PADERBORN UNIVERSITY

DEGREE THESIS

Simulating Measurement-induced Non Linearity with a Quantum-Optical Computational toolbox

Author:
Jaime MILLA VAL

Supervisor:
J.-Prof. Dr. Tim BARTLEY

A thesis submitted in fulfillment of the requirements for the degree of Physics

in the

Mesoscopic Quantum Optics
Department of Physics

July 11, 2018

Declaration of Authorship

I, Jaime MILLA VAL, hereby declare that I prepared this thesis entirely on my own and have not used outside sources without declaration in the text. Any concepts or quotations applicable to these sources are clearly attributed to them. This thesis has not been submitted in the same or substantially similar version, not even in part, to any other authority for grading and has not been published elsewhere.

City, Date	Signature

PADERBORN UNIVERSITY

Abstract

Faculty of Science
Department of Physics

Physics

Simulating Measurement-induced Non Linearity with a Quantum-Optical Computational toolbox

by Jaime MILLA VAL

Generating measurement-induced non linearity with a **BS** (Beam splitter) is an easy way to produce entangled states and potentially reduce the quadratures uncertainty as **BS** is a simple tool that can be implemented in a huge number of experimental setups without too much difficulty. The importance of this kind of states arises in the fields of quantum information and communication and metrology.

To study this method, simulations, with the help of the software package QuTiP for Python, of three different setups involving **BS**, detectors and phase selectors have been done varying different parameters of the setup. In a simulation with one **BS** and a detector a reduction of the quadratures uncertainty has been observed. It has been found that same levels of entanglement are reached with one **BS** or two **BS** for a photon and a coherent state as input states.

Acknowledgements

I would like to thank my supervisor, J.-Prof. Dr. Tim BARTLEY, who gave me the opportunity of working with him in his research group. I also would like to highlight his dedication and all the time and help he gave me during writing and doing this work. I also want to thank to Jan Philipp Höpker for his comments and advice during the revision of this work. I am really grateful. Also a special mention to all my previous teachers and professors who definitely gave me the basis to be able, today, to finish this work. I also want to thank all my family and friends who always were there to support me.

Contents

Chapter 1

Introduction

Quantum mechanics, more specifically, Quantum Optics, is a topic that will be around and linked to most of the parts and issues of this work. The aim of this work is to study a simply and easy way to generate entangled and/or squeezed states through measurement-induced non linearity. The applications of that kind of states go from quantum communications to metrology methods.

Specifically, in the following sections using one of the main tools of the Quantum Optics, the beam splitter (**BS**), different states of light (single photons, coherent and squeezed states) will be studied through some computer experiments and simulations.

1.1 Applications and historical review of Quantum Optics. Motivation of the work

The way individual quanta of light (photons) interact with atoms and molecules is a regarding issue of Quantum Optics [33]. This field of study is mainly the consequence of joining quantum theory and light.

According to Gerard Milburn (an expert in quantum optics from the University of Queensland in Australia) this branch of knowledge date back to the 1960s [41]. Roy Glauber (then at Harvard University) started it with his study about the optical coherence of quantized electromagnetic fields – thanks to this study he was granted a Nobel Prize in 2005, having published his first results in 1963 [1]. In the 1970s the prediction and observation of photon antibunching was achieved thanks to the study of the quantum features of photon-counting statistics [7][36]. Later on, in the 1980s, complementary "wave" aspect of light was studied, especially focusing on phase-dependent properties such as squeezing [25]. In the 1990s, the non-classical aspects of entanglement arise as the major area of investigation. Also in the 1990s, two main branches of study made important progress, quantum information and new fields of atomic condensates [41]. Quantum Optics became ideal for testing new and advanced concepts and ideas developed in quantum information theory has reached a huge level of success since then [35][47]. A lot of features, thoughts and predictions of quantum theory (such as teleportation and the violation of Bell's inequality) have been proven in the Quantum Optics field [28][13]. With this precedents, Quantum Optics (especially on the direction of communication and computing) will continue producing remarkable advances in the future [44].

Applications are currently limited by the available hardware, particularly to photon detectors and single photon on-demand sources. As the technology to improve these devices advances, the number of applications will increase and their quality will improve. One remarkable fact is the important advances in the scaling of quantum optical systems nowadays and the development of sophisticated integrated optical circuits [29].

But not only applications of quantum optics are linked to quantum computing and communication. Also, quantum theory plays an important role in aspects of metrology and measurement schemes. By preparing quantum states in a specifics way, it is possible to squeeze the uncertainty of measurements to levels never reached before [4].

But achieving useful states for this kind of applications is sometimes difficult. Normally it requires to induced some kind of nonlinearities [5]. However these types of effects are often neglected in the most common scenarios of the applications of interest (low photon flux, photon-photon interaction...) [5][8]. Nevertheless, some of these photon-photon interactions at low light flux levels have been achieved. Some of them through single atoms in cavities, atomic ensembles, many-body physics with strongly interacting photons... [8], using quantum catalysis [5] and so on.

Particularly, in this work, measurement-induced non linearity is simulated with the action of **BS**. The objective is to get entangled states or a reduction in some of the quadrature or both for quantum communication applications or to obtain measurements with lower levels of uncertainly.

Why generate measurement induced no linearity with **BS**? A **BS** is used because it is a simple tool (also relatively cheap) that can be manipulated easily inside the setup of an experiment that requires entangled or squeezed states. Also, a **BS** could be implemented in an integrated optical circuit resulting in a powerful, valuable and compact element that could be used in a lot of applications mentioned above.

With the help of **BS** it is going to be generated measurement-induced non linearity. A quantum toolbox in Python (QuTiP [see section 2.4] for more details about this software) is going to be used in this work to simulate three computer experiments. The setup consists basically of one or two beam splitters and some detectors differently arranged depending on the concrete simulation. The input states that are going to be studied and tested in the different setups are: single photons, coherent states and squeezed states. A further and more detailed explanation of each simulation is described in Chapter 3.

Chapter 2

Theoretical background

In this chapter the most important elements that are used in this work are going to be explained:

Wigner function (Section 2.1) – Wigner function and quadratures.

Beam Splitter (Section 2.2) – Theory and input states.

Entanglement measures (Section 2.3) – *What is?* and *log-negativity*.

QuTiP (Section 2.4) – Description of the software used package. Comparing dimensions size of Hilbert space.

2.1 Wigner function

In quantum mechanics there is an uncertainty principle that makes it impossible to know both q or x (position) and p (momentum) at the same time. In the standard formulation of quantum mechanics one works with probability densities instead. It would be desirable to have a single function that could display the probability in both x and p. The Wigner function is a function constructed to do that although it will not have a direct physical interpretation as a probability distribution we know from classical physics. For example the Wigner function can be negative in regions of phase space, which have no physical meaning if one thinks of it as a probability distribution. [6]

The quadratures operators $\hat{X}$ and $\hat{P}$ are the dimensionless equivalent to position and momentum defined in terms of the creator and annihilation operators as:

$$\hat{X} = \frac{1}{2}(\hat{a}^\dagger + \hat{a}) \tag{2.1}$$

$$\hat{P} = \frac{i}{2}(\hat{a}^\dagger - \hat{a}) \tag{2.2}$$

The definition of the Wigner function given a state ψ is:

$$W(x,p) = \frac{1}{2\pi\hbar}\int dy e^{\frac{-ipy}{\hbar}}\psi(x+\frac{y}{2})\psi^*(x-\frac{y}{2}) \tag{2.3}$$

In this work, all the computed Wigner functions are calculated through a QuTiP specific given routine.

2.2 Beam Splitter

2.2.1 Theory

The beam splitter is the basic element of the setups that are going to be simulated. With a beam splitter an input light beam can be sent to different output arms, it depends on the splitting ratio. A schematic is shown in figure 2.1).

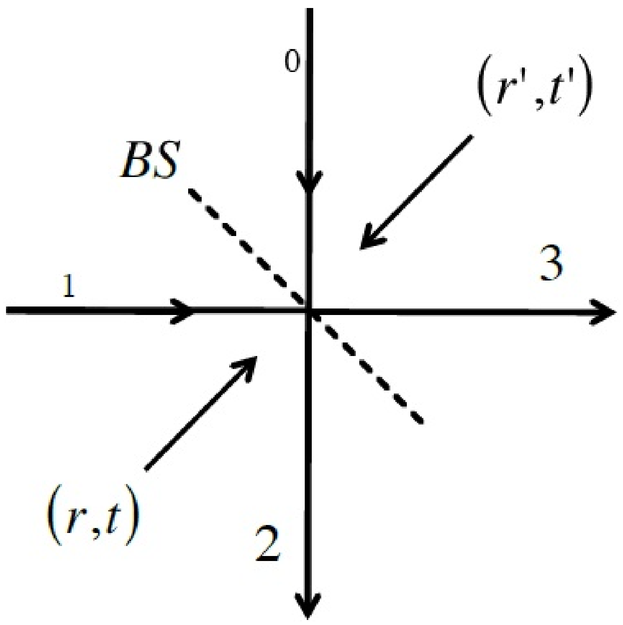


FIGURE 2.1: A general **BS** ([49] *FIG 1* modified).

In the figure 2.1, 0 and 1 are the input modes and 2 and 3 the output modes of the **BS**.

The annihilation operators $\hat{a}$ of the different incident fields is transformed following relation [9]:

$$\begin{pmatrix} \hat{a}_2 \\ \hat{a}_3 \end{pmatrix} = \begin{pmatrix} t' & r \\ r' & t \end{pmatrix} \begin{pmatrix} \hat{a}_0 \\ \hat{a}_1 \end{pmatrix} \tag{2.4}$$

The coefficients have to fulfil the so called reciprocity relations:

$$|r'| = |r| \tag{2.5}$$

$$|t| = |t'| \tag{2.6}$$

$$|r|^2 + |t|^2 = 1 \tag{2.7}$$

$$r^* t' + r' t^* = 0 \tag{2.8}$$

$$r^* t + r' t'^* = 0 \tag{2.9}$$

So finally we get:

$$\hat{a}_0^\dagger = t' \hat{a}_2^\dagger + r' \hat{a}_3^\dagger \tag{2.10}$$

$$\hat{a}_1^\dagger = r \hat{a}_2^\dagger + t \hat{a}_3^\dagger \tag{2.11}$$

In the general case, there is a phase relation between the coefficients that depends on the construction of the **BS**. r', t' may differ from r, t respectively depending on the way the **BS** is made. But, for simplicity, we are going to assume a simple case in which our **BS** is lossless, $r = r'$ and $t = t'$ and the reflected beam suffers a $\frac{\pi}{2}$ phase shift. With this assumptions, and taking into account that in a general case:

$$\begin{pmatrix} \hat{a}_0^\dagger \\ \hat{a}_1^\dagger \end{pmatrix} = \begin{pmatrix} t' & r' \\ r & t \end{pmatrix} \begin{pmatrix} \hat{a}_2^\dagger \\ \hat{a}_3^\dagger \end{pmatrix} \tag{2.12}$$

the matrix of a 50 : 50 **BS** would be:

$$B = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix} \tag{2.13}$$

As the transformation is lossless, B has to be unitary. This means $B^\dagger B = \mathbb{I}$, that in this case is clearly satisfied.

2.2.2 Input states

Now we are going to see how we can transform the different input states. Especially we will focus on three main states that will be combined in different ports of the **BS**:

- Single Photon
- Coherent state
- Squeezed state

Single Photon

In the following picture (Figure 2.2) the Wigner function of a single photon is visualized.

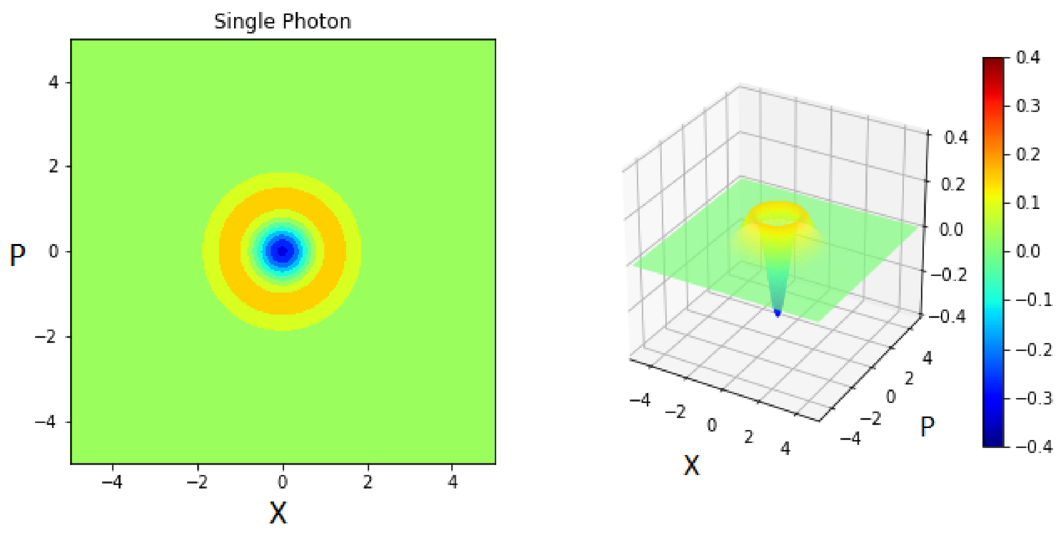


FIGURE 2.2: Wigner function of a photon (made with own code, dimension of Hilbert space of 7)

The next step to be able to describe a photon after a **BS** is to write it in terms of annihilation operators. In this case $|1\rangle = \hat{a}^\dagger |0\rangle$ so the whole system after the **BS** would be (following Eq 2.10):

$$|1\rangle_0 |0\rangle_1 = \hat{a}_0^\dagger |0\rangle_0 |0\rangle_1 \xrightarrow{\textbf{BS}} (t'\hat{a}_2^\dagger + r'\hat{a}_3^\dagger) |0\rangle_2 |0\rangle_3 \tag{2.14}$$

This means, that in the case of a 50 : 50 **BS** you can find the photon in any output port with equally probability.

Coherent state

A coherent state is a quantum state of the quantum harmonic oscillator whose dynamics is closely resembling the oscillatory behavior of a classical harmonic oscillator. The Wigner function of a coherent state is displayed in the following picture (Figure 2.3).

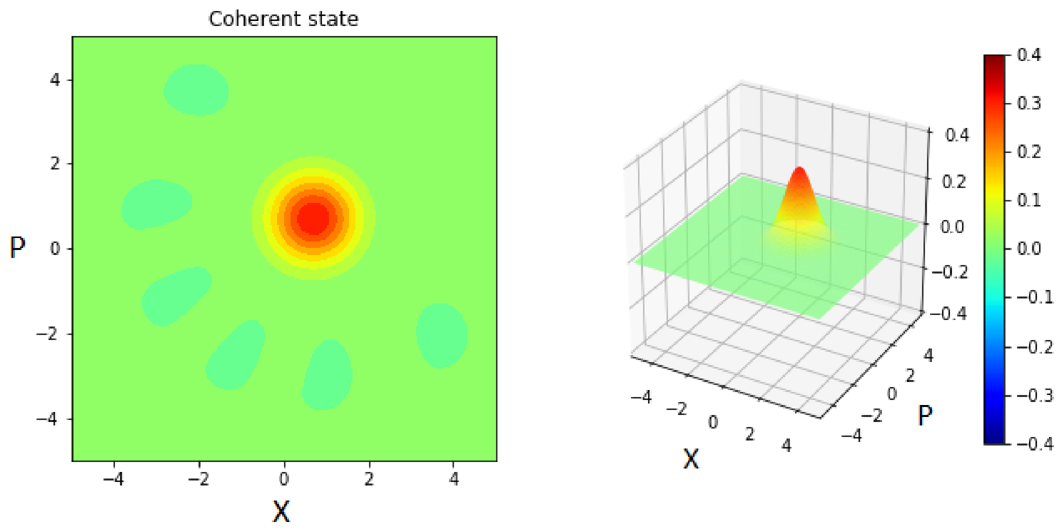


FIGURE 2.3: Wigner function of a coherent state with $\alpha = 0.5 + 0.5i$ (made with own code, dimension of Hilbert space of 7)

Now, to describe a coherent state after a **BS** we need to define the displacement operator that give us a coherent state from the vacuum $|\alpha\rangle = \hat{D}(\alpha)|0\rangle$ [10].The displacement operator $\hat{D}(\alpha)$ is defined as:

$$\hat{D}(\alpha) = \exp\left\{\alpha\hat{a}^{\dagger} - \alpha^{*}\hat{a}\right\} \tag{2.15}$$

With $\alpha \in \mathbb{C}$ being the parameter of the coherent state (the so called amplitude of the state).

Now introducing a coherent state in the Port 1 (following Eq 2.11) the whole system after the **BS** can be described as:

$$|0\rangle_0 |\alpha\rangle_1 = \hat{D}_1(\alpha)|0\rangle_0 |0\rangle_1 \xrightarrow{\textbf{BS}} \exp\left\{\alpha(r\hat{a}_2^{\dagger} + t\hat{a}_3^{\dagger}) - \alpha^{*}(r\hat{a}_2^{\dagger} + t\hat{a}_3^{\dagger})^{\dagger}\right\}|0\rangle_2 |0\rangle_3 \tag{2.16}$$

Squeezed state

Squeezed states have the variance of one quadrature reduced, while the other must increase to preserve the Heisenberg uncertainty relation. The Wigner function of a squeezed state is displayed in the following picture (Figure 2.4).

Now, to describe a squeezed state after a **BS** we need to define first the "squeeze" operator that gives us a squeezed state from the vacuum $|\alpha\rangle = \hat{S}(\xi)|0\rangle$ [11].$\hat{S}(\xi)$ is defined as:

$$\hat{S}(\xi) = \exp\left\{\frac{1}{2}(\xi^{*}\hat{a}^{2} - \xi\hat{a}^{\dagger 2})\right\} \tag{2.17}$$

With $\xi \in \mathbb{C}$ being the parameter of the squeezed state.

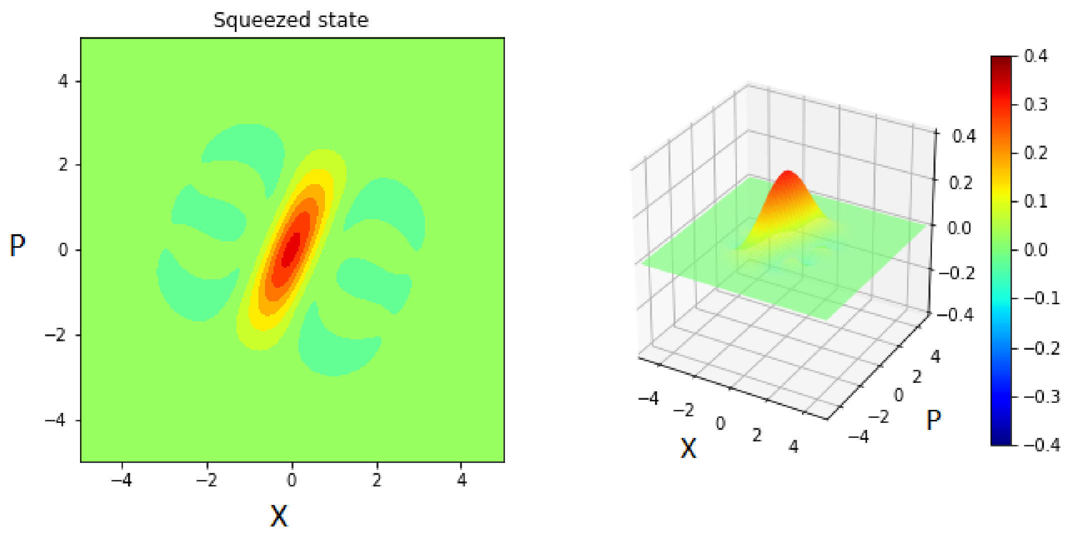


FIGURE 2.4: Wigner function of a squeezed state with $\xi = 0.5 - 0.5i$ (made with own code, dimension of Hilbert space of 7)

After the action of the **BS** the final state would be:

$$|0\rangle_0 |\xi\rangle_1 = \hat{S}_1(\xi) |0\rangle_0 |0\rangle_1 \xrightarrow{\mathbf{BS}} \exp\left\{ \frac{1}{2}(\xi^*(r\hat{a}_2^\dagger + t\hat{a}_3^\dagger)^{\dagger 2} - \xi(r\hat{a}_2^\dagger + t\hat{a}_3^\dagger)^2) \right\} |0\rangle_2 |0\rangle_3 \tag{2.18}$$

2.3 Entanglement measures

In the following sections entanglement is going to be described. Also a way to measure it through the logarithmic negativity is going to be given.

2.3.1 *What is entanglement?*

Entanglement is a quantum phenomenon, with no classical equivalent, in which the quantum states of two or more objects must be described by a single state that involves all the objects in the system, even when the objects are spatially separated. This leads to correlations between observable physical properties. But a more detailed and rigorous definitions is given by [24]:

"The definition of entangled state is made in terms of the physical resources needed for the preparation of the state: a multipartite state is said to be entangled if it cannot be prepared from classical correlations using local quantum operations."

Given the typical example for quantum communications of *Alice* and *Bob* (see Figure 2.5), a local quantum operation (**LOCC**) is an operation performed in either Alice or Bob's states (such as measurement of the state, a quantum gate like Hadamard, CNOT...). In the abbreviation (**LOCC**) of this kind of operation over this types of system is also included **CC** that stands for *Classical Communication*. It refers to the classical channel which also exists between *Alice* and *Bob*.

Although the most common way to generate entangled states is the Spontaneous Parametric Down Conversion (**SPDC**), in this work entangled states are also generated by passing different input states (Single photon, coherent states and squeezed

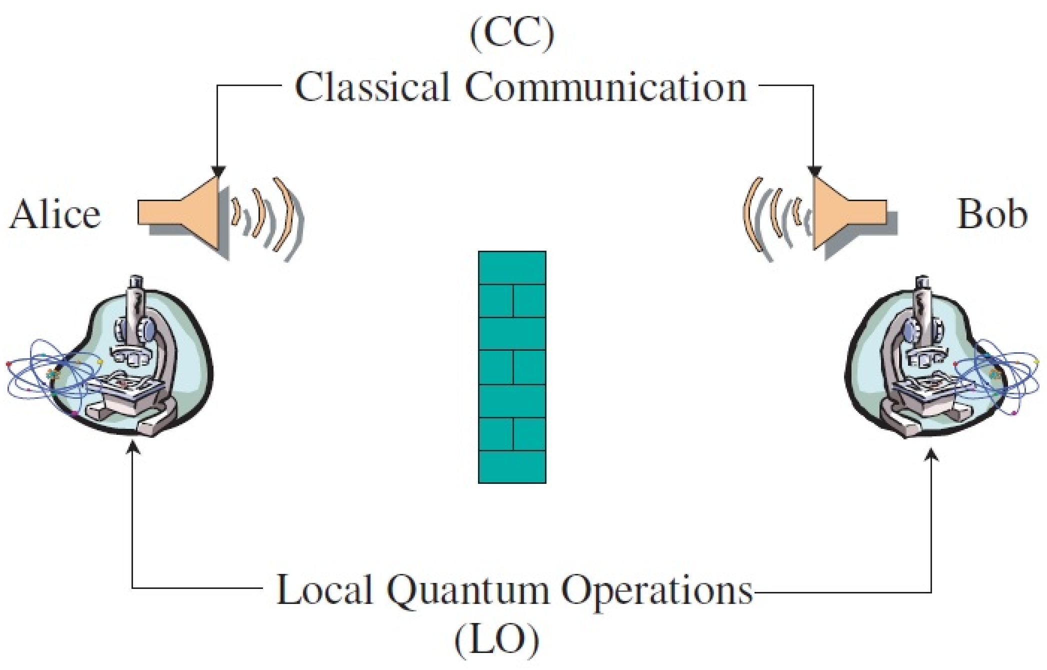


FIGURE 2.5: Standard Quantum Communication setup [26]

states) through a **BS**. Moreover, it exists a lot of procedures to achieve entangled states such as [14]:

- entangled photon pairs from calcium atoms [22].
- entangled ions prepared in electromagnetic Paul traps [30].
- entangled atoms in quantum electrodynamic cavities [21].
- long-living entanglement between macroscopic atomic ensembles [39].
- entangled microwave photons from quantum dots [23].
- entanglement between nuclear spins within a single molecule [14].
- entanglement between light and an atomic ensembles[32].

2.3.2 Logarithmic Negativity

Entanglement is essential in a huge amount of quantum communication processes. That is why it is also crucial to measure and understand it to be able to quantify entanglement [38]. For this purpose, a quantity called *Logarithmic Negativity*, $E_N(\rho)$, is defined. Given, for example, a bipartite [1] quantum system represented by a density matrix in a given orthonormal basis $\rho = \sum \rho_{ij,kl} |i\rangle\langle j| \otimes |k\rangle\langle l|$ the *partial transposition* with respect to party B is given by [26]:

[1] A system with two parts (in this case, particles) is called bipartite. During the simulations that are going to take place in this work, the system may be higher than only a bipartite state. It is the case when two **BS** are used, where we have a four partites system, as it will be explained later in the Chapter 3.

$$\rho^{T_B} = \sum_{i,j,k,l} \rho_{ij,kl} \, |i\rangle\langle j| \otimes |l\rangle\langle k| \tag{2.19}$$

With this, $E_N(\rho)$ can be defined as [26]:

$$E_N(\rho) := \log_2 \left\| \rho^{T_B} \right\| \tag{2.20}$$

Where $\|X\|$ is the so called *trace norm*, defined as [26]:

$$\|X\| := Tr\sqrt{X^{\dagger}X} \tag{2.21}$$

The properties of $E_N(\rho)$ are:

- can be computed very easily [26].
- is monotic under **LOCC** and *positive partial transpose*[2] (**PPT**) operations [38].
- is an upper bound to distillable entanglement [48].
- possesses an operational interpretation as a special type of entanglement cost under **PPT** operations [3].

The second property arise from the fact that logarithmic negativity is neither convex nor concave. However, this convexity is just a mathematical requirement for entanglement monotones that generally does not correspond to a physical process in which a quantum system may loss information [38].

2.4 QuTiP

2.4.1 Description

QuTiP [17][18] is a Quantum Toolbox in Python and the main tool used in the simulations made in this work. It is an open-source software for simulating the dynamics of open quantum systems. QuTiP aims to provide user-friendly and efficient numerical simulations of a wide variety of Hamiltonians, including those with arbitrary time-dependence, commonly found in a wide range of physics applications such as quantum optics, trapped ions, superconducting circuits, and quantum nanomechanical resonators [43].

This software is developed in the Python language programming and the QuTiP library depends on other numerical Pyhon packages like *NumPy* [34], *SciPy* [46][19] and *Cython* [12]. Moreover, to be able of seeing and evaluating all the different outputs and results, the *Matplotlib* [27][15] plotting package is also used.

All programs, codes and Python routines have been written and run in a Jupyter [20] environment, which is developed with a IPython [16][37] kernel.

2.4.2 Comparing different Hilbert space dimensions

While working with QuTiP, you have to take care of some parameters and details. One of these parameters is the dimension of the Hilbert space you are working in. This variable is a kind of convergence parameter and may affect to the performance,

[2]Positive partial transpose (**PPT**) operations are a kind of operations that preserve the positiveness of the partial transpose across all possible bi-partite splits[26]. For a deeper explanation of the **PPT** conditions see [40].

quality and computational time of the current simulation. For this reason, it has to be cleverly chosen trying to achieve an equilibrium between quality and required computational time of the simulation. Of course, you have to take into account the resources and computational power. It is clear that, the higher dimension you use, the better result's quality you will get. On the other hand, the higher Hilbert space dimension you use, the more computer demanding it will be.

To try to evidence the dependence of the Hilbert space dimension, N, with the overall performance of the simulation a test is going to be done. The test consists of calculating the Wigner function of a single photon, coherent state and squeezed state several times with different dimension in each time through QuTiP routines. The technical information of which machine is used to run the simulations or the specific version of the software packages can be consulted in Appendix A.1.

In the Appendix A.2 in the Figures A.1, A.2 and A.3 the results of the simulations are shown and the times required are listed in the Table A.3.

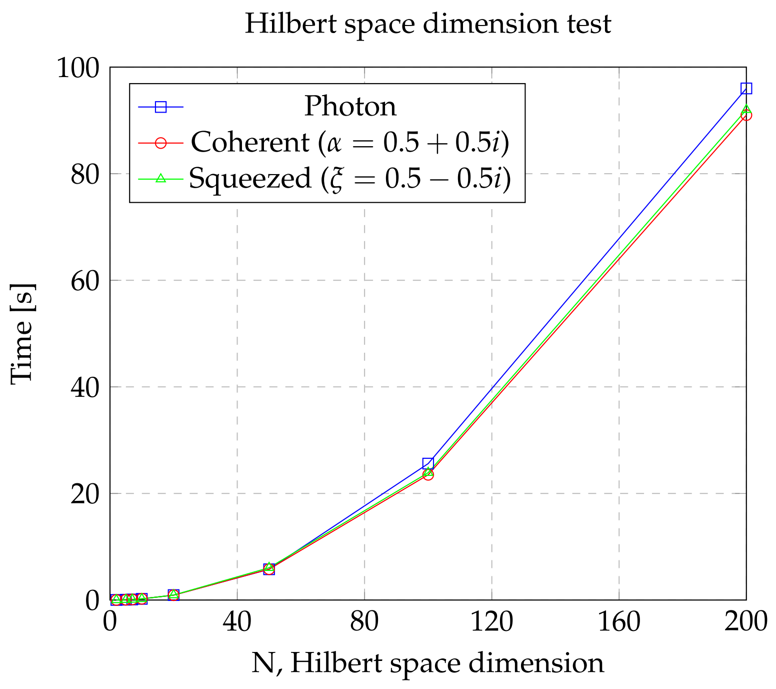


FIGURE 2.6: Graph of the required time for a given state and a Hilbert space dimension to compute the Wigner function.

There is almost no dependency between the input states and the time required to compute the Wigner function. It just increases as N increases, as shown in figure 2.6. This increment is quite big, for example:

- going from $N = 5 \rightarrow N = 10$ the required time increases by a factor of $\sim \times 4$.
- going from $N = 10 \rightarrow N = 20$ the required time increases by a factor of $\sim \times 5$.
- going from $N = 10 \rightarrow N = 100$ the required time increases by a factor of $\sim \times 100$.

As you work with higher N the factor by which the required time is multiplied also increases.

Talking about results quality, single photon does not care about N because it only need a basis of two elements ($|0\rangle$ and $|1\rangle$) to completely define itself. Coherent and squeezed states improve (less blue spots around the centre [Figures A.2, A.3] and a more "squeezed" figure in the case of squeezed states) as increasing dimension up to $N \sim 20$ more or less comparing the results with those given in [42] and [45] (only squeezed states).

For a N higher than 20 the only appreciable change is the increment in the required time.

It has to be taken into account that this test consists just computing the Wigner function, a simple, easy and quick task comparing the operations in the simulations in the following sections (Chapter 3).

For this reason and the fact that the used device for the simulations in this work is a domestic laptop (see Appendix A.1 for more details) , a Hilbert space dimension of $N = 7$ has been chosen. Seeing the results of the test (Figures A.1, A.2, A.3) it can be ensured that the quality of the following simulations will be good enough. It was not possible to set a higher N because of memory problems on the used machine. It has to be said that it might exist a way of optimising the used software and the routines in which, maybe, you may fully exploit all the resources the devices has.

Chapter 3

Simulations

As explained in previous sections, with the help of BS, measurement-induced non linearity is generated to produce quantum states for applications such as quantum communication.

There are three different systems which are going to be simulated with various input states and varying the value of some of their parameters. Each system correspond to one simulation:

- Simulation 1 – Only one **BS**. The entanglement of the final state is calculated. (Section 3.1)
- Simulation 2 – One **BS** also, but with a detector in one of the output ports. Here the quadratures are studied. (Section 3.2)
- Simulation 3 – Two identical **BS** with detectors in between. Wigner function in different stages of the system and entanglement are computed varying parameters such as transmission of the **BS** or the phase between the states of the two channels of the setup. (Section 3.3)

The possible options as input states in the simulations are combinations of:

- Single photon – $|1\rangle$
- Squeezed state – $|\xi\rangle$
- Coherent state $|\alpha\rangle$
- Vacuum $|0\rangle$

As there are already a lot of parameters that can be varied in the simulations, the decision of setting α and ξ fixed has been taken to try to simplify the parameter analysis. The coherent amplitude is $\alpha = 1$ and the squeezed parameter is $\xi = 0.5 - 0.5i$. The values of these parameters pretend to be as general as possible and of the order of the more widely used ones (based on [49] and [5]). Note that, for example, a change in the phase of α or ξ, just results in a rotation of the Wigner functions of the states which would be irrelevant for what is going to be studied in this work.

In all following simulations the Hilbert space dimension, N, is set to $N = 7$ as explained in section 2.4.2. If we call t the transmission and r the reflectivity of the **BS** we have:

$$t = |t| \in \mathbb{Z} \tag{3.1}$$

$$r = |r|i \in \mathbb{C} \tag{3.2}$$

In simulations 2 and 3, the action of the detectors have been implemented by multiplying the final state, after the **BS**, by an operator that projects into $|1\rangle\langle 1|$ the modes where the detectors are while keeping the rest. For simulation 2 this operator is:

$$P_D^{sim2} = |1\rangle\langle 1|_A \otimes \mathbb{I}_B \tag{3.3}$$

And for simulation 3:

$$P_D^{sim3} = |1\rangle\langle 1|_{A1} \otimes |1\rangle\langle 1|_{A2} \otimes \mathbb{I}_{B1} \otimes \mathbb{I}_{B2} \tag{3.4}$$

Where $\mathbb{I}$ is the identity matrix and $A's$ refer to the detector's modes and $B's$ refer to the output modes without detection. So if we call $|\psi'\rangle$ the state after the **BS**, the final state after the detectors action is given by:

$$|\psi\rangle = P_D\,|\psi'\rangle \tag{3.5}$$

The variance of the quadratures has been calculated by a QuTiP routine. From a given operator, specifically $\hat{X}$ and $\hat{P}$ (defined in eq 2.1 and 2.2), it returns:

$$\langle\Delta\hat{X}\rangle^2 = \langle\hat{X}^2\rangle - \langle\hat{X}\rangle^2 \tag{3.6}$$

$$\langle\Delta\hat{P}\rangle^2 = \langle\hat{P}^2\rangle - \langle\hat{P}\rangle^2 \tag{3.7}$$

The machine in which the simulations have been run is a personal laptop , to see more details about the device see table A.1 . The version of the used software is presented in table A.2. All programs, codes and Python routines that were used to run the following simulations have been written and run in a Jupyter [20] environment, which is developed with a IPython [16][37] kernel. A copy of the developed code for running this simulations can be seen in Appendix A.3.

3.1 Simulation 1 - One beam splitter and entanglement

3.1.1 Setup and method

In the Figure 3.1 the setup for the simulation 1 is presented.

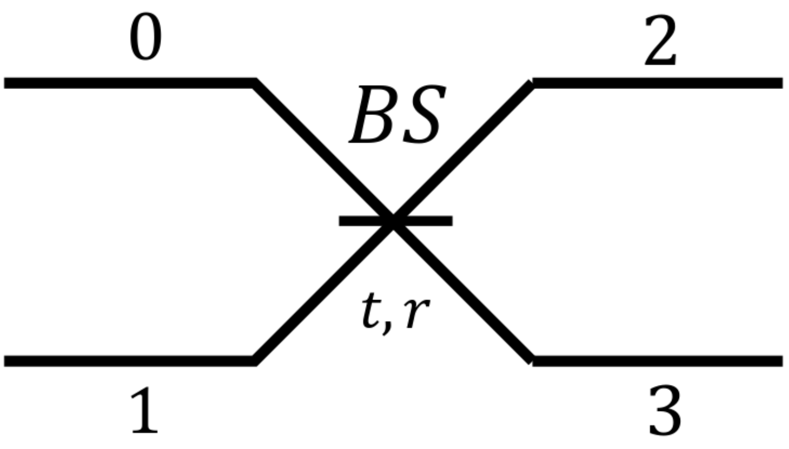


FIGURE 3.1: Setup of the simulation 1

The input ports of the **BS** are labelled as 0 and 1, with t and r being the transmission and reflectivity respectively. During the simulations, different combinations of states will be placed in the 0 and 1 ports. Some aspects of the resulting output will be studied (entanglement mainly) for a set of different values of t.

Remembering the assumptions made about the **BS** parameters (equations 3.1, 3.2) and what it was said in the **BS** section (section 2.2) the creation operators transform as follow by the action of the **BS**:

$$\hat{a}_0^\dagger = t\hat{a}_2^\dagger + r\hat{a}_3^\dagger \tag{3.8}$$

$$\hat{a}_1^\dagger = r\hat{a}_2^\dagger + t\hat{a}_3^\dagger \tag{3.9}$$

With this and knowing how to express the different input states in terms of the creator operators (see section 2.3) it is able to calculate the final states through the **BS**.

3.1.2 Results

Here some results of interest are presented. It consist of two simulations with two specific values for t with, as input state, the combination (with concrete values for $\alpha \neq 1$): single photon + coherent state (Figure 3.2) just to check the results given by the developed code with those in [49]. Also entanglement measurements for different combinations of input states over a set of values of t (Figure 3.3) are studied (this time $\alpha = 1$).

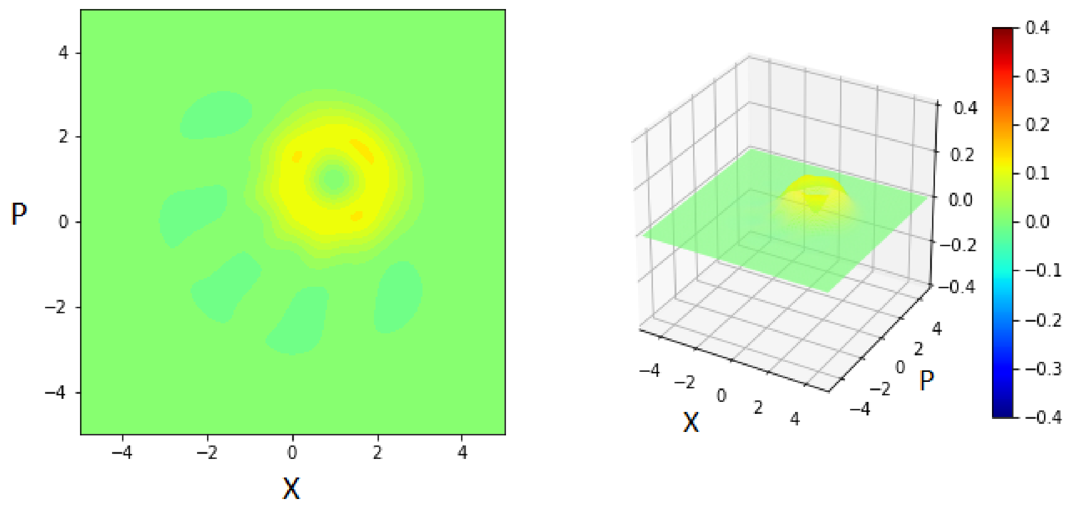
As it can be observed from figure 3.2, the results are almost the same. This fact gives reliability to the code and the following results.

What one can read from the Figure 3.3 is that coherent state does not contribute to the entanglement, at least through the action of a **BS**, just looking at Coherent + Vacuum and Coherent + Coherent figures (3.3e and 3.3h) that display an entanglement (log-negativity) of 0. For example, a Photon + Vacuum (Figure 3.3d) or a Squeezed state + Vacuum (Figure 3.3f) have the same entanglement as Photon + Coherent state (Figure 3.3a) or a Squeezed state + Coherent state (Figure 3.3c). This is an expected result since coherent states are classical and entanglement is a quantum feature.

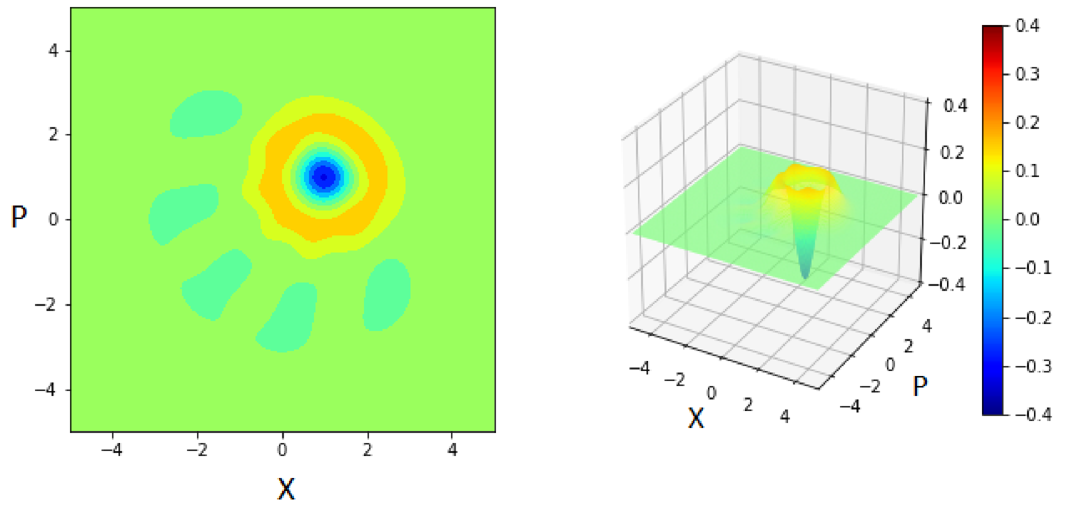
Also we have for the case of Photon + Vacuum (Figure 3.3d) for $t = 1/\sqrt{2}$ a $E_N = 1$ which match perfectly with what is said on [2].

For the setup of this simulation, just one **BS**, the maximal E_N is achieved at $t = 1/\sqrt{2}$. More concrete, in the case of Photon + Photon (Figure 3.3g) a $E_N \approx 2.5$, for the case of Squeezed state + Photon (Figure 3.3b) a E_N slightly higher than 1 is also achieved.

From this results it can be concluded that, if you need as much entanglement as you can, the best option is to use a 50 : 50 **BS** with a Photon as input for each port. However, for a concrete experiment (e.g. in quantum communication), it may be more appropriate not to use such as combination because it could be too difficult to implement in the setup. To solve these there are more combination of input states that gives entanglement such as only one Photon + Vacuum/Coherent state, Squeezed state + Vacuum/Coherent state or Squeezed state + Photon.



(A) Photon + Coherent state. $\alpha = \sqrt{2}e^{i\pi/4}$ and $t = 1/\sqrt{2}$



(B) Photon + Coherent state. $\alpha = 10e^{i\pi/4}$ and $t = 1/10$

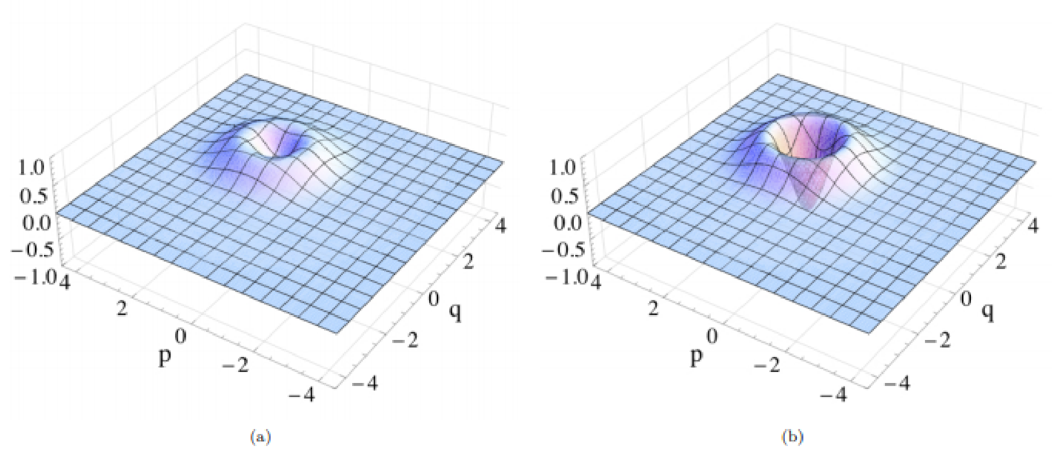
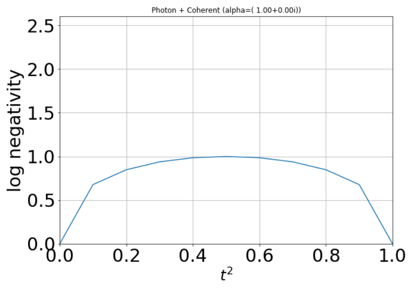


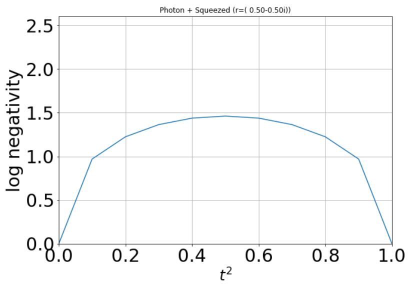
FIG. 2. Wigner function ($\times\pi$, $\hbar = \omega = 1$) of the beam splitter output 3, Eq. (12). It is a statistical mixture between a coherent state and a displaced Fock state. (a) 50:50 beam splitter ($t = 1/\sqrt{2}$) with the coherent input state $|\alpha = \sqrt{2}e^{i\pi/4}\rangle$. (b) Highly reflective beam splitter ($t = 1/10$) with the coherent input state $|\alpha = 10e^{i\pi/4}\rangle$. Note, that $t\alpha$ is identical for both (a) and (b).

(C) Figure obtained from [49]

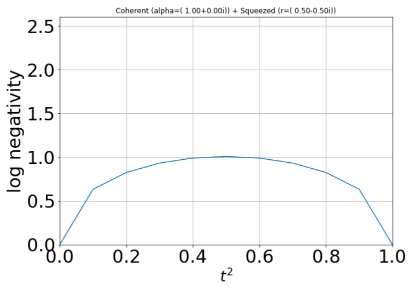
FIGURE 3.2: Simulation 1. Checking the code with the results given in [49].



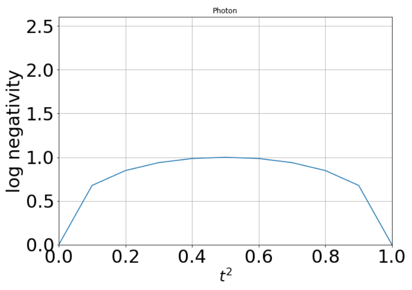
(A) Photon + Coherent state



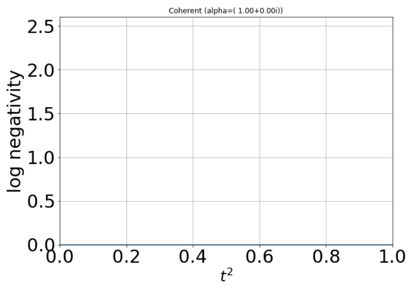
(B) Squeezed state + Photon



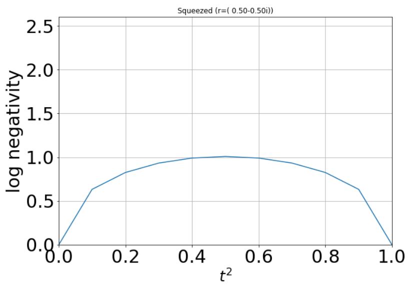
(C) Squeezed state + Coherent state



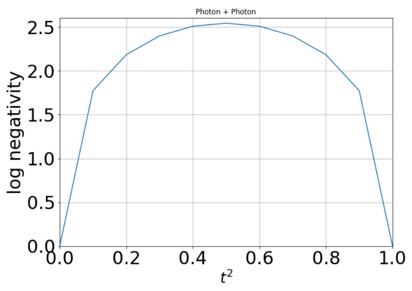
(D) Photon + Vacuum



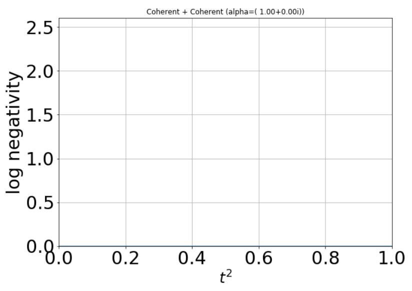
(E) Coherent state + Vacuum



(F) Squeezed state + Vacuum



(G) Photon + Photon



(H) Coherent state +Coherent state

FIGURE 3.3: Simulation 1. Entanglement for different input states varying t.

3.2 Simulation 2 - One beam splitter and a detector

3.2.1 Setup and method

In the following picture (Figure 3.4) the setup of the simulation 2 is shown.

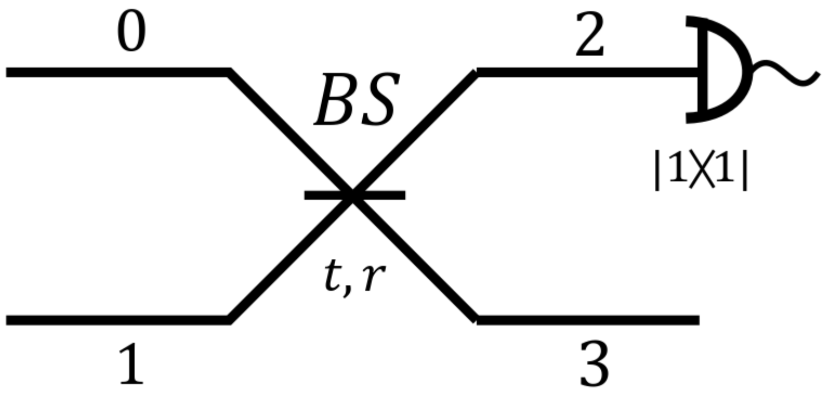


FIGURE 3.4: Setup of the simulation 2

It is mainly similar to the setup of simulation 1 (Figure 3.1) but with a difference that the detector is at the end of the port 2. This detector consists just of a projection of the state onto a single photon state. As before (Section 3.1), different combinations of states will be placed in the input ports (0 and 1) and some aspects of the resulting output will be studied (this time we will focus on Wigner functions and quadratures essentially) for different values of t.

The transformation of the creator operators are the same as in the simulation 1 (equations 3.8, 3.9).

3.2.2 Results

In the figure 3.5 the variance of the quadratures obtained for the case Photon + Coherent state is presented.

From the figure 3.5 can be seen the reduction over the variance of $\hat{X}$ of the state in the mode 3 for the case of Photon + Coherent state. The highest reduction is achieved for $t^2 = 0.7$ which is the same results as in [5].

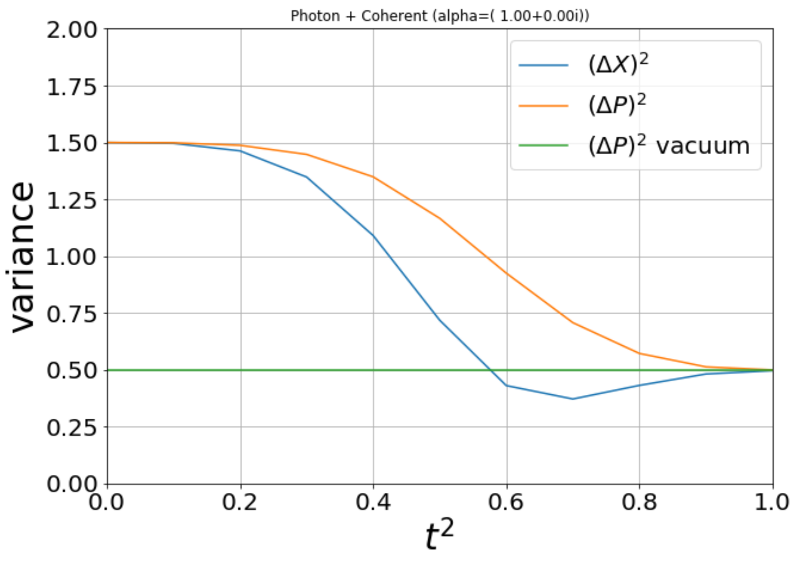


FIGURE 3.5: Simulation 2. Variance of quadratures for the case Photon + Coherent state varying t.

3.3 Simulation 3 - Two beam splitters

3.3.1 Setup and method

In the Figure 3.6 the setup for the simulation 3 is presented.

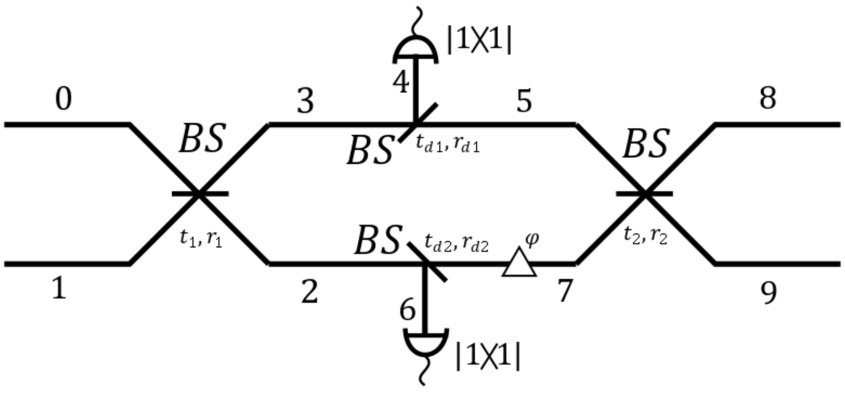


FIGURE 3.6: Setup of the simulation 3

In contrast to the previous simulations now we have two identical **BS** (four in fact if we count the two that are placed for the detectors) and two detectors between them. Also a phase selector is placed in one of the parts of the setup (7 looking at the Figure 3.6).

The parameters of the **BS** for the detectors, t_{d1}, r_{d1}, t_{d2} and r_{d2}, are fixed to a balanced beam splitters:

$$t_{d1} = t_{d2} = \frac{1}{\sqrt{2}} \tag{3.10}$$

$$r_{d1} = r_{d2} = \frac{1}{\sqrt{2}} i \tag{3.11}$$

If one detector should be removed it would be enough setting $r_{d(1,2)} = 0$ and, of course, $t_{d(1,2)} = 1$. For this case, the transformation of the creator operators in the different stages of the setup is a little bit more complicated than in the previous simulations.

$$\hat{a}_0^\dagger = t_1 \hat{a}_2^\dagger + r_1 \hat{a}_3^\dagger \tag{3.12}$$

$$\hat{a}_1^\dagger = t_1 \hat{a}_3^\dagger + r_1 \hat{a}_2^\dagger \tag{3.13}$$

$$\hat{a}_2^\dagger = t_{d2} \hat{a}_7^\dagger e^{i\varphi} + r_{d2} \hat{a}_6^\dagger \tag{3.14}$$

$$\hat{a}_3^\dagger = t_{d1} \hat{a}_5^\dagger + r_{d1} \hat{a}_4^\dagger \tag{3.15}$$

$$\hat{a}_5^\dagger = t_2 \hat{a}_9^\dagger + r_2 \hat{a}_8^\dagger \tag{3.16}$$

$$\hat{a}_7^\dagger = t_2 \hat{a}_8^\dagger + r_2 \hat{a}_9^\dagger \tag{3.17}$$

Although the two **BS** are identical, t_1, r_1, t_2, r_2 have been remained separated to avoid losing generality. In what follows, we will refer to t_1, t_2 as just t and r_1, r_2 as just r.

3.3.2 Results

The dependency of entanglement while varying t and φ is displayed in the figure 3.7.

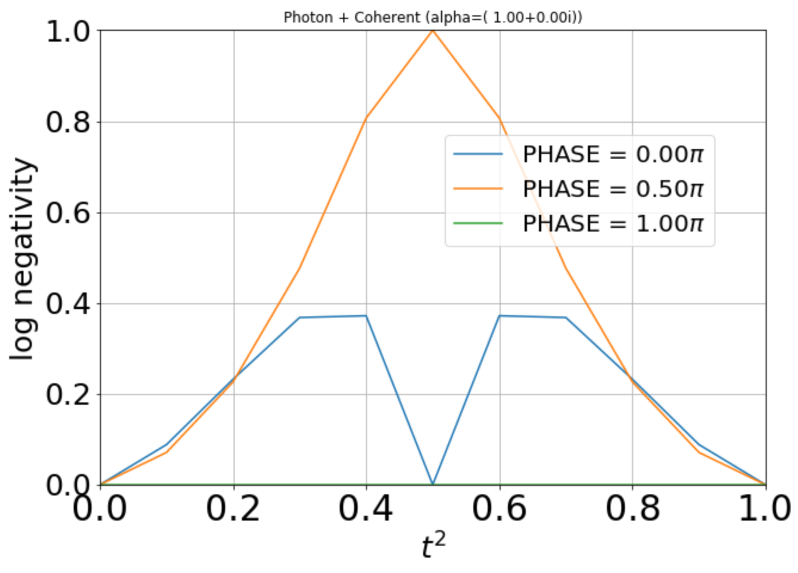


FIGURE 3.7: Simulation 3. Entanglement varying t and φ for the case Photon + Coherent.

It is a special case when $\varphi = \pi$ because there is no entanglement in the final state for any value of t. We can think in terms of an interference experiment and take into account that each **BS** introduces a phase shift of $\pi/2$ to analyses the φ dependency. With this, for a $\varphi = \pi$ the final phase difference is 2π so there is no phase difference between the two output states. Comparing with the results in the simulation 1, the same level of entanglement has been achieved for $\varphi = \pi/2$. Note that a $\varphi = \pi/2$ is the same phase difference that introduces just one **BS** in simulation 1.

One last consideration about this results is when $t^2 = 0.5$. In this case, the variation of φ plays a important role. For a $\varphi = 0$ there is no entanglement while for $\varphi = \pi/2$ the maximum is achieved because of the same interference argument used above.

Chapter 4

Conclusions

Generating measurement induced no linearity with a **BS** is an easy way to produce entangled states because a **BS** is a simple tool that can be implemented in a huge number of experimental setups without no too much difficulties. This entangled states are really useful in a lot of applications and protocols for quantum information and communication.

Simulations, with the software package QuTiP for Python, for three different setups involving **BS**, detectors and phase selectors have been done varying different parameters of the setup (such as transmission of the **BS**, the phase...) to study this method of producing entanglement.

A fact that appears in all executed simulations is that coherent states do not contribute to the entanglement at all for this kind of setups as it is a classical state. For this reason coherent states are not suitable to use to get entanglement with the setups presented in this work. Nevertheless, in the simulation 2 it can be seen a reduction in the variance of $\hat{X}$ of the state in the mode 3 for the case of Photon + Coherent state. The highest reduction is achieved for $t^2 = 0.7$ which is the same results as in [5].It has been found that same levels of entanglement are reached with one **BS** or two **BS** for a photon and a coherent state as input states.A special case occurs in the experiment 3 when $\varphi = \pi$ is set. No entanglement is generated. A remarkable consideration about the simulation 3 results is when $t^2 = 0.5$. In this case, the variation of φ plays a important role. For a $\varphi = 0$ there is no entanglement while for $\varphi = \pi/2$ the maximum is achieved because of interference arguments.

There are some improvements or further investigations about these topics. Some would be: Adding losses to system to make it more realistic. In the real setups a big problem is the losses in system and a good way to handle them would be simulating them. It could be easily added to the simulations by just inserting additional **BS** in each step you want to introduce a loss as done in [31]. Another one is putting more **BS** in a row to see how this may affect the entanglement. It has been seen that for two **BS** not too much entanglement has been obtained. But this is not indication for this would be a rule of the more **BS** you have the less entanglement you get. Lastly to deepthly explore all the available possibilities which offer this setups of **BS**a good idea is to see how the entanglement (or other features of the states) changes when you use no identical **BS** (in contrast to the experiment 3) or other combinations of transmissions and phases.

Appendix A

Appendix

A.1 Technical information

TABLE A.1: Device details used for the simulations

Type	Personal laptop
Model	Asus A550L (2014)
Operating system	Windows 10 Home 64 bits
Processor	Intel(R) Core(TM) i7-4510U CPU @ 2.00GHz up to 2.60 GHz
Cores	2
Virtual logic processors	4
RAM memory	8 GB | DDR3 DIMM 1600 MHz

TABLE A.2: Used software version

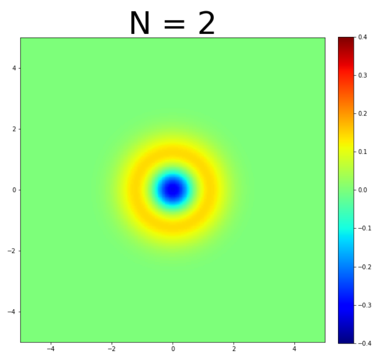
Software	**Version**
QuTiP	4.2.0
Numpy	1.13.1
SciPy	0.19.1
matplotlib	2.0.2
Cython	0.26.1
IPython	6.1.0
Python	3.6.2 | Anaconda custom (64-bit)

A.2 Hilbert space dimension test. Graphs and tables

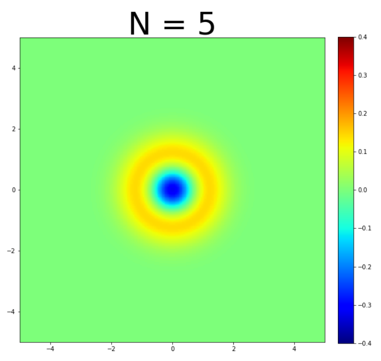
In this Appendix some graphs and tables regarding to the Hilbert space dimension test made in section 2.4.2 are presented.

N	**Time (s)**		
	Photon	**Coherent ($\alpha = 0.5 + 0.5i$)**	**Squeezed ($\xi = 0.5 - 0.5i$)**
2	0.018	0.018	0.018
5	0.060	0.062	0.062
7	0.107	0.111	0.106
10	0.220	0.237	0.217
20	0.927	0.910	0.938
50	5.770	5.790	6.060
100	25.600	23.500	23.900
200	96.000	91.000	92.000

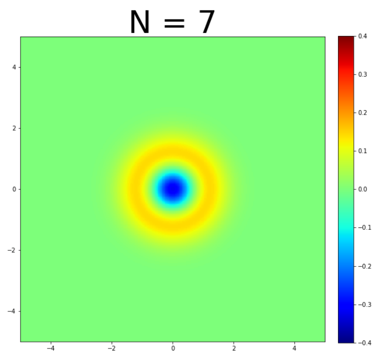
TABLE A.3: Time required for a given state and a Hilbert space dimension to compute the Wigner function.



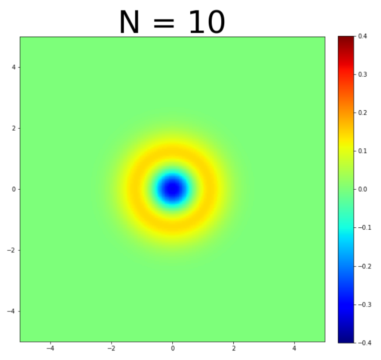
(A)



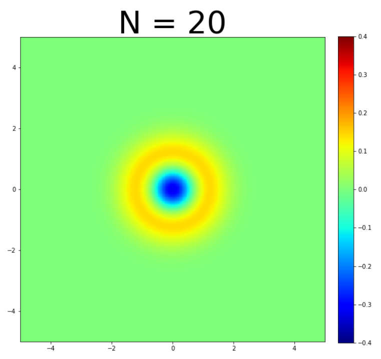
(B)



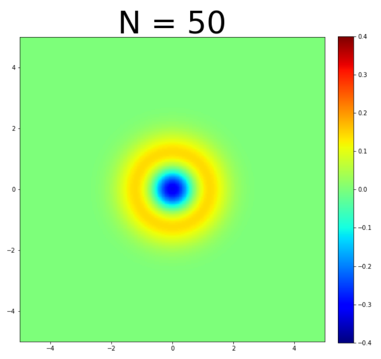
(C)



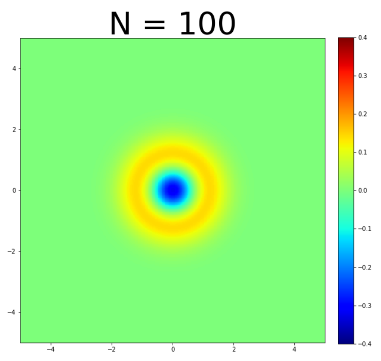
(D)



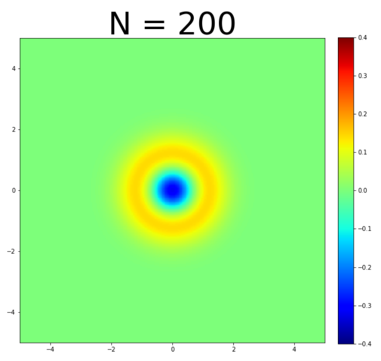
(E)



(F)

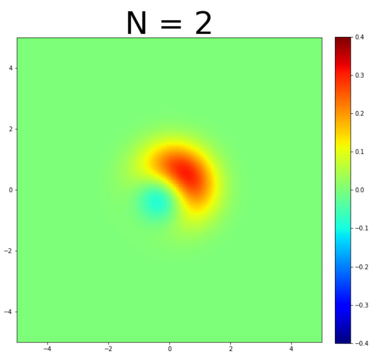


(G)

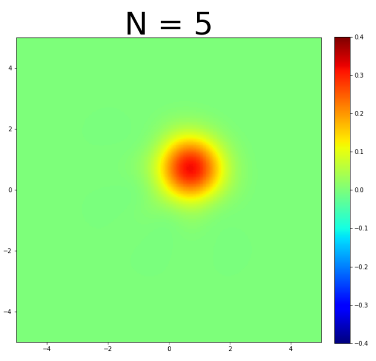


(H)

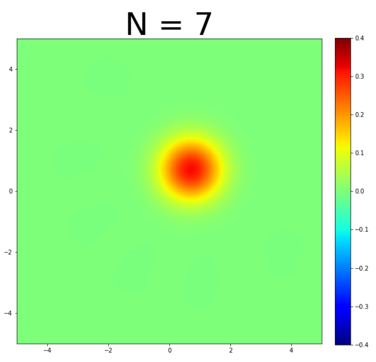
FIGURE A.1: Hilbert dimension test made with single photons



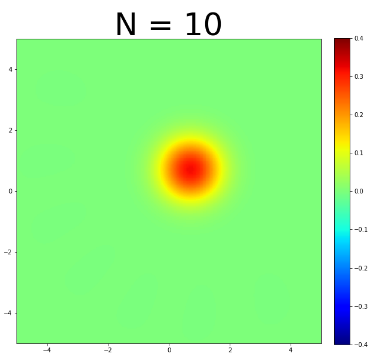
(A)



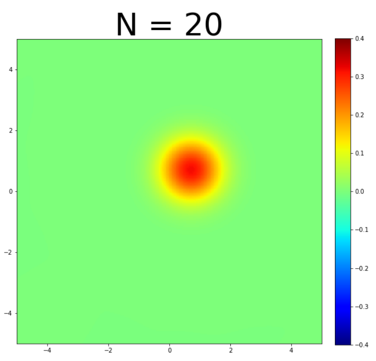
(B)



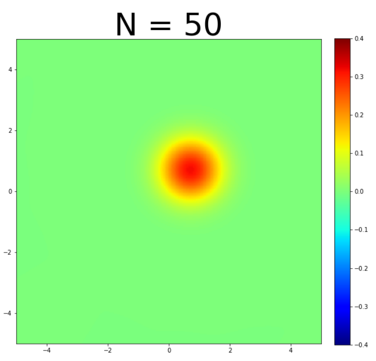
(C)



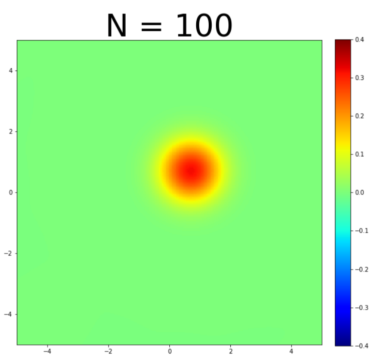
(D)



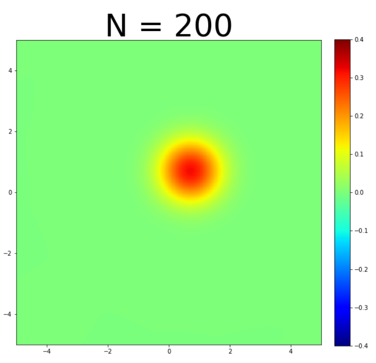
(E)



(F)

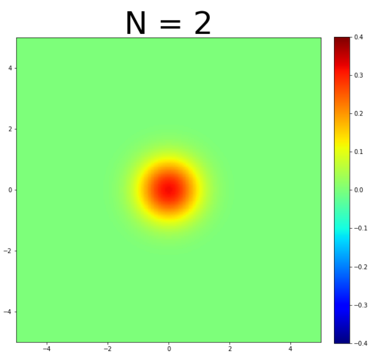


(G)

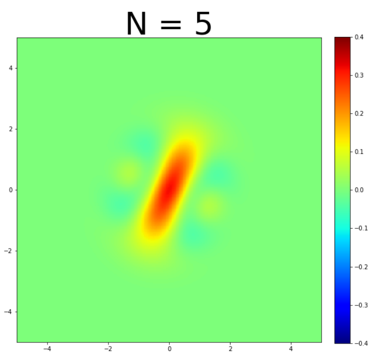


(H)

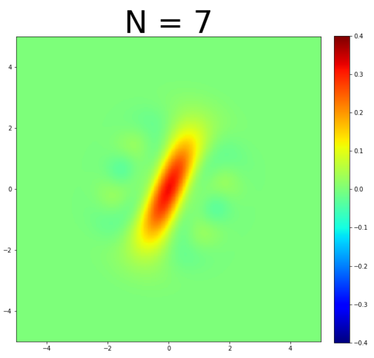
FIGURE A.2: Hilbert dimension test made with coherent states ($\alpha = 0.5 + 0.5i$)



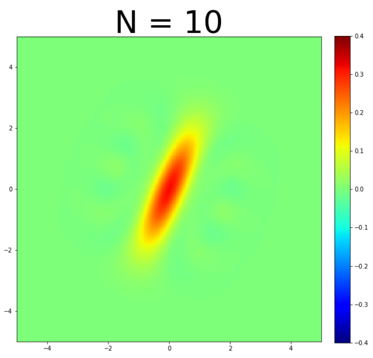
(A)



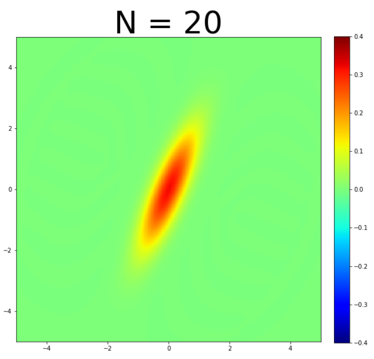
(B)



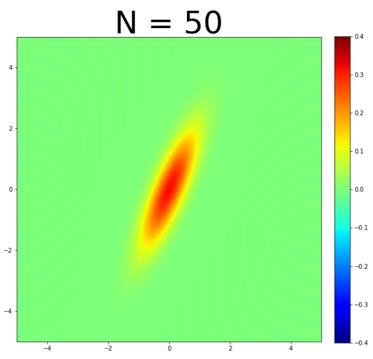
(C)



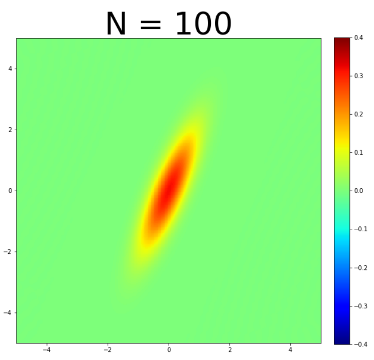
(D)



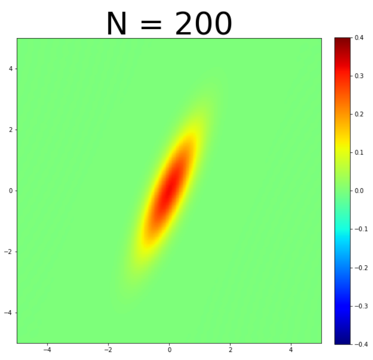
(E)



(F)



(G)



(H)

FIGURE A.3: Hilbert dimension test made with squeezed states ($\xi = 0.5 - 0.5i$)

A.3 Developed code for the simulations

```
In [ ]: # Jupyter commands
        %matplotlib inline
        %config InlineBackend.figure_format = 'retina'

        # Import the required libraries
        import matplotlib.pyplot as plt
        import numpy as np
        import imageio
        import subprocess
        import os
        import sys

        from qutip import *
        from mpl_toolkits.mplot3d import Axes3D
        from matplotlib import cm
        from pylab import *

In [ ]: # Number of photons
        N = 7

        def plot_wigner (state, T, titulo, direc):
            complex_format = "(% .2f%+.2fi)"
            xvec = np.linspace(-5, 5, 300)
            X, Y = np.meshgrid(xvec, xvec)

            W = wigner(state.unit(), xvec, xvec)

            fig = plt.figure(figsize = (10, 10))
            plt.subplot(1, 2, 1, aspect = 'equal')
            plt.title(titulo + ' $\eta$=' + str(round((T/100.00), 2)))
            vmin = - 0.4
            vmax =   0.4
            patata = plt.contourf(xvec, xvec, W, vmin = vmin, vmax = vmax, cmap=cm.jet)

            ax = fig.add_subplot(122, projection = '3d', aspect = 'equal')

            c = ax.plot_surface(X, Y, W, rstride=2, cstride=2, cmap=cm.jet,lw=.1, vmin= vmin,
        vmax= vmax);
            ax.set_xlim3d(-5,5);
            ax.set_ylim3d(-5,5);
            ax.set_zlim3d(vmin, vmax)
            fig.colorbar(c, fraction=0.046, pad=0.04)

            savefig(direc + '/' + titulo + '_eta=' + '%.2f'%(T/100.00) +".png");
            close()

            return direc + '/' + titulo + '_eta=' + '%.2f'%(T/100.00) +".png", W;


        def coefficients (N, psi_fin, T, titulo, direc):
        #     It returns an array with the pair (n,m) for each coefficients
        #     of the basis |n,m> for a given "psi_fin" state
        #     Also it plots the results
            coefficients = []

            for n in np.arange(N):
                for m in np.arange(N):
                    try:
                        toco = tensor(fock(N, n), fock(N, m)).dag() * ket2dm(psi_fin) *
        tensor(fock(N, n), fock(N, m))
                    except TypeError:
                        toco = tensor(fock(N, n), fock(N, m)).dag() * psi_fin * tensor(fock(N,
        n), fock(N, m))
                    patata = (n, m), float(toco[0][0][0])
                    coefficients.append(patata)

            coefficients = np.array(coefficients)

            fig = plt.figure(figsize = (10, 10))
            plt.title(titulo + ' $\eta$=' + str(round((T/100.00), 2)))
        ax = fig.add_subplot(111, projection = '3d', aspect = 'equal')

        _x = np.arange(N)
        _y = np.arange(N)
        _xx, _yy = np.meshgrid(_x, _y)
        x, y = _xx.ravel(), _yy.ravel()

        ax.bar3d(x, y, np.zeros_like(coefficients[:,1]), 1, 1, coefficients[:,1])
        ax.set_ylabel('n')
        ax.set_xlabel('m')
        ax.set_zlabel('c_nm^2')
        ax.set_zlim3d(0, 0.4)
        ax.set_xlim3d(0, 10)
        ax.set_ylim3d(0, 10)

        savefig(direc + "/Coeff_" + titulo + '_eta=' + '%.2f'%(T/100.00) +".png");

        close()

        return coefficients, direc + "/Coeff_" + titulo + '_eta=' + '%.2f'%(T/100.00)
+".png"
```

```
In [ ]: # ------ ONE BEAM SPLITTER ------

        # DECISION = 0 -> Single photon  +  coherent state
        # DECISION = 1 -> Single photon  +  squeezed state
        # DECISION = 2 -> Coherent state +  squeezed state
        # DECISION = 3 -> Single photon
        # DECISION = 4 -> Squeezed state
        # DECISION = 5 -> Coherent state
        # DECISION = 6 -> Single photon  +  Single photon
        # DECISION = 7 -> Coherent state  +  Coherent state

        DECISION = 0

        a = destroy(N)

        x =       (a + a.dag())/sqrt(2)
        p = -1j * (a - a.dag())/sqrt(2)

        # Coherent amplitude
        alpha = 1

        # Squeezed parameter
        rr = 0.5 - 0.5j

        toco1 = []

        complex_format = "(% .2f%+.2fi)"
        Labels = []
        Labels = np.append(Labels, 'Photon + Coherent (alpha=' +
        complex_format%(float(real(alpha)), float(imag(alpha))) + ')')
        Labels = np.append(Labels, 'Photon + Squeezed (r=' + complex_format%(float(real(rr)),
        float(imag(rr))) + ')')
        Labels = np.append(Labels, 'Coherent (alpha='  + complex_format%(float(real(alpha)),
        float(imag(alpha))) + ') + ' + 'Squeezed (r=' + complex_format%(float(real(rr)),
        float(imag(rr))) + ')')
        Labels = np.append(Labels, 'Photon')
        Labels = np.append(Labels, 'Squeezed (r=' + complex_format%(float(real(rr)),
        float(imag(rr))) + ')')
        Labels = np.append(Labels, 'Coherent (alpha='  + complex_format%(float(real(alpha)),
        float(imag(alpha))) + ')')
        Labels = np.append(Labels, 'Photon + Photon')
        Labels = np.append(Labels, 'Coherent + Coherent (alpha=' +
        complex_format%(float(real(alpha)), float(imag(alpha))) + ')')

        # Create the folders where store the data

        folder = []
        folder = np.append(folder, './1BS/PhotonCoherent')
        folder = np.append(folder, './1BS/PhotonSqueezed')
        folder = np.append(folder, './1BS/CoherentSqueezed')
        folder = np.append(folder, './1BS/Photon')
folder = np.append(folder, './1BS/Squeezed')
folder = np.append(folder, './1BS/Coherent')
folder = np.append(folder, './1BS/PhotonPhoton')
folder = np.append(folder, './1BS/CoherentCoherent')

try:
    os.makedirs(folder[DECISION])
except OSError:
    if not os.path.isdir(folder[DECISION]):
        raise

print (Labels [DECISION])
toco1 = []
pictures1 = [];
pictures2 = [];
picturesD1 = [];

pic_coef = [];
pictures_quad1 = [];
pictures_quad2 = [];
pictures_quadD1 = [];

x2 = []
p2 = []
x2p2 = []

detec = tensor(fock_dm(N, 1), qeye(N))

for T in np.linspace(0, 100, 11):
    t = np.sqrt(T/100.0);
    r = - np.sqrt((100 - T)/100.0) * 1j;

    a1_dag = tensor(conj(t) * a.dag(), qeye(N)) + tensor(qeye(N), conj(r) * a.dag())
    a2_dag = tensor(r * a.dag(), qeye(N)) + tensor(qeye(N), t * a.dag())

#     Here you decide which states are incoming
    if DECISION == 0:
        psi_fin = Qobj(alpha * a2_dag - conj(alpha) * a2_dag.dag()).expm() * a1_dag *
(tensor(fock(N, 0), fock(N, 0)))
    else:
        if DECISION == 1:
            psi_fin = Qobj(0.5 * (conj(rr) * a1_dag.dag() * a1_dag.dag() - rr * a1_dag *
a1_dag)).expm() * a2_dag * (tensor(fock(N, 0), fock(N, 0)))
        else:
            if DECISION == 2:
                psi_fin = Qobj(alpha * a2_dag - conj(alpha) * a2_dag.dag()).expm() *
Qobj(0.5 * (conj(rr) * a1_dag.dag() * a1_dag.dag() - rr * a1_dag * a1_dag)).expm() *
(tensor(fock(N, 0), fock(N, 0)))
            else:
                if DECISION == 3:
                    psi_fin = a1_dag * (tensor(fock(N, 0), fock(N, 0)))
                else:
                    if DECISION == 4:
                        psi_fin = Qobj(0.5 * (conj(rr) * a1_dag.dag() * a1_dag.dag() -
rr * a1_dag * a1_dag)).expm() * (tensor(fock(N, 0), fock(N, 0)))
                    else:
                        if DECISION == 5:
                            psi_fin = Qobj(alpha * a2_dag - conj(alpha) *
a2_dag.dag()).expm() * (tensor(fock(N, 0), fock(N, 0)))
                        else:
                            if DECISION == 6:
                                psi_fin = a2_dag * a1_dag * (tensor(fock(N, 0), fock(N,
0)))
                            else:
                                if DECISION == 7:
                                    psi_fin = Qobj(alpha * a2_dag - conj(alpha) *
a2_dag.dag()).expm() * Qobj(alpha * a1_dag - conj(alpha) * a1_dag.dag()).expm() *
(tensor(fock(N, 0), fock(N, 0)))

    print("T = %.2f"%T)

#     Detector action
```

```
    psi_fin = detec * psi_fin

# #     Output port 1
#     aux, W = plot_wigner(psi_fin.ptrace(0), T, Labels[DECISION] + '_01',
folder[DECISION])
#     pictures1 = np.append(pictures1, aux)
    coef, pic = coefficients(N, psi_fin, T, Labels[DECISION], folder[DECISION]);
    pic_coef = np.append(pic_coef, pic)

# #     --------QUADRATURES--------

#     fig = plt.figure(figsize = (10, 10))
#     plot(np.linspace(-5, 5, 300), np.sum(W, axis=0))
#     plot(np.linspace(-5, 5, 300), np.sum(W, axis=1))
#     plot(np.linspace(-5, 5, 300), np.sum(wigner(fock(N, 0), np.linspace(-5, 5, 300),
np.linspace(-5, 5, 300)), axis=1))
#     plt.xlabel('x / y')
#     plt.ylim([0, 30])
#     plt.title(r'$\eta = %.2f$ - '%T + Labels[DECISION])
#     plt.legend(['X - Quadrature', 'Y - Quadrature', 'Vacuum'])
#     plt.savefig(folder[DECISION] + "/Quadrature_01_eta=%.2f"%T + ".png")
#     pictures_quad1 = np.append(pictures_quad1, folder[DECISION] +
"/Quadrature_01_eta=%.2f"%T + ".png")
#     close()

#     --------___________--------

#     Output port 2

    aux, W = plot_wigner(psi_fin.ptrace(1), T, Labels[DECISION] + '_02',
folder[DECISION])
    pictures2 = np.append(pictures2, aux)

#     --------QUADRATURES--------

    fig = plt.figure(figsize = (10, 10))
    plot(np.linspace(-5, 5, 300), np.sum(W, axis=0))
    plot(np.linspace(-5, 5, 300), np.sum(W, axis=1))
    plot(np.linspace(-5, 5, 300), np.sum(wigner(fock(N, 0), np.linspace(-5, 5, 300),
np.linspace(-5, 5, 300)), axis=1))
    plt.xlabel('x / y')
    plt.ylim([0, 30])
    plt.title(r'$\eta = %.2f$ - '%T + Labels[DECISION])
    plt.legend(['X - Quadrature', 'Y - Quadrature', 'Vacuum'])
    plt.savefig(folder[DECISION] + "/Quadrature_02_eta=%.2f"%T + ".png")
    pictures_quad2 = np.append(pictures_quad2, folder[DECISION] +
"/Quadrature_02_eta=%.2f"%T + ".png")
    close()

# #     --------___________--------

#     Log negativity
    rho_toco = partial_transpose(ket2dm(psi_fin), [1, 0])
    toco1 = np.append(toco1, np.log2((rho_toco.dag() * rho_toco).sqrtm().tr()))

    x2 = np.append(x2, variance(x, psi_fin.ptrace(1) / psi_fin.ptrace(1).norm()))
    p2 = np.append(p2, variance(p, psi_fin.ptrace(1) / psi_fin.ptrace(1).norm()))
    x2p2 = np.append(x2p2, variance(x, psi_fin.ptrace(1)) * variance(p,
psi_fin.ptrace(1)))

# #     Maiking .gif with all the obtained pictures
# with imageio.get_writer(folder[DECISION] + './N%d_'%N + Labels[DECISION].replace(" ",
"").replace("=(", "=").replace("(", "_").replace(")","") + '_01.gif', mode='I', duration
= .5) as writer:
#     for filename in pictures1:
#         image = imageio.imread(filename)
# #         os.remove(filename) #This works on Windows, idk if it works on linux
#         writer.append_data(image)

# with imageio.get_writer(folder[DECISION] + './N%d_'%N + Labels[DECISION].replace(" ",
"").replace("=(", "=").replace("(", "_").replace(")","") + '_quadra_01.gif', mode='I',
duration = .5) as writer:
```

```
#     for filename in pictures_quad1:
#         image = imageio.imread(filename)
# #         os.remove(filename) #This works on Windows, idk if it works on linux
#         writer.append_data(image)

with imageio.get_writer(folder[DECISION] + './N%d_'%N + Labels[DECISION].replace(" ",
"").replace("=(", "=").replace("(", "_").replace(")","") + '_02.gif', mode='I', duration
= .5) as writer:
    for filename in pictures2:
        image = imageio.imread(filename)
#         os.remove(filename) #This works on Windows, idk if it works on linux
        writer.append_data(image)

with imageio.get_writer(folder[DECISION] + './N%d_'%N + Labels[DECISION].replace(" ",
"").replace("=(", "=").replace("(", "_").replace(")","") + '_quadra_02.gif', mode='I',
duration = .5) as writer:
    for filename in pictures_quad2:
        image = imageio.imread(filename)
#         os.remove(filename) #This works on Windows, idk if it works on linux
        writer.append_data(image)

# with imageio.get_writer(folder[DECISION] + './N%d_'%N + Labels[DECISION].replace(" ",
"").replace("=(", "=").replace("(", "_").replace(")","") + '_coef.gif', mode='I',
duration = .5) as writer:
#     for filename in pic_coef:
#         image = imageio.imread(filename)
# #         os.remove(filename) #This works on Windows, idk if it works on linux
#         writer.append_data(image)

# Ploting the log-negativity

plot(np.arange(0, 1.1, 0.1), toco1)
plt.xlim(0, 1)
plt.ylim(0, 2.6)
plt.xlabel('$t^2$')
plt.ylabel('log negativity')

plt.grid()
plt.title(Labels[DECISION])
savefig(folder[DECISION] + '/N%d_'%N + Labels[DECISION].replace(" ", "").replace("=(",
"=").replace("(", "_").replace(")","") + "_entan_1BS.png");
close()

# Ploting the quadratures variance

fig = plt.figure(figsize = (10, 7))
plot(np.arange(0, 1.1, 0.1), x2)
plot(np.arange(0, 1.1, 0.1), p2)
plot(np.arange(0, 1.1, 0.1), np.full(x2.shape, variance(x, fock_dm(N, 0))))

plt.xlim(0, 1)
plt.ylim(0, 2)
plt.xlabel('$t^2$', fontsize = 30)
plt.ylabel('variance', fontsize = 30)
plt.xticks(fontsize = 20)
plt.yticks(fontsize = 20)

plt.legend(['$(\Delta X)^2$', '$(\Delta P)^2$', '$(\Delta P)^2$ vacuum'], fontsize = 20)

plt.grid()
plt.title(Labels[DECISION])
savefig(folder[DECISION] + '/N%d_'%N + Labels[DECISION].replace(" ", "").replace("=(",
"=").replace("(", "_").replace(")","") + "_variance_1BS.png");
close()

# Exporting the log-negativity data to a .txt

file = open(folder[DECISION] + '/N%d_'%N + 'LogNeg.txt', 'w')
file.write(str(toco1).replace('\n','').replace('[','').replace(']','').strip())
file.close

# Exporting the quadratures variance data to a .txt
```

```
        file = open(folder[DECISION] + '/N%d_'%N + 'x2.txt', 'w')
        file.write(str(x2).replace('\n','').replace('[','').replace(']','').strip())
        file.close()

        file = open(folder[DECISION] + '/N%d_'%N + 'p2.txt', 'w')
        file.write(str(p2).replace('\n','').replace('[','').replace(']','').strip())
        file.close()

        file = open(folder[DECISION] + '/N%d_'%N + 'x2p2.txt', 'w')
        file.write(str(x2p2).replace('\n','').replace('[','').replace(']','').strip())
        file.close()

        print ("END")

In [ ]: # ------ TWO BEAM SPLITTER ------

        # DECISION = 0 -> Single photon  +  coherent state
        # DECISION = 1 -> Single photon  +  squeezed state
        # DECISION = 2 -> Coherent state +  squeezed state
        # DECISION = 3 -> Single photon
        # DECISION = 4 -> Squeezed state
        # DECISION = 5 -> Coherent state
        # DECISION = 6 -> Single photon  +  Single photon
        # DECISION = 7 -> Coherent state  +  Coherent state

        DECISION = 1
        PHASE = 4 * np.pi / 4;
        phase_txt = 'PHASE=' + '%.2f'%(round(PHASE / np.pi, 2)) + 'pi - '

        a = destroy(N)

        x =       (a + a.dag())/sqrt(2)
        p = -1j * (a - a.dag())/sqrt(2)

        # Coherent parameter
        alpha = 1

        # Squeezed parameter
        rr = 0.5 - 0.5j

        toco1 = []

        complex_format = "(% .2f%+.2fi)"
        Labels = []
        Labels = np.append(Labels, phase_txt + 'Photon + Coherent (alpha=' +
        complex_format%(float(real(alpha)), float(imag(alpha))) + ')')
        Labels = np.append(Labels, phase_txt + 'Photon + Squeezed (r=' +
        complex_format%(float(real(rr)), float(imag(rr))) + ')')
        Labels = np.append(Labels, phase_txt + 'Coherent (alpha='  +
        complex_format%(float(real(alpha)), float(imag(alpha))) + ') + ' + 'Squeezed (r=' +
        complex_format%(float(real(rr)), float(imag(rr))) + ')')
        Labels = np.append(Labels, phase_txt + 'Photon')
        Labels = np.append(Labels, phase_txt + 'Squeezed (r=' + complex_format%(float(real(rr)),
        float(imag(rr))) + ')')
        Labels = np.append(Labels, phase_txt + 'Coherent (alpha='  +
        complex_format%(float(real(alpha)), float(imag(alpha))) + ')')
        Labels = np.append(Labels, phase_txt + 'Photon + Photon')
        Labels = np.append(Labels, phase_txt + 'Coherent + Coherent (alpha=' +
        complex_format%(float(real(alpha)), float(imag(alpha))) + ')')

        # Create the folders where store the data

        folder = []
        folder = np.append(folder, './2BS/PhotonCoherent')
        folder = np.append(folder, './2BS/PhotonSqueezed')
        folder = np.append(folder, './2BS/CoherentSqueezed')
        folder = np.append(folder, './2BS/Photon')
        folder = np.append(folder, './2BS/Squeezed')
        folder = np.append(folder, './2BS/Coherent')
        folder = np.append(folder, './2BS/PhotonPhoton')
        folder = np.append(folder, './2BS/CoherentCoherent')
```

```
try:
    os.makedirs(folder[DECISION])
except OSError:
    if not os.path.isdir(folder[DECISION]):
        raise

print (Labels [DECISION])
toco1 = []
pictures1 = [];
pictures2 = [];
picturesD1 = [];
picturesD2 = [];
pic_coef = [];
pictures_quad1 = [];
pictures_quad2 = [];
pictures_quadD1 = [];
pictures_quadD2 = [];

x1 = []
p1 = []
x2 = []
p2 = []

detec1 = tensor(qeye(N), qeye(N), fock_dm(N, 1), fock_dm(N, 1))

for T in np.linspace(0, 100, 11):
    t1 = np.sqrt(T/100.0) * 1;
    r1 = np.sqrt((100 - T)/100.0) * 1j;
    td1 = 1.0 / np.sqrt(2);
    rd1 = np.sqrt((1.0 - td1 * td1)) * 1j;
    td2 = 1.0 / np.sqrt(2);
    rd2 = np.sqrt((1.0 - td2 * td2)) * 1j;
    t2 = t1;
    r2 = r1;

    a6_dag = tensor(t2 * a.dag(), qeye(N), qeye(N), qeye(N)) + tensor(qeye(N), r2 *
a.dag(), qeye(N), qeye(N))
    a8_dag = tensor(r2 * a.dag(), qeye(N), qeye(N), qeye(N)) + tensor(qeye(N), t2 *
a.dag(), qeye(N), qeye(N))

    a4_dag = td1 * a6_dag + rd1 * tensor(qeye(N), qeye(N), a.dag(), qeye(N))
    a3_dag = td2 * a8_dag * np.exp(PHASE * 1j) + rd2 * tensor(qeye(N), qeye(N), qeye(N),
a.dag())

    a1_dag = t1 * a3_dag + r1 * a4_dag
    a2_dag = r1 * a3_dag + t1 * a4_dag

#     Here you decide which states are incoming
    if DECISION == 0:
        psi_fin = Qobj(alpha * a2_dag - conj(alpha) * a2_dag.dag()).expm() * a1_dag *
(tensor(fock(N, 0), fock(N, 0), fock(N, 0), fock(N, 0)))
    else:
        if DECISION == 1:
            psi_fin = Qobj(0.5 * (conj(rr) * a1_dag.dag() * a1_dag.dag() - rr * a1_dag *
a1_dag)).expm() * a2_dag * (tensor(fock(N, 0), fock(N, 0), fock(N, 0), fock(N, 0)))
        else:
            if DECISION == 2:
                psi_fin = Qobj(alpha * a2_dag - conj(alpha) * a2_dag.dag()).expm() *
Qobj(0.5 * (conj(rr) * a1_dag.dag() * a1_dag.dag() - rr * a1_dag * a1_dag)).expm() *
(tensor(fock(N, 0), fock(N, 0), fock(N, 0), fock(N, 0)))
            else:
                if DECISION == 3:
                    psi_fin = a1_dag * (tensor(fock(N, 0), fock(N, 0), fock(N, 0),
fock(N, 0)))
                else:
                    if DECISION == 4:
                        psi_fin = Qobj(0.5 * (conj(rr) * a1_dag.dag() * a1_dag.dag() -
rr * a1_dag * a1_dag)).expm() * (tensor(fock(N, 0), fock(N, 0), fock(N, 0), fock(N, 0)))
                    else:
                        if DECISION == 5:
                            psi_fin = Qobj(alpha * a2_dag - conj(alpha) *
a2_dag.dag()).expm() * (tensor(fock(N, 0), fock(N, 0), fock(N, 0), fock(N, 0)))
```

```
                        else:
                            if DECISION == 6:
                                psi_fin = a2_dag * a1_dag * (tensor(fock(N, 0), fock(N,
0), fock(N, 0), fock(N, 0)))
                            else:
                                if DECISION == 7:
                                    psi_fin = Qobj(alpha * a2_dag - conj(alpha) *
a2_dag.dag()).expm() * Qobj(alpha * a1_dag - conj(alpha) * a1_dag.dag()).expm() *
(tensor(fock(N, 0), fock(N, 0), fock(N, 0), fock(N, 0)))

    print("T = %.2f"%T)

    psi_fin = detec1 * psi_fin

#     --------Output1--------

    aux, W = plot_wigner(psi_fin.ptrace(0), T, Labels[DECISION] + '_01',
folder[DECISION])
    pictures1 = np.append(pictures1, aux)
    coef, pic = coefficients(N, psi_fin.ptrace([0, 1]), T, Labels[DECISION],
folder[DECISION]);
    pic_coef = np.append(pic_coef, pic)

# #     --------QUADRATURES--------

#     fig = plt.figure(figsize = (10, 10))
#     plot(np.linspace(-5, 5, 300), np.sum(W, axis=0))
#     plot(np.linspace(-5, 5, 300), np.sum(W, axis=1))
#     plot(np.linspace(-5, 5, 300), np.sum(wigner(fock(N, 0), np.linspace(-5, 5, 300),
np.linspace(-5, 5, 300)), axis=1))
#     plt.xlabel('x / y')
#     plt.ylim([0, 30])
#     plt.title(r'$\eta = %.2f$ - Out1 -'%T + Labels[DECISION])
#     plt.legend(['X - Quadrature', 'Y - Quadrature', 'Vacuum'])
#     plt.savefig(folder[DECISION] + "/" + phase_txt + "Quadrature_01_eta=%.2f"%T +
".png")
#     pictures_quad1 = np.append(pictures_quad1, folder[DECISION] + "/" + phase_txt +
"Quadrature_01_eta=%.2f"%T + ".png")
#     close()

# #     --------___________--------

#     --------Output2--------

    aux, W = plot_wigner(psi_fin.ptrace(1), T, Labels[DECISION] + '_02',
folder[DECISION])
    pictures2 = np.append(pictures2, aux)

# #     --------QUADRATURES--------

#     fig = plt.figure(figsize = (10, 10))
#     plot(np.linspace(-5, 5, 300), np.sum(W, axis=0))
#     plot(np.linspace(-5, 5, 300), np.sum(W, axis=1))
#     plot(np.linspace(-5, 5, 300), np.sum(wigner(fock(N, 0), np.linspace(-5, 5, 300),
np.linspace(-5, 5, 300)), axis=1))
#     plt.xlabel('x / y')
#     plt.ylim([0, 30])
#     plt.title(r'$\eta = %.2f$ - Out2 -'%T + Labels[DECISION])
#     plt.legend(['X - Quadrature', 'Y - Quadrature', 'Vacuum'])
#     plt.savefig(folder[DECISION] + "/" + phase_txt + "Quadrature_02_eta=%.2f"%T +
".png")
#     pictures_quad2 = np.append(pictures_quad2, folder[DECISION] + "/" + phase_txt +
"Quadrature_02_eta=%.2f"%T + ".png")
#     close()

# #     --------___________--------



# #     --------___________--------

#     --------ENTANGLEMENT--------
```

```
    rho_toco = partial_transpose(psi_fin.ptrace([0, 1]) / Qobj(psi_fin.ptrace([0,
1])).norm(), [1, 0])
    toco1 = np.append(toco1, np.log2((rho_toco.dag() * rho_toco).sqrtm().tr()))

#     --------___________--------

    x1 = np.append(x1, variance(x, psi_fin.ptrace(0) / psi_fin.ptrace(0).norm()))
    p1 = np.append(p1, variance(p, psi_fin.ptrace(0) / psi_fin.ptrace(0).norm()))
    x2 = np.append(x2, variance(x, psi_fin.ptrace(1) / psi_fin.ptrace(1).norm()))
    p2 = np.append(p2, variance(p, psi_fin.ptrace(1) / psi_fin.ptrace(1).norm()))

with imageio.get_writer(folder[DECISION] + '/N%d_'%N + Labels[DECISION].replace(" ",
"").replace("=(", "=").replace("(", "_").replace(")","") + '_out1_2BS.gif', mode='I',
duration = .5) as writer:
    for filename in pictures1:
        image = imageio.imread(filename)
#         os.remove(filename) #This works on Windows, idk if it works on linux
        writer.append_data(image)

with imageio.get_writer(folder[DECISION] + '/N%d_'%N + Labels[DECISION].replace(" ",
"").replace("=(", "=").replace("(", "_").replace(")","") + '_out2_2BS.gif', mode='I',
duration = .5) as writer:
    for filename in pictures2:
        image = imageio.imread(filename)
#         os.remove(filename) #This works on Windows, idk if it works on linux
        writer.append_data(image)

with imageio.get_writer(folder[DECISION] + '/N%d_'%N + Labels[DECISION].replace(" ",
"").replace("=(", "=").replace("(", "_").replace(")","") + '_quadra_out1_2BS.gif',
mode='I', duration = .5) as writer:
    for filename in pictures_quad1:
        image = imageio.imread(filename)
#         os.remove(filename) #This works on Windows, idk if it works on linux
        writer.append_data(image)

with imageio.get_writer(folder[DECISION] + '/N%d_'%N + Labels[DECISION].replace(" ",
"").replace("=(", "=").replace("(", "_").replace(")","") + '_quadra_out2_2BS.gif',
mode='I', duration = .5) as writer:
    for filename in pictures_quad2:
        image = imageio.imread(filename)
#         os.remove(filename) #This works on Windows, idk if it works on linux
        writer.append_data(image)

# with imageio.get_writer(folder[DECISION] + './N%d_'%N + Labels[DECISION].replace(" ",
"").replace("=(", "=").replace("(", "_").replace(")","") + '_coef_2BS.gif', mode='I',
duration = .5) as writer:
#     for filename in pic_coef:
#         image = imageio.imread(filename)
# #         os.remove(filename) #This works on Windows, idk if it works on linux
#         writer.append_data(image)

# A rough plot of log-negativity

plot(toco1)
plt.title(phase_txt + Labels[DECISION])
plt.xlabel(r'$\eta$/10')
plt.ylabel('log negativity')
savefig(folder[DECISION] + '/N%d_'%N + Labels[DECISION].replace(" ", "").replace("=(",
"=").replace("(", "_").replace(")","") + "_entan_2BS.png");
close()



# Ploting the quadratures variance

fig = plt.figure(figsize = (10, 7))
plot(np.arange(0, 1.1, 0.1), x1)
plot(np.arange(0, 1.1, 0.1), p1)
plot(np.arange(0, 1.1, 0.1), np.full(x2.shape, variance(x, fock_dm(N, 0))))

plt.xlim(0, 1)
```

```
        plt.ylim(0, 2)
        plt.xlabel('$t^2$', fontsize = 30)
        plt.ylabel('variance', fontsize = 30)
        plt.xticks(fontsize = 20)
        plt.yticks(fontsize = 20)

        plt.legend(['$(\Delta X)^2$', '$(\Delta P)^2$', '$(\Delta X)^2$ vacuum'], fontsize = 20)

        plt.grid()
        plt.title(Labels[DECISION])
        savefig(folder[DECISION] + '/N%d_'%N + Labels[DECISION].replace(" ", "").replace("=(",
        "=").replace("(", "_").replace(")","") + "_out1_variance_2BS.png");
        close()

        fig = plt.figure(figsize = (10, 7))
        plot(np.arange(0, 1.1, 0.1), x2)
        plot(np.arange(0, 1.1, 0.1), p2)
        plot(np.arange(0, 1.1, 0.1), np.full(x2.shape, variance(x, fock_dm(N, 0)) * variance(p,
        fock_dm(N, 0))))

        plt.xlim(0, 1)
        plt.ylim(0, 2)
        plt.xlabel('$t^2$', fontsize = 30)
        plt.ylabel('variance', fontsize = 30)
        plt.xticks(fontsize = 20)
        plt.yticks(fontsize = 20)

        plt.legend(['$(\Delta X)^2$', '$(\Delta P)^2$', '$(\Delta X)^2$ vacuum'], fontsize = 20)

        plt.grid()
        plt.title(Labels[DECISION])
        savefig(folder[DECISION] + '/N%d_'%N + Labels[DECISION].replace(" ", "").replace("=(",
        "=").replace("(", "_").replace(")","") + "_out2_variance_2BS.png");
        close()

        # Exporting the log-negativity data to a .txt

        file = open(folder[DECISION] + '/N%d_'%N + phase_txt + 'LogNeg.txt', 'w')
        file.write(str(toco1).replace('\n','').replace('[','').replace(']','').strip())
        file.close()

        # Exporting the quadratures variance data to a .txt

        file = open(folder[DECISION] + '/N%d_'%N + 'x1.txt', 'w')
        file.write(str(x1).replace('\n','').replace('[','').replace(']','').strip())
        file.close()

        file = open(folder[DECISION] + '/N%d_'%N + 'p1.txt', 'w')
        file.write(str(p1).replace('\n','').replace('[','').replace(']','').strip())
        file.close()

        file = open(folder[DECISION] + '/N%d_'%N + 'x2.txt', 'w')
        file.write(str(x2).replace('\n','').replace('[','').replace(']','').strip())
        file.close()

        file = open(folder[DECISION] + '/N%d_'%N + 'p2.txt', 'w')
        file.write(str(p2).replace('\n','').replace('[','').replace(']','').strip())
        file.close()

        print ("END")

In [ ]: # To make a comparative log-negativity plot including variation on t and phase

        DECISION = 0

        file = open(folder[DECISION] + '/N7_PHASE=000pi-LogNeg.txt', 'r')
        a = file.readline()
        file.close()

        # file = open(folder[DECISION] + '/N7_PHASE=025pi-LogNeg.txt', 'r')
        # b = file.readline()
        # file.close()
```

```
file = open(folder[DECISION] + '/N7_PHASE=050pi-LogNeg.txt', 'r')
c= file.readline()
file.close()

# file = open(folder[DECISION] + '/N7_PHASE=075pi-LogNeg.txt', 'r')
# d = file.readline()
# file.close()

file = open(folder[DECISION] + '/N7_PHASE=100pi-LogNeg.txt', 'r')
e = file.readline()
file.close()

a = a.split()
# b = b.split()
c = c.split()
# d = d.split()
e = e.split()

toco_a = []
toco_c = []
toco_e = []

for kk in np.arange(0, len(a), 2):
    toco_a = np.append(toco_a, float(a[kk]))
#     b[kk] = float(b[kk])
    toco_c = np.append(toco_c, float(c[kk]))
#     d[kk] = float(d[kk])
    toco_e = np.append(toco_e, float(e[kk]))

fig = plt.figure(figsize = (10, 7))

plot(np.arange(0, 1.1, 0.1), toco_a)
# plot(np.arange(0, 1.1, 0.1), b)
plot(np.arange(0, 1.1, 0.1), toco_c)
# plot(np.arange(0, 1.1, 0.1), d)
plot(np.arange(0, 1.1, 0.1), toco_e)

plt.xlim(0,1)
plt.ylim(0,1)
plt.xlabel('$t^2$', fontsize = 25)
plt.ylabel('log negativity', fontsize = 25)
plt.xticks(fontsize = 30)
plt.yticks(fontsize = 25)

plt.legend(['PHASE = 0.00$\pi$',
#              'PHASE = 0.25$\pi$',
            'PHASE = 0.50$\pi$',
#              'PHASE = 0.75$\pi$',
            'PHASE = 1.00$\pi$'], loc = 0, bbox_to_anchor=(0.5, 0.5), fontsize = 20)
plt.grid()
plt.title(Labels[DECISION][15:])
savefig('C:/Users/Jaime/Documents/Universidad/Curso_2017-2018/TFG/BeamSplitter/2BS/' +
'N%d_'%N + Labels[DECISION][15:].replace(" ", "").replace("=(", "=").replace("(",
"_").replace(")","").replace('.','') + "_entan_2BS.png");
plt.show()
close()
print('toco')
```

```
In [ ]: # To make a comparative log-negativity plot including variation on t and phase
        DECISION = 3
        file = open('C:/Users/Jaime/Documents/Universidad/Curso_2017-2018/TFG/BeamSplitter/1BS/P
        hoton/N7_LogNeg.txt', 'r')
        a = file.readline()
        file.close()


        a = a.split()
        b = []

        for kk in np.arange(0, len(a)):
            b = np.append(b, float(a[kk]))
        #     print('%.2f %.2f %d'%(b[int(kk/2)], float(a[kk]), int(kk/2)))
        fig = plt.figure(figsize = (10, 7))

        plot(np.arange(0, 1.1, 0.1), b)

        plt.xlim(0,1)
        plt.ylim(0,2.6)
        plt.xlabel('$t^2$', fontsize = 25)
        plt.ylabel('log negativity', fontsize = 30)
        plt.xticks(fontsize = 30)
        plt.yticks(fontsize = 30)

        plt.grid()
        plt.title(Labels[DECISION][15:])
        savefig('C:/Users/Jaime/Documents/Universidad/Curso_2017-2018/TFG/BeamSplitter/1BS/' +
        '/N%d_'%N + Labels[DECISION][15:].replace(" ", "").replace("=(", "=").replace("(",
        "_").replace(")","").replace('.','') + "_entan_1BS.png");
        plt.show()
        close()
        print('toco')

In [ ]: # Version of the used software

        from qutip.ipynbtools import version_table

        version_table()
```

Bibliography

[1] Nobel Media AB 2014. *Roy J. Glauber - Facts*. 2018. URL: http://www.nobelprize.org/nobel_prizes/physics/laureates/2005/glauber-facts.html (visited on 07/03/2018).

[2] G.S. Agarwal. "Quantum Optics". In: Quantum Optics. Cambridge University Press, 2013. Chap. 3, p. 58. ISBN: 9781107006409. URL: https://books.google.de/books?id=7KKw_XIYaioC.

[3] K. Audenaert, M. B. Plenio, and J. Eisert. "Entanglement Cost under Positive-Partial-Transpose-Preserving Operations". In: *Phys. Rev. Lett.* 90 (2 2003), p. 027901. DOI: 10.1103/PhysRevLett.90.027901. URL: https://link.aps.org/doi/10.1103/PhysRevLett.90.027901.

[4] Konrad Banaszek, Rafal Demkowicz-Dobrzański, and Ian A. Walmsley. "Quantum states made to measure". In: *Nature Photonics* 3 (Dec. 2009), p. 673. DOI: 10.1038/nphoton.2009.223. URL: http://dx.doi.org/10.1038/nphoton.2009.223.

[5] Tim J. Bartley et al. "Multi-photon state engineering by heralded interference between single photons and coherent states". In: *Phys. Rev.* 86 (043820 2012). DOI: 10.1103/PhysRevA.86.043820. URL: https://arxiv.org/abs/1205.0497.

[6] Jon Brogaard. "Wigner function formalism in Quantum mechanics". Bachelor's project in Physics. MA thesis. Niels Bohr Institute University of Copenhagen, June 2015. URL: https://cmt.nbi.ku.dk/student_projects/bachelor_theses/Jon_Brogaard_Bachelorthesis_2015.pdf.

[7] Naresh Chandra and Hari Prakash. "Anticorrelation in Two-Photon Attenuated Laser Beam". In: *Phys. Rev. A* 1 (6 1970), pp. 1696–1698. DOI: 10.1103/PhysRevA.1.1696. URL: https://link.aps.org/doi/10.1103/PhysRevA.1.1696.

[8] Darrick E. Chang, Vladan Vuletić, and Mikhail D. Lukin. "Quantum nonlinear optics — photon by photon". In: *Nature Photonics* 8 (043820 Aug. 2014), p. 685. DOI: 10.1038/nphoton.2014.192. URL: http://dx.doi.org/10.1038/nphoton.2014.192.

[9] Peter L.Knight Christopher C.Gerry. "Beam splitters and interferometers". In: *Introductory Quantum Optics*. Cambridge University Press, 2005. Chap. 6, pp. 135–149.

[10] Peter L.Knight Christopher C.Gerry. "Beam splitters and interferometers". In: *Introductory Quantum Optics*. Cambridge University Press, 2005. Chap. 3, pp. 43–73.

[11] Peter L.Knight Christopher C.Gerry. "Beam splitters and interferometers". In: *Introductory Quantum Optics*. Cambridge University Press, 2005. Chap. 7, pp. 150–190.

[12] Cython. *C-Extensions for Python*. 2018. URL: http://cython.org/ (visited on 06/24/2018).

[13] A. Furusawa et al. “Bell Inequality Violation with Two Remote Atomic Qubits”. In: *Science* 282 (5389 1998), pp. 706–709. DOI: 10.1126/science.282.5389.706. URL: http://science.sciencemag.org/content/282/5389/706/tab-article-info.

[14] Ryszard Horodecki et al. “Quantum entanglement”. In: *Rev.Mod.Phys.* 81 (2007), pp. 865–942. URL: https://arxiv.org/abs/quant-ph/0702225v2.

[15] J. D. Hunter. “Matplotlib: A 2D graphics environment”. In: *Computing In Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[16] IPython. *Interactive Computing*. 2018. URL: https://ipython.org/ (visited on 06/24/2018).

[17] J.R. Johansson, P.D. Nation, and Franco Nori. “QuTiP 2: A Python framework for the dynamics of open quantum systems”. In: *Computer Physics Communications* 184.4 (2013), pp. 1234 –1240. ISSN: 0010-4655. DOI: https://doi.org/10.1016/j.cpc.2012.11.019. URL: http://www.sciencedirect.com/science/article/pii/S0010465512003955.

[18] J.R. Johansson, P.D. Nation, and Franco Nori. “QuTiP: An open-source Python framework for the dynamics of open quantum systems”. In: *Computer Physics Communications* 183.8 (2012), pp. 1760 –1772. ISSN: 0010-4655. DOI: https://doi.org/10.1016/j.cpc.2012.02.021. URL: http://www.sciencedirect.com/science/article/pii/S0010465512000835.

[19] Eric Jones, Travis Oliphant, Pearu Peterson, et al. *SciPy: Open source scientific tools for Python*. [Online; accessed 2018-06-24]. 2001–. URL: http://www.scipy.org/.

[20] Jupyter. *Project Jupyter*. 2018. URL: http://jupyter.org (visited on 06/24/2018).

[21] Raymond K. et al. “The Rac GTPase-activating protein RotundRacGAP interferes with Drac1 and Dcdc42 signalling in Drosophila melanogaster”. In: *The Journal of Biological Chemistry* (July 2001). DOI: 10.1074/jbc.M105779200. URL: http://www.jbc.org/content/276/38/35909.

[22] Carl A. Kocher and Eugene D. Commins. “Polarization Correlation of Photons Emitted in an Atomic Cascade”. In: *Phys. Rev. Lett.* 18 (15 1967), pp. 575–577. DOI: 10.1103/PhysRevLett.18.575. URL: https://link.aps.org/doi/10.1103/PhysRevLett.18.575.

[23] N. Lambert, C. Emary, and T. Brandes. “Entanglement and entropy in a spin-boson quantum phase transition”. In: *Phys. Rev. A* 71 (5 2005), p. 053804. DOI: 10.1103/PhysRevA.71.053804. URL: https://link.aps.org/doi/10.1103/PhysRevA.71.053804.

[24] Andrew C. Doherty Lluis Masanes Yeong-Cherng Liang. “All bipartite entangled states display some hidden nonlocality”. In: *Physical Review Letters* 100 (Mar. 2007). URL: https://arxiv.org/abs/quant-ph/0703268v1.

[25] R Loudon and P Knight. “Squeezed Light”. In: 34 (June 1987), pp. 709–759.

[26] S. Virmani Martin B. Plenio. “An introduction to entanglement measures”. In: *Quant.Inf.Comput* 7 (2007), pp. 1–51. URL: https://arxiv.org/abs/quant-ph/0504163v3.

[27] Matplotlib. *Matplotlib*. 2018. URL: https://matplotlib.org/ (visited on 06/24/2018).

[28] D. N. Matsukevich et al. "Bell Inequality Violation with Two Remote Atomic Qubits". In: *Phys. Rev. Lett.* 100 (15 2008), p. 150404. DOI: 10.1103/PhysRevLett.100.150404. URL: https://link.aps.org/doi/10.1103/PhysRevLett.100.150404.

[29] J. C. F. Matthews et al. "Journal". In: *Nature Photon* 3 (2009), pp. 346–350.

[30] D. M. Meekhof et al. "Generation of Nonclassical Motional States of a Trapped Atom". In: *Phys. Rev. Lett.* 76 (11 1996), pp. 1796–1799. DOI: 10.1103/PhysRevLett.76.1796. URL: https://link.aps.org/doi/10.1103/PhysRevLett.76.1796.

[31] Evan Meyer-Scott et al. "Discorrelated quantum states". In: *Scientific Reports* 7 (Feb. 2017). DOI: 10.1038/srep41622. URL: https://arxiv.org/abs/1606.04369v2.

[32] Christine A. Muschik et al. "Efficient quantum memory and entanglement between light and an atomic ensemble using magnetic fields". In: *Phys. Rev. A* 73 (6 2006), p. 062329. DOI: 10.1103/PhysRevA.73.062329. URL: https://link.aps.org/doi/10.1103/PhysRevA.73.062329.

[33] Nature. *Quantum Optics*. 2018. URL: https://www.nature.com/subjects/quantum-optics (visited on 06/26/2018).

[34] NumPy. *NumPy*. 2018. URL: http://www.numpy.org/ (visited on 06/24/2018).

[35] Z. Y. Ou and L. Mandel. "Violation of Bell's Inequality and Classical Probability in a Two-Photon Correlation Experiment". In: *Phys. Rev. Lett.* 61 (1 1988), pp. 50–53. DOI: 10.1103/PhysRevLett.61.50. URL: https://link.aps.org/doi/10.1103/PhysRevLett.61.50.

[36] H. Paul. "Photon antibunching". In: *Rev. Mod. Phys.* 54 (4 1982), pp. 1061–1102. DOI: 10.1103/RevModPhys.54.1061. URL: https://link.aps.org/doi/10.1103/RevModPhys.54.1061.

[37] Fernando Pérez and Brian E. Granger. "IPython: a System for Interactive Scientific Computing". In: *Computing in Science and Engineering* 9.3 (May 2007), pp. 21–29. ISSN: 1521-9615. DOI: 10.1109/MCSE.2007.53. URL: http://ipython.org.

[38] M.B. Plenio. "The logarithmic negativity: A full entanglement monotone that is not convex". In: *Phys. Rev. Lett* 95 (Feb. 2006), p. 090503. DOI: 10.1103/PhysRevLett.95.090503. URL: https://arxiv.org/abs/quant-ph/0505071.

[39] E Polzik et al. "Entanglement and quantum teleportation with multi-atom ensembles". In: 361 (Aug. 2003), pp. 1391–9. URL: https://www.researchgate.net/publication/10655574_Entanglement_and_quantum_teleportation_with_multi-atom_ensembles.

[40] Quantiki. *Positive partial transpose*. 2015. URL: https://www.quantiki.org/wiki/positive-partial-transpose (visited on 06/24/2018).

[41] "Quantum evolution". In: *Nature Photonics* 3 (Dec. 2009), p. 669. DOI: 10.1038/nphoton.2009.219. URL: http://dx.doi.org/10.1038/nphoton.2009.219.

[42] University of Calgary. Faculty of Science. Institute for Quantum Science and Technology. *A galery of Wigner functions*. URL: https://www.iqst.ca/quantech/wiggalery.php (visited on 07/04/2018).

[43] QuTiP. *Quantum Toolbox in Python*. 2013. URL: http://qutip.org/ (visited on 06/24/2018).

[44] Timothy C. Ralph and Ping K. Lam. "A bright future for quantum communications". In: *Nature Photonics* 3 (Dec. 2009), p. 671. DOI: 10.1038/nphoton.2009.222. URL: http://dx.doi.org/10.1038/nphoton.2009.222.

[45] J. Ries, B. Brezger, and A. I. Lvovsky. "Experimental Vacuum Squeezing in Rubidium Vapor via Self-Rotation". In: *Physical Review A* 68.2 (2003), 025801 (4). DOI: 10.1103/PhysRevA.68.025801. URL: http://www.iqst.ca/quantech/pubs/2003/sq-pra.pdf.

[46] SciPy. *SciPy*. 2018. URL: https://www.scipy.org/ (visited on 06/24/2018).

[47] W. Tittel et al. "Violation of Bell Inequalities by Photons More Than 10 km Apart". In: *Phys. Rev. Lett.* 81 (17 1998), pp. 3563–3566. DOI: 10.1103/PhysRevLett.81.3563. URL: https://link.aps.org/doi/10.1103/PhysRevLett.81.3563.

[48] G. Vidal and R. F. Werner. "Computable measure of entanglement". In: *Phys. Rev. A* 65 (3 2002), p. 032314. DOI: 10.1103/PhysRevA.65.032314. URL: https://link.aps.org/doi/10.1103/PhysRevA.65.032314.

[49] A. Windhager et al. "Quantum Interference between a Single-Photon Fock State and a Coherent State". In: *Optics Communications* 284 (Sept. 2011), pp. 1907–1912. URL: https://arxiv.org/abs/1009.1844.