mathbb{Y}^\dagger$  can be thought of as a row of orthonormal column vectors,

$$\Lambda^\dagger = (|v_1\rangle, |v_2\rangle, \dots |v_N\rangle), \quad (47)$$

and,

$$\mathbb{Y}^\dagger = (|d_1\rangle, |d_2\rangle, \dots |d_N\rangle). \quad (48)$$

Using this description, for  $p = 1$ ,

$$\mathbb{L}^{(1)}|v_1\rangle = |d_1\rangle. \quad (49)$$

The elements of the column vectors  $|d_k\rangle$  have value zero except for the  $k^{th}$  element, which is equal to  $e^{-i\psi_k}$ . The action of the unitary matrix  $\mathbb{L}^{(1)}$  can be seen as rotating the vector  $|v_1\rangle$  into  $|d_1\rangle$  which changes the first element to  $e^{-i\psi_1}$  and the rest to zero. A similar rotation occurs if the matrices  $\Lambda^\dagger$  and  $\mathbb{Y}^\dagger$  are viewed as stacked orthonormal row vectors, where the first row vector of  $\Lambda^\dagger$  is rotated into the first row vector of  $\mathbb{Y}^\dagger$  by the action of  $\mathbb{L}^{(1)}$  multiplying the row vector from the right.

Generalizing, for any  $p$ , the matrix  $\mathbb{L}^{(p)}$  can also be seen as rotations in the complex space of modes. The outcome of rotations by successive  $\mathbb{L}^{(p)}$  matrices acting on  $\Lambda^\dagger$  is represented by a new matrix  $\Lambda^{(p)}$ . This matrix is also defined when  $p = 0$  as well as for  $1 \leq p \leq N - 1$ . For  $p = 0$ , we define  $\Lambda^{(0)} = \Lambda^\dagger$ . For  $p > 0$ ,

$$\Lambda^{(p)} = (\mathbb{L}^{(p)}\mathbb{L}^{(p-1)}\dots\mathbb{L}^{(2)}\mathbb{L}^{(1)})\Lambda^{(0)} = \mathbb{L}^{(p)}\Lambda^{(p-1)}. \quad (50)$$

To help visualize the transformations, let  $\Lambda_{sub}^{(p)}$  be a  $(N - p) \times (N - p)$  unitary submatrix embedded within the block matrix  $\Lambda^{(p)}$ . For  $p = 0$ , we define  $\Lambda_{sub}^{(0)} = \Lambda^{(0)}$ . To describe  $\Lambda_{sub}^{(p)}$  for  $p > 0$ , we regard the placement of this submatrix within the structure of  $\Lambda^{(p)}$  as follows,

$$\Lambda^{(p)} = \left( \begin{array}{cccc|c} e^{-i\psi_1} & 0 & \dots & 0 & \emptyset \\ 0 & e^{-i\psi_2} & \dots & 0 & \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & \dots & e^{-i\psi_p} & \\ \hline & & & \emptyset^T & \Lambda_{sub}^{(p)} \end{array} \right), \quad (51)$$

where  $\emptyset$  is a  $p \times (N - p)$  zero matrix, and  $\emptyset^T$  is its transpose, a  $(N - p) \times p$  zero matrix. For  $p > 0$ , the matrix  $\mathbb{L}^{(p)}$  acting on  $\Lambda^{(p-1)}$  as in Eq. 50 can be regarded as essentially only transforming  $\Lambda_{sub}^{(p-1)}$  while preserving the rest of matrix  $\Lambda^{(p-1)}$ . Eq. 51 can be used to help visualize the results of the transformations as they are sequentially applied.

We can now consider a iterative, algorithmic method for finding the values of  $\theta^{(p,q)}$  and  $\varphi^{(p,q)}$  that transform  $\Lambda^\dagger$  into  $\mathbb{Y}^\dagger$  as in Eq. 46. To begin, we regard any matrix  $\Lambda^{(p)}$  as a row of column vectors as before in Eq. 47,

$$\Lambda^{(p)} = (|v_1^{(p)}\rangle, |v_2^{(p)}\rangle, \dots |v_N^{(p)}\rangle). \quad (52)$$

We require  $\mathbb{L}^{(p)} |v_p^{(p-1)}\rangle = |d_p\rangle$ . We start with  $p = 1$  and require as identically shown in Eq. 49,

$$\mathbb{L}^{(1)} |v_1^{(0)}\rangle = (\lambda^{(1,N)} \lambda^{(1,N-1)} \dots \lambda^{(1,3)} \lambda^{(1,2)}) |v_1^{(0)}\rangle = |v_1^{(1)}\rangle = |d_1\rangle. \quad (53)$$

Delving further into detail, we look at the first rotation when  $q = 2$ ,

$$\lambda^{(1,2)} |v_1^{(0)}\rangle = \lambda^{(1,2)} \begin{pmatrix} \Lambda_{1,1}^{(0)} \\ \Lambda_{2,1}^{(0)} \\ \vdots \\ \Lambda_{N,1}^{(0)} \end{pmatrix}. \quad (54)$$

As discussed before in Sec. 2.2, the matrix  $\lambda^{(1,2)}$ , only acts on the first and second elements of  $|v_1^{(0)}\rangle$ . We can look at the more interesting 2x2 subspace of Eq. 54:

$$\chi^{(1,2)} \begin{pmatrix} \Lambda_{1,1}^{(0)} \\ \Lambda_{2,1}^{(0)} \end{pmatrix} = \begin{pmatrix} e^{i\varphi^{(1,2)}} \cos(\theta^{(1,2)}) & \sin(\theta^{(1,2)}) \\ -e^{i\varphi^{(1,2)}} \sin(\theta^{(1,2)}) & \cos(\theta^{(1,2)}) \end{pmatrix} \begin{pmatrix} \Lambda_{1,1}^{(0)} \\ \Lambda_{2,1}^{(0)} \end{pmatrix} = \begin{pmatrix} z^{(1,2)} \\ \Lambda_{2,1}^{(1)} \end{pmatrix} \quad (55)$$

where  $z^{(1,2)}$  is the complex element (1,1) of the new matrix after  $\lambda^{(1,2)}$  has been applied. It has the form,

$$z^{(1,2)} = e^{i\varphi^{(1,2)}} \cos(\theta^{(1,2)}) \Lambda_{1,1}^{(0)} + \sin(\theta^{(1,2)}) \Lambda_{2,1}^{(0)}. \quad (56)$$

The matrix  $\lambda^{(1,2)}$  is the only matrix within  $\mathbb{L}^{(1)}$  that acts on the second element of the column vector  $|v_1^{(0)}\rangle$ , and so the transformation described in Eq. 55 defines  $\Lambda_{2,1}^{(1)}$ . Referring to Eq. 53, it is required that the second element of  $|v_1^{(1)}\rangle$  must equal the second element of

$|d_1\rangle$ , which is zero. Using these ideas as well as Eq. 55, we can make the important assertion that,

$$-e^{i\varphi^{(1,2)}} \sin(\theta^{(1,2)})\Lambda_{1,1}^{(0)} + \cos(\theta^{(1,2)})\Lambda_{2,1}^{(0)} = \Lambda_{2,1}^{(1)} = 0. \quad (57)$$

From this we can define  $\theta^{(1,2)}$  and  $\varphi^{(1,2)}$ ,

$$\begin{cases} \tan(\theta^{(1,2)}) = \left| \frac{\Lambda_{2,1}^{(0)}}{\Lambda_{1,1}^{(0)}} \right| \\ \varphi^{(1,2)} = \text{Arg}(\Lambda_{2,1}^{(0)}) - \text{Arg}(\Lambda_{1,1}^{(0)}) \end{cases}. \quad (58)$$

Once  $\theta^{(1,2)}$  and  $\varphi^{(1,2)}$  are determined,  $z^{(1,2)}$  can be calculated using Eq. 56.

Now consider the next matrix,  $\lambda^{(1,3)}$ , and the 2x2 subspace associated with it,

$$\chi^{(1,3)} \begin{pmatrix} \Lambda_{1,1}^{(0)} \\ \Lambda_{3,1}^{(0)} \end{pmatrix} = \begin{pmatrix} e^{i\varphi^{(1,3)}} \cos(\theta^{(1,3)}) & \sin(\theta^{(1,3)}) \\ -e^{i\varphi^{(1,3)}} \sin(\theta^{(1,3)}) & \cos(\theta^{(1,3)}) \end{pmatrix} \begin{pmatrix} z^{(1,2)} \\ \Lambda_{3,1}^{(0)} \end{pmatrix} = \begin{pmatrix} z^{(1,3)} \\ \Lambda_{3,1}^{(1)} \end{pmatrix} = \begin{pmatrix} z^{(1,3)} \\ 0 \end{pmatrix}. \quad (59)$$

The new values  $\theta^{(1,3)}$ ,  $\varphi^{(1,3)}$  and  $z^{(1,3)}$  are given by,

$$\begin{cases} \tan(\theta^{(1,3)}) = \left| \frac{\Lambda_{3,1}^{(0)}}{z^{(1,2)}} \right| \\ \varphi^{(1,3)} = \text{Arg}(\Lambda_{3,1}^{(0)}) - \text{Arg}(z^{(1,2)}) \\ z^{(1,3)} = e^{i\varphi^{(1,3)}} \cos(\theta^{(1,3)})z^{(1,2)} + \sin(\theta^{(1,3)})\Lambda_{3,1}^{(0)} \end{cases}. \quad (60)$$

Once  $\theta^{(1,3)}$ ,  $\varphi^{(1,3)}$ , and  $z^{(1,3)}$  are discovered in a similar fashion as for the previous unknowns, we move onto  $\lambda^{(1,4)}$ , and so on. In general, for  $p = 1$  and allowing  $z^{(1,1)} = \Lambda_{1,1}^{(0)}$ , the following recursive relationships can be developed,

$$\begin{pmatrix} e^{i\varphi^{(1,q)}} \cos(\theta^{(1,q)}) & \sin(\theta^{(1,q)}) \\ -e^{i\varphi^{(1,q)}} \sin(\theta^{(1,q)}) & \cos(\theta^{(1,q)}) \end{pmatrix} \begin{pmatrix} z^{(1,q-1)} \\ \Lambda_{q,1}^{(0)} \end{pmatrix} = \begin{pmatrix} z^{(1,q)} \\ 0 \end{pmatrix}, \quad (61)$$

and,

$$\begin{cases} \tan(\theta^{(1,q)}) = \left| \frac{\Lambda_{q,1}^{(0)}}{z^{(1,q-1)}} \right| \\ \varphi^{(1,q)} = \text{Arg}(\Lambda_{q,1}^{(0)}) - \text{Arg}(z^{(1,q-1)}) \\ z^{(1,q)} = e^{i\varphi^{(1,q)}} \cos(\theta^{(1,q)})z^{(1,q-1)} + \sin(\theta^{(1,q)})\Lambda_{q,1}^{(0)} \end{cases}. \quad (62)$$

It can also be seen that  $z^{(1,N)} = \Lambda_{1,1}^{(1)} = e^{-i\psi_1}$  and therefore,

$$\psi_1 = -\text{Arg}(z^{(1,N)}). \quad (63)$$

We can now work with  $\mathbb{L}^{(2)}$  acting on  $\Lambda^{(1)}$  by seeing how the elements of the second row and column of the new  $\Lambda^{(1)}$  matrix are replaced, zero by zero, by the elements of the second row and column of  $\mathbb{Y}^\dagger$  to make  $\Lambda^{(2)}$ . In order to generalize to any values of  $p$  and  $q$ , we let the general  $z^{(p,q)}$  be the element  $(p, q)$  of the resultant matrix after the matrix  $\lambda^{(p,q)}$  has been applied in the correct sequence. We also define the special case of  $z^{(p,q)}$  when  $p = q$  and when  $p = N$ , in which case we define  $z^{(p,p)}$  as the diagonal  $(p, p)$  element of  $\Lambda^{(p-1)}$ . In general, for any  $\lambda^{(p,q)}$ , we can find its elements and also the elements of  $\mathbb{Y}$  with the following relationships,

$$\left\{ \begin{array}{l} \tan(\theta^{(p,q)}) = \left| \frac{\Lambda_{q,p}^{(p-1)}}{z^{(p,q-1)}} \right| \\ \varphi^{(p,q)} = \text{Arg}(\Lambda_{q,p}^{(p-1)}) - \text{Arg}(z^{(p,q-1)}) \\ z^{(p,p)} = \Lambda_{p,p}^{(p-1)} \\ z^{(p,q)} = e^{i\varphi^{(p,q)}} \cos(\theta^{(p,q)}) z^{(p,q-1)} + \sin(\theta^{(p,q)}) \Lambda_{q,p}^{(p-1)} \\ \psi_k = -\text{Arg}(z^{(k,N)}) \end{array} \right. \quad (64)$$

Thus, starting at  $p = 1$  and  $q = 2$ , we can use the recursive, iterative relationships in Eq. 64 to find all the parameters  $\theta^{(p,q)}$ ,  $\varphi^{(p,q)}$  and  $\psi_k$  that define a given unitary matrix  $\Lambda$ . The MATLAB function `getAngles.m` found in the Appendix C.3 takes an arbitrary unitary matrix and outputs a data structure containing all the angles described in Eq. 64.

## B Stars, Bars and Binary Numbers

Using the unique Stars and Bars method detailed by Feller [2], one can derive the dimensionality of the Fock space in equation 2, generate the complete Fock space of a given Fock state, and assign a unique numeric label to any Fock state.

The Stars and Bars method is used to explore the combinatorics of  $n$  balls being sorted into  $N$  walled bins, but this applies easily to the case of photons being sorted into different modes. Instead of the stars and bars of Feller, we will use binary bits, 1s and 0s. Let the 1s represent the items to be sorted, photons, and 0s represent the figurative walls of the bins, the spatial modes. We need  $n$  total 1s and  $N - 1$  total 0s (as we do not count the outer walls), and thus a  $N + n - 1$  bit number can represent a Fock state. As an example, to represent  $|2, 0, 1\rangle$ , we use the 5-bit binary number 11001, and  $|1, 1, 1\rangle$  would be associated with 10101.

By finding all ways to sort the 1s and 0s in an  $N + n - 1$  bit binary number, we arrive at the dimensionality  $A$  by noting that there are  $(N + n - 1)$  choose  $n$  different ways to sort  $n$  number of 1s and  $N - 1$  number of 0s in an  $N + n - 1$  bit binary number. By generating all these binary numbers, one can retrieve the entire Fock space of which a given Fock state is a member. Two MATLAB functions written by van der Geest, `permpos.m`[3] and `nextpermpos.m`, can be useful for finding all permutations of a given binary number.

The binary numbers associated with Fock states can also be converted into decimal numbers that can be used for compact, easy to read labels for the states  $|\Psi^{(\alpha)}\rangle$  and  $|\Psi^{(\beta)}\rangle$  within a given Fock space that naturally have their own ordering system.

## C MATLAB code

This part of the Appendix contains MATLAB code used to simulate the boson sampler and generate plots. It should be noted that much of the code contains redundant error checking and the core of most functions are relatively short. However, the entirety of the code is shown here for the sake of completeness.

### C.1 Finding Complex Amplitudes of Output Fock States

The function `getScheelMatrix.m` takes a unitary matrix and returns another matrix of the same size modified from the original dependent on the input and output states as described by Scheel [11].

```
function S = getScheelMatrix(B,L,A)
% getScheelMatrix.m constructs a new matrix as described by Scheel (2008).
% This functions creates a new matrix given an input state in the form of a
% row vector A, and ouput state B, and the original matrix L. This new
% matrix is used to find its permanent in order to describe  $\langle B|L|A\rangle$ .

% BEGIN INPUT CHECK
% A and B have to be the same size
[la , NA] = size(A);
[lb , NB] = size(B);
if NA ~= NB
    error('A and B must be the same size. ');
end

% A and B have to be row vectors
if la ~= 1 || lb ~= 1
    error('A and B must be row vectors ');
end

% A and B have to have the same total number of photons
if sum(A) ~= sum(B)
```

```

    error('A and B must have the same number of photons. ');
end

N = NA; % set N equal to vector length of A (same as B's).

% Matrix L has to be a square matrix of size NxN
[N1,N2] = size(L);
if N1 ~= N || N2 ~= N
    error('L must be a square matrix that can multiply A and B');
end

% Matrix L must be unitary
% This is tested by finding L'*L and seeing how close it is to eye(N);
tol = 1e-12; % tolerance of error checking
if max(max(abs(L'*L - eye(N)))) > tol
    error('L must be unitary');
end
% END INPUT CHECK

% create modified matrix
mtag = 1;
ntag = 1;

for n=1:N % first go through each elements of the A and B vectors
    if A(n) ~= 0 % if there is a nonzero number of photons...
        for mp = 1:A(n) % for each photon in this mode...
            Mrow(mtag) = n; % we repeat a row index,
            mtag = mtag + 1; % and increment the index
        end
    end
    if B(n) ~= 0 % same for the vector A as for vector B, but...
        for mp = 1:B(n)
            Ncol(ntag) = n; % we repeat a column index for each photon,
            ntag = ntag + 1; % and increment the other index.
        end
    end
end

for j = 1:length(Mrow) % for each of the repeated indicies...
    for k = 1:length(Ncol)
        % S is created from L using the new indicies gathered into
        % Mrow and Ncol.
        S(j,k) = L(Mrow(j),Ncol(k));
    end
end

end

```

The function `getAmp.m` can be used to find the complex amplitude  $\langle \Psi^{(\beta)} | \hat{S} | \Psi^{(\alpha)} \rangle$ . This code calculates the permanent using `permanentRyser.m` written by Winslow [14].

```
function amp = getAmp(B,L,A)
%getAmp.m -- get the complex amplitude of <B|L|A>
% amp = getAmp(B,L,A) returns the complex amplitude of the objet <B|L|A>
% as related to the problem of indistinguishable boson sampling.
% A is the N-length row vector [n1,n2,...,nN] that represents an input
% state and B is the N-length output state row vector. L must be an NxN
% unitary matrix. This function makes use of the function
% permanentRyser.m written by Winslow (2012).

% BEGIN INPUT CHECK
% A and B have to be the same size
[la , NA] = size(A);
[lb , NB] = size(B);
if NA ~= NB
    error('A_and_B_must_be_the_same_size. ');
end

% A and B have to be row vectors
if la ~= 1 || lb ~= 1
    error('A_and_B_must_be_row_vectors ');
end

% A and B have to have the same total number of photons
if sum(A) ~= sum(B)
    error('A_and_B_must_have_the_same_number_of_photons. ');
end

N = NA; % set N equal to vector length of A (same as B's).

% Matrix L has to be a square matrix of size NxN
[N1,N2] = size(L);
if N1 ~= N || N2 ~= N
    error('L_must_be_a_square_matrix_that_can_multiply_A_and_B');
end

% Matrix L must be unitary
% This is tested by finding L'*L and seeing how close it is to eye(N);
tol = 1e-12; % tolerance of error checking
if max(max(abs(L'*L - eye(N)))) > tol
    error('L_must_be_unitary ');
end

% END INPUT CHECK

% calculate amplitude using the permanent
amp = (1/sqrt(prod(factorial(B))))*(1/sqrt(prod(factorial(A))))*...
    permanentRyser(getScheelMatrix(B,L,A));
```

```
% the function getScheelMatrix.m creates a modified version of L based on
% the input state A and the output state B.
```

```
end
```

## C.2 Generating Random Samples from the Boson Sampling Probability Distribution

The function `getBosonSample.m` returns a specified number of randomly determined possible output states  $|\Psi^{(\beta)}\rangle$  each weighted by the absolute squares of their amplitudes as calculated using `getAmp.m`. The user provides the function with an input state in vector form and a unitary matrix  $\Lambda$ . It should be noted that once the weights have been calculated over the entire Fock space, generating random samples from this distribution is trivial.

```
function samples = getBosonSample(Lambda, inputState, numsamples)
% Generates a sample output state as a boson sampling device.
% getBosonSample.m takes an NxN unitary scattering matrix L and a input
% state represented by a N-length row vector and returns numsamples number
% of samples taken from the probability distribution calculated for a boson
% sampling device.

% BEGIN INPUT CHECKING
tol = 1e-12; % default tolerance

% get the dimensions of input Lambda, Lambda is of size NxM
[N,M] = size(Lambda);

% test to see if Lambda is square
if N ~= M
    error('Matrix must be square. ');
end

% test if Lambda is a scalar and not a matrix of at least 2x2 size
if N < 2
    error('Matrix cannot be a scalar. ');
end

% Finds the largest element in Lambda*Lambda' - I
% to check if Lambda is unitary
check = max(max(abs(Lambda*Lambda' - eye(N))));

% Check to see if any element's modulus is larger than the tolerance
if check > tol
    error('Matrix must be unitary. ');
end
```

```

% check to see if inputstate is an N-length row vector
[m,j] = size(inputState);
if j ~= N
    error('Input state vector must the same dimension as Lambda. ');
end
if m ~= 1
    error('Input state vector must be a row vector. ');
end

% check to see if inputstate has real values
if ~isreal(inputState)
    error('Input state must contain real nonnegative integers. ');
end
% force inputstate to have nonnegative integer values
inputState = abs(round(inputState));

% check numsamples
if ~isreal(numsamples) || numsamples < 1
    error(['Number of samples must be a real positive number ', ...
        'greater than 0. ']);
end
% END INPUT CHECKING

n = sum(inputState);           % get number of photons
N = length(inputState);      % get number of modes
D = N + n - 1;               % find number of bits needed

A = getBiOutputAmps(Lambda, inputState);
    % calculate all amplitudes of each Fock state in binary form

A = cell2mat(A);             % convert data struct to matrix for ease of use.
    % the last column will contain amplitudes
weights = (abs(A(:,end)).^2)';
    % convert amplitudes to probabilities and make a row vector

data = A(:,1:end-1);         % get rows of binary numbers
data = bi2de(data, 'left-msb');
    % convert remaining rows of binary numbers to base-10 into row vec.

decimal_samples = datasample(data, numsamples, 'Weights', weights);
    % collect random samples with native MATLAB function datasample

    % convert decimal state labels into Fock states one by one
for j=1:numsamples
    samples{j} = sb2Fock(de2bi(decimal_samples(j), D, 'left-msb'));
end
end

```

The function `getBiOutputAmps.m` is the workhorse used by `getBosonSample.m`. It takes as input a row vector representing the input state and the unitary matrix  $\Lambda$  and outputs a data structure with each Fock state within the relevant Fock space represented as binary numbers in ascending order, each with their own corresponding complex amplitude. As  $N$  and  $n$  increase, the data structure consumes poorly-scaling memory resources, and the loop can take unreasonable calculation time to calculate permanents for each member of the Fock space.

```
function outputData = getBiOutputAmps(Lambda, inputState)
% Returns a data structure containing all members of a Fock space and
% their respective complex amplitudes in binary ascending order.
% getBiOutputAmps takes the modal scattering NxN matrix L and an input
% row vector of length N and generates the complex amplitudes of each Fock
% state in the scattered superposition. The states are contained in the
% first column of the data structure and are arranged in ascending binary
% order. The second column contains the complex amplitude.

% BEGIN INPUT CHECKING
tol = 1e-12; % default tolerance

% get the dimensions of input Lambda, Lambda is of size NxM
[N,M] = size(Lambda);

% test to see if Lambda is square
if N ~= M
    error('Matrix must be square. ');
end

% test if Lambda is a scalar and not a matrix of at least 2x2 size
if N < 2
    error('Matrix cannot be a scalar. ');
end

% Finds the largest element in Lambda*Lambda' - I
% to check if Lambda is unitary
check = max(max(abs(Lambda*Lambda' - eye(N))));

% Check to see if any element's modulus is larger than the tolerance
if check > tol
    error('Matrix must be unitary. ');
end

% check to see if inputstate is an N-length row vector
[m,n] = size(inputState);
if n ~= N
    error('Input state vector must the same dimention as Lambda. ');
end
if m ~= 1
```

```

        error('Input_state_vector_must_be_a_row_vector. ');
    end

    % check to see if inputstate has real values
    if ~isreal(inputState)
        error('Input_state_must_contain_real_nonnegative_integers. ');
    end
    % force inputstate to have nonnegative integer values
    inputState = abs(round(inputState));
    % END INPUT CHECK

    % generate all members of the Fock space given the input
    FockSpace = getBiFockSpace(inputState);

    % get the dimensionality A of the Fock Space.
    [A,x] = size(FockSpace);

    % calculate complex amplitude for each member of the space
    for a = 1:A
        outputData{a,1} = FockSpace(a,:);
        % Fock state stored in first column
        outputData{a,2} = ...
            getAmp( sb2Fock(FockSpace(a,:)), Lambda, inputState );
        % amplitudes store in second column
        % sb2Fock converts the binary number into a Fock state using
        % the "stars and bars" method.
    end
end
end

```

The function **getBiFockSpace** returns a matrix whose rows contain each member of a Fock space in binary form given a starting Fock state. It uses the “stars and bars” method detailed by Feller [2] and discussed in Appendix B. The function **permpos.m** written by van der Geest [3] is used to find all permutations of the different ways to sort  $n$  photons into  $N$  modes.

```

function FockSpace = getBiFockSpace(v)
% return the entire Fock space given an input row vector v
% getBiFockSpace returns the entire Fock space that the state represented
% by the row-vector v is a member. Each state is converted into a binary
% number and are arranged in ascending order by the value of the
% corresponding binary number. This function makes use of permpos.m
% written by van der Geest (2007).

% BEING INPUT CHECK
% check to see if inputstate is an N-length row vector
[m,nv] = size(v);
if m ~= 1 || nv < 2
    error(['Input_state_vector_must_be_a_row_vector_with_at_' ],...)
end

```

```

        [ 'least_2_elements' ]);
end
% check to see if inputstate has real values
if ~isreal(v)
    error('State_vector must contain real nonnegative integers. ');
end
% force inputstate to have nonnegative integer values
v = abs(round(v));
% END INPUT CHECK

n = sum(v); % find number of photons
N = length(v); % find number of modes
D = N + n - 1; % find number of bits needed
FockSpace = de2bi(sort(bi2de(permpos(n,D), 'left -msb')), 'left -msb');
% First finds all permutations of "stars and bars" using permpos.m, then
% converts these binary numbers to decimals, sorts them in ascending order,
% then reconverts them back to binary numbers for output.

end

```

The function **sb2Fock.m** takes a vector containing a binary number that represents the “stars and bars” and converts it to a vector that represents a Fock state. For instance, it converts the vector [0,1,0,1,1] into [0,1,2].

```

function vp = sb2Fock(v)
% converts a vector containing bits of a binary number into a Fock state.
% sb2Fock.m takes a vector containing bits of a binary number and converts
% it into another vector that represents a Fock state. The notion of
% converting a Fock state to a binary number is taken from the "stars and
% bars" method developed by Feller (1957).

D = length(v); % get the bit length
n = sum(v); % find the total number of photons
N = D - n + 1; % find the number of modes
vp = zeros([1,N]); % start with an N-length row vector of all zeros
k = 1; % current index to be incremented

% go through a loop examining each bit
for d=1:D
    if v(d) == 0 % if there is a "wall" ,...
        k = k + 1; % don't add a photon and keep going
    elseif v(d) == 1 % if there is a photon in this bit ,...
        vp(k) = vp(k) + 1; % add it to the current "bin," or mode
    else
        error('Invalid_vector!');
        % gives an error if any element of the vector is not a 0 or 1.
    end
end
end

```

end

### C.3 Finding Angles that Describe an Arbitrary Unitary Matrix

The MATLAB function `getangles.m` accepts a unitary matrix as input and returns a data structure that contains the angles that describe beam splitters and phase shifters that could be arranged in a boson sampler that would scatter the photon modes as described by the original input matrix.

```
function A = getAngles(Lambda)
% The function getAngles takes an arbitrary unitary matrix Lambda and
% returns the data structure A that contains the calculated angles.
% The elements {p,q} of the data structure A contain a two element vector
% [theta,phi] that contain the values of the angles at the point {p,q}.
% The element {1,1} of A contains the diagonal Y matrix.

    tol = 1e-12;    % default tolerance

% get the dimention of input Lambda, Lambda is of size NxM
[N,M] = size(Lambda);

% test to see if Lambda is square
if N ~= M
    error('Matrix_must_be_square. ');
end

% test if Lambda is a scalar and not a matrix of at least 2x2 size
if N < 2
    error('Matrix_cannot_be_a_scalar. ');
end

% Finds the largest element in Lambda*Lambda' - I
% to check if Lambda is unitary
check = max(max(abs(Lambda*Lambda' - eye(N))));

% Check to see if any element's modulus is larger than the tolerance
if check > tol
    error('Matrix_must_be_unitary. ');
end

% Lp is equal to the transpose conjugate of original unitary matrix for
% the first step, it will be run through the loop to extract
% the rotation matrices from Lambda.
Lp = Lambda';
% Ltest is used to check if the generated matrices multiplied by
% themselves in order yield back the original matrix Lambda.
Ltest = eye(N);
```

```

% begin main loop for generating lambda with indices p,q
% 1 <= p <= q - 1, 2 <= q <= N
for p = 1:N-1
    for q = p+1:N
        z_pp = Lp(p,p);      % set values for z_pp and z_pq
        z_pq = Lp(q,p);
        % using equation for theta:
        theta = atan(abs(z_pq/z_pp));
        % using equation for phi:
        phi = angle(z_pq)-angle(z_pp);

        % adjusting angels for ease of reading
        if theta > 2*pi
            theta = theta - 2*pi;
        end
        if theta < -2*pi
            theta = theta + 2*pi;
        end
        if phi > 2*pi
            phi = phi - 2*pi;
        end
        if phi < -2*pi
            phi = phi + 2*pi;
        end

        % makeX creates the Chi matrix for given angles for testing
        X = makeX(N,[p,q],[theta,phi]);

        % Multiply Lp by lambda in order to find new z, and to ensure
        % accuracy of the method (i.e. the correct elements are set to
        % zero).
        Lp = X*Lp;
        % z_pp = Lp(p,q);
        % Error checking by multiplying all the Lambdas by each other
        % in order to see if we can recreate the original U
        Ltest = X*Ltest;
        A{p,q} = [theta,phi];
        % store theta and phi in the {p,q} element of the output data
        % structure A
    end
end

Y = conj(Lp);
% the diagonal Y matrix is the conjugate of the last matrix

A{1,1} = Y;
% the element {1,1} of the output data structure contains the Y matrix
Ltest = Y*Ltest; % Finally, multiply the last Ltest matrix by Y.

```

```

% check to see if Utest is the same as the original U matrix
check2 = max(max(abs(Ltest - Lambda)));

% If Utest is not the same as the original U within the tolerance, then
% the fuctions returns an error message
if check2 > tol
    disp('Algorithm_failed!');
end

end

```

The function **makeX.m** is a simple function that constructs a particular  $\chi^{(p,q)}$  matrix given  $N, p, q, \theta^{(p,q)}$  and  $\varphi^{(p,q)}$ . It is used within **getAngles.m** for error checking.

```

function X = makeX(N, index, angles)
% This function returns an NxN matrix X constructed from an identity matrix
% with the (p,p), (p,q), (q,p) and (q,q) elements replaced with values
% determined by [theta, phi] taken from the angles input. The indices [p,q]
% are contained within the index input.

% define p, q, th and phi from the inputs
p = index(1);
q = index(2);
th = angles(1);
phi = angles(2);

% check to see if p and q are valid indicies
if p >= q
    error('Index_'p'_'must_be_less_than_index_'q''.');
end
if p >= N || q > N
    error('Invalid_indicies. ');
end

% starting with the NxN identity matrix
X = eye(N);

% replace elements of X depending on th and phi
X(p,p) = exp(i*phi)*cos(th);
X(p,q) = sin(th);
X(q,p) = -exp(i*phi)*sin(th);
X(q,q) = cos(th);

end

```

## C.4 Generating Random Unitary Matrices

The function `getMedrazzi.m` generates a random unitary matrix taken from the Haar measure following the method described by Medrazzi [7].

```
function U = getMedrazzi(N)
% Return a random NxN unitary matrix taken from the Haar measure
% This function takes the input N and returns a random unitary NxN matrix
% taken from the Haar measure as described by Medrazzi 2007. The document
% can be found at: http://arxiv.org/abs/math-ph/0609050

% step #1, p. 11
A = (1/sqrt(2))*(rand(N) + i*rand(N)); % A is random normal complex matrix
% step #2
[Q,R] = qr(A); % Q and R matrices from QR decomposition
% step #3
D = eye(N);
for n=1:N
    D(n,n) = R(n,n)/abs(R(n,n));
    % set diagonal elements to normalized diagonal elements of R
end
% step #4
U = Q*D; % U is the final random unitary matrix

end
```