Politecnico di Torino

Université de Paris

Master's Degree in Nanotechnologies and Quantum Devices

Final Internship Report

Feasibility Study of a Quantum Repeater Link on the Jussieu Campus

Supervisors:
Prof. M.L. Della Rocca
Prof. C. Ricciardi

Author:
Paolo Fittipaldi

Academic Year 2020/2021

Contents

Preliminary note: *The error in the frequency estimation that was present in the former draft was fixed in the present iteration of this work. All reported results come directly from the simulator. The new implementation of the frequency calculation is reflected in the definitive version of the code and was presented during the internship discussion on 30/06/2021, but is not reported in the appendices of the present document.*

Chapter 1

Introduction

1.1 Preliminary Context

Since the dawn of times, humans have needed and wanted to communicate to thrive and move forward as a species. Rising beyond the well-established paradigms of Classical Information, the last few decades have witnessed the rise of Quantum Communication, a new branch of science that seeks to harness the potential of Quantum Mechanics in the realization of novel protocols to be employed alongside classical ones, or in some cases even replace them.

A preliminary idea of Quantum Communication's potential can be obtained by translating two elementary principles of quantum mechanics in the language of information, namely the No Cloning Theorem and the Measurement Postulate: given any quantum system, it is impossible to build a perfect replica of its state. Moreover, the mere action of measurement irreversibly perturbs said state. If the quantum state of the system mentioned in these general statements encodes a message, it is impossible for a malevolent eavesdropper to forge a perfect copy of the message, or even read the existing copy without changing its information, rendering it useless for both themselves and the intended recipient, and signaling the presence of an intruder to the original communicating parties.

The theoretical key to understanding and implementing Quantum Communication schemes is entanglement, a purely quantum phenomenon under which the state of a system cannot be fully factorized in terms of the states of its components: observing one of the subsystems simultaneously affects the other(s) and immediately breaks entanglement.

Finding a way to exploit such a property is what sparked interest in some of the main areas of Quantum Communication, such as distributed quantum computing or quantum key distribution.

It is therefore of paramount importance to find a manner to distribute entangled states in a remote and distributed fashion: the system of choice for sending quantum information over long distances is almost always the single photon, sent through free space or standard optical fiber. Despite them being the best system to carry quantum information, single photon signals introduce a severe problem of attenuation. This issue is easily solved in classical systems by amplifying the signal several times along the distance it needs to cross, but the same solution is not viable in quantum implementations due to the previously mentioned No Cloning Theorem: since it is impossible to create a perfect copy of an information-carrying photon, it is unfeasible to amplify a quantum signal in the usual sense as well. The quantum solution to this challenge employs the technique of entanglement swapping: if there are two entangled pairs AB and CD, a particular measurement on B and C can project A and D in an entangled state. Iterating, one can split the distance between the two end nodes an arbitrary number of times. However, the success probability is still exponential: entanglement generation is probabilistic, and with this setup 2N links must be established successfully and simultaneously. The natural follow-up is a device that not only can halve the distance over which entanglement must be established, but also store successful attempts while waiting for the other elementary links to be ready. Once this condition is verified, swapping is repeatedly performed until the two end nodes are entangled. It is therefore to be expected that one of the main goals of current Quantum Communication is to provide an experimental implementation of such a device, known as a quantum repeater.

1.2 The Laboratory and the Group

The internship was carried out in the Quantum Information group at Laboratoire d'Informatique de Paris 6 (LIP6), with the additional support of researchers from Laboratoire Kastler Brossel (LKB), both located on the Jussieu campus of Sorbonne University. LIP6 is primarily a Computer Science lab, with its 21 teams spread over four main research axes spanning from information theory to systems architecture, passing through domains such as Data Science and Cybersecurity. The Quantum Information team works both on the theoretical and experimental side by striving to gain further understanding of quantum mechanics as seen through information theory and to exploit such understanding in applications such as blind computing, CVQKD and quantum money.

LKB is on the other hand a "pure" atomic physics/optics laboratory. Despite the internship not being carried out there, several researchers provided their support and guidance throughout the project and developed the memory technology that is studied in the present document.

1.3 Presentation of the Project and Scientific Context

Quantum Memories are devices able to store and subsequently re-emit photons in a given quantum state. Nowadays, quantum memories (and consequently repeaters) are a thriving field, with research focusing primarily on the implementation of communication protocols [8] and networks [11] since there exist some well-established techniques to implement the memories themselves. Two of the most frequently employed among such techniques are described in the following section and then compared in table 1.1. A brief overview of some more advanced protocols and additional theoretical discussions not strictly related to memory protocols complete the section.

1.4 Survey of memory protocols

1.4.1 Electromagnetically Induced Transparency (EIT)

To describe the principle of EIT[6], it is useful to think of an ensemble of Λ-level systems, i.e. triplets of states $|g\rangle$, $|s\rangle$ and $|e\rangle$ (respectively called "ground", "storage" and "excited") with dipole coupling along the $|g\rangle \leftrightarrow |e\rangle$ and $|s\rangle \leftrightarrow |e\rangle$ transitions, but a forbidden $|g\rangle \leftrightarrow |s\rangle$ transition (with long radiative decay). Sending a signal pulse together with a strong control laser field in such a way that they are in two-photon resonance with the storage state, but both detuned of a small amount with respect to the linewidth of $|e\rangle$, causes a quantum interference phenomenon that opens a window of transparency and strong dispersion in an otherwise opaque medium. Inside such window, the group velocity of light is greatly decreased, in such a manner that the signal pulse is effectively compressed. The realization of a quantum memory exploiting this effect is obtained by turning off the control field when the signal pulse is completely inside the EIT medium. The medium returns opaque , impeding further propagation. What transpires in reality is that the photons of the pulse are mapped to collective atomic excitations delocalized across the entire ensemble. Within the coherence time of the memory, the control field can be turned on again at will to perform a read operation. It is worth noting that quantum storage and retrieval can be realized also in the case in which the detuning from the $|e\rangle$ state is larger than the linewidth of the state itself. In that case, the physical mechanism involved is off-resonant Raman scattering.

1.4.2 Atomic Frequency Comb (AFC)

AFC memories are implemented with inhomogeneously broadened media, i.e. arrays of systems that feature a narrow transition inside a matrix: every individual system will have its particular surroundings, causing the global profile of the transition to be broadened. Such a broadened profile can be optically pumped via spectral hole burning in a periodic comb-like structure. An excitation absorbed by such a profile will naturally dephase until it hits the following repetition of the periodic profile. When that happens, the excitation is reemitted as a photon. The storage time of the memory is fixed by optical pumping, that sets the period of the comb.

Table 1.1: Comparison between EIT and AFC

EIT	**AFC**
On demand readout*	Fixed storage time
Needs filtering of the noise introduced by the control beam	Requires no control fields
Naturally compatible with spatial multiplexing [1]	Spatial multiplexing is not as straightforward as temporal, but feasible
Temporal multiplexing is extremely difficult and performs poorly	Naturally compatible with temporal multiplexing

1.4.3 Additional Protocols[7]

A partial solution to the fixed AFC storage time problem comes from a "contamination" with EIT: once the signal excitation has been absorbed by the comb, a control pulse can map it to an additional state, i.e. store it as a collective spin wave. The readout procedure includes retrieving the excitation from the third state and waiting for the intrinsic rephasing time to have reemission. Two additional relevant protocols work in a different direction: whenever an ensemble collectively absorbs a photon, the individual emitters undergo a natural dephasing process. Such process is different for each individual emitter and is therefore unpredictable. Techniques such as CRIB and its improved version GEM rely on controlling this dephasing through external factors such as electric or magnetic field: keeping the field for some time and then reversing it flips the dephasing of every individual emitter, effectively rephasing the stored excitation and reemitting it.

1.5 Other useful theoretical concepts

1.5.1 Multiplexing

Multiplexing is a vital property when working with quantum memories: as mentioned in the introduction, entanglement generation is a probabilistic process. When distributing an entangled state between two memories of efficiency η, the communication trial is successful only if both memories work, and therefore it depends on η^2. However, if the two memories are designed in such a way as to support photon storage in more than one mode, the communication attempt is considered successful as long as any one mode in memory A is entangled with any mode in memory B. This effectively increases the probability of successful entanglement, ultimately boosting the distribution rate. As seen explicitly in [8], a first approximation for the effects of multiplexing is that the heralding rate is linear on the number of employed modes, while the state quality does not appreciably vary. Such an approximation is acceptable in the scope of the present work: therefore, the simulator code is always intrinsically monomode and the effects of multiplexing are modeled with a scaling factor on the heralding frequency, while the efficiency of the single mode can be replaced in the `SaR_efficiency` parameter of the `SaR.m` function. The only limitation introduced by this approach is that all modes will be assumed to have the same efficiency. This is not a limiting assumption: there exist some memories whose modes have different efficiencies, but it is always possible to define an effective mode number and efficiency to successfully model the memory's behavior.

1.5.2 From Fock Space Entanglement to Polarization Entanglement

All the protocols and techniques described until now, as seen in their simplest implementation, are suitable for the distribution of a single photon state, namely an entangled state in the Fock basis. Although the states themselves are relatively simple to distribute, many single-qubit operations are quite difficult to perform in the Fock basis. A solution to this problem is introduced in the ground-breaking paper published in 2001 by Duan, Lukin, Cirac and Zoller (hereby DLCZ)[4]: at first, a method to generate a single photon entangled state is introduced by using two atomic ensembles of Λ systems to create Fock space entanglement[1] and then the authors present how such a state can be converted to a polarization-like entangled state:

- To achieve conversion, one needs two sets of ensembles $L_1 - L_2$ and $R_1 - R_2$.
- The first step is distributing two independent Fock space entangled states across $L_1 \leftrightarrow R_1$ and $L_2 \leftrightarrow R_2$.

[1] Entanglement is created through off-resonant Raman excitation of two ensembles and mixing of the two Stokes photons: detecting a single Stokes photon without being able to pinpoint which ensemble has emitted projects the two ensembles in an entangled state.

- The second step is a projection of the system in a polarization-like entangled state, carried out via Bell state measurements.

In their article, the authors demonstrate how their protocol can reproduce standard quantum information setups such as teleportation and Bell inequality violation.

1.5.3 Spontaneous Parametric Down-Conversion (SPDC)

All the sources mentioned in this work rely on SPDC to generate photon pairs.
A strict theoretical derivation[14] of SPDC is not in the scope of this document, but to gain an intuitive grasp of it one can think of SPDC as the time-reversed equivalent of Sum Frequency Generation (degenerate SPDC) or Second Harmonic Generation (non-degenerate SPDC): one pump photon at a wavelength λ enters a strongly non-linear medium, and two higher-wavelength photons (conventionally named signal and idler) come out. SPDC is therefore a second-order non-linear optical process. Despite having extremely low efficiency, this process is ubiquitous in quantum information due to its ability to generate entangled pairs, as shown in figure 1.1:

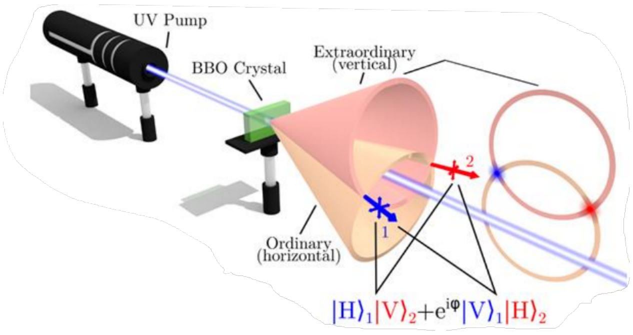


Figure 1.1: Entangled state creation through SPDC by a β-BBO crystal. The entangled photons are in the crossing points between the cones. Figure adapted from [2]

For the purpose of this document, an SPDC source is approximated as a black box with three figures of merit:

- The pair emission rate, i.e. how often the source emits a pair of photons;
- The conditional autocorrelation function $g^{(2)}(0)$, which quantifies how often double pairs are emitted. $g^{(2)}(0) \sim \frac{2P_2}{(P_1+2P_2)^2}$[10], where P_1 and P_2 are the probabilities to emit one and two pairs respectively;
- The Heralding Efficiency, i.e. the probability that given detection of a photon on one output line there is a photon on the other one as well.

Note that the $g^2(0)$ parameter marks the trade-off that every SPDC application must satisfy: increasing the pump power linearly raises the pair emission rate, but it also increases the probability to emit multiple pairs. Throughout this document, the discussed process will actually be cSPDC, i.e. cavity-enhanced SPDC. Cavities are required to control the emission wavelength (various combinations of signal, idler and pump can be resonant with the cavity) and most importantly bandwidth, since a regular SPDC process has a bandwidth in the order of GHz, far higher than what can be compatible with memories.

1.5.4 Useful Definitions

Throughout the document, various figures of merit of different setups are discussed. As a reference, their definition is provided below:

Concurrence of an entangled state

When the state of a system is not entangled, it is said to be separable: the exact opposite of a separable state is a maximally entangled state. In-between a separable state and a maximally entangled state, infinite possibilities exist in terms of "how much" two systems are entangled. A quantification of entanglement E in its own right can be provided in theoretical terms by passing through the concept of Von Neumann Entropy, although there is a set of functions that are monotonically related to it and equivalently serve as quantifiers of the entanglement of a state. One such function is the concurrence, which is also related to experimentally measurable values, thus often being the preferred figure of merit in evaluating entanglement. In particular, the expression of concurrence that will be used in the present document refers to two-photon states and is reported in [9] as:

$$C = max(2|d| - 2\sqrt{p_{00}p_{11}}, 0)$$

where $d \approx V\frac{p_{10}+p_{01}}{2}$. In this last equation, V is an interference fringe visibility. It is obtained by combining the two entangled photons in a 50/50 beam splitter and recording the count rate as a function of the phase shift between the two BS inputs. All the other terms have various difficulties with regard to actual experimental measurement, but they are all accessible through standard photon statistics methods. The concurrence ranges from zero to one, with higher values entailing better entanglement.

Multiphoton suppression h

The reason why some of the measurements above are difficult to perform is that in order have nonzero concurrence, i.e. an entangled state, another requirement is to observe a suppression of two-photon events. This is also a "practical" requirement: when distributing an entangled state, the ideal case would be to distribute a pure entangled state. What is observed in reality is a mixture of an entangled state($a\left|0\right\rangle\left|1\right\rangle + b\left|1\right\rangle\left|0\right\rangle$), a vacuum component ($\left|0\right\rangle\left|0\right\rangle$) and a biphoton component ($\left|1\right\rangle\left|1\right\rangle$). The vacuum component is not a problem in the quantum repeater field, because it is common for protocols to have built-in purification of it[4], but the multiphoton component instead constitutes an issue, therefore it must be minimized. To quantify its presence in the final distributed state, the multiphoton suppression h parameter is introduced:

$$h = \frac{p_{11}}{p_{10}p_{01}}$$

The multiphoton suppression should always be lower than one: the closer it is to zero, the lower the multiphoton contribution in the distributed state is, and the higher the quality of the distributed entangled state is.

Fidelity of a state

Fidelity is a quantity that is defined between two states. When dealing with pure states, one may calculate the overlap between them. Generalizing to mixed states, the Fidelity is found:

$$F(\rho, \sigma) = \left(\mathrm{Tr} \left\{ \sqrt{\sqrt{\rho} \sigma \sqrt{\rho}} \right\} \right)^2$$

Where ρ and σ are the density matrices of the two states. In this particular context, the interest is in the fidelity between the distributed state and the ideal $\sigma = 1/2(|10\rangle \langle 10| + |10\rangle \langle 01| + |01\rangle \langle 10| + |01\rangle \langle 01|)$. However, the present document will not refer to this definition for fidelity. The adopted fidelity will be the effective one, that is calculated by using as ρ the density matrix for the distributed state with its p_{00} component set to zero. Following the form reported in [8], the effective fidelity is calculated throughout this document as:

$$F_e = \frac{(p_{01} + p_{10})(1 + V)}{p_{01} + p_{10} + p_{11}}$$

Chapter 2

The Project

The internship project consisted in assessing the feasibility and the expected performance of a quantum repeater link between LIP6 and LKB, both located in the Jussieu campus of Sorbonne University, in Paris. Despite the envisioned setup being a rather conventional one (local generation of entangled pair with swapping in a central station) [8] [13], the goal of the project is to eventually realize such a link employing the memory technology developed by LKB [1] [16], i.e. EIT on ensembles of Cesium atoms with spatial multiplexing. The high efficiency of these memories makes them suitable for communication in the no-cloning regime [5] and their storage time is sufficient to store a signal for the time it takes to propagate an idler along a few kilometers of fiber. The project featured a strong bibliographical part about quantum memories in general and possible applications [13] [4] [8] [11] [7] [17]. Additional bibliographical work was required to fully understand the material strictly relevant to the project [3] [15] and to obtain suitable input data for the simulations [10] [12]. The practical part of the internship revolved around the modelization of the system. To this end, a bottom-up simulator was implemented out of original MATLAB/Octave code. Professional tools exist and are common in the field, but a bottom-up simulator allows to directly analyze the physics underneath the setup and grants control of low-level parameters that are often cumbersome to work with when using more refined advanced tools.

2.1 The Model

As previously mentioned, the code was written with a bottom-up approach in mind: all the elements of the setup are implemented through (pseudo)random checks to mimic their probabilistic nature. A rendering of the setup and of its correspondence with the code is provided in figure 2.1:

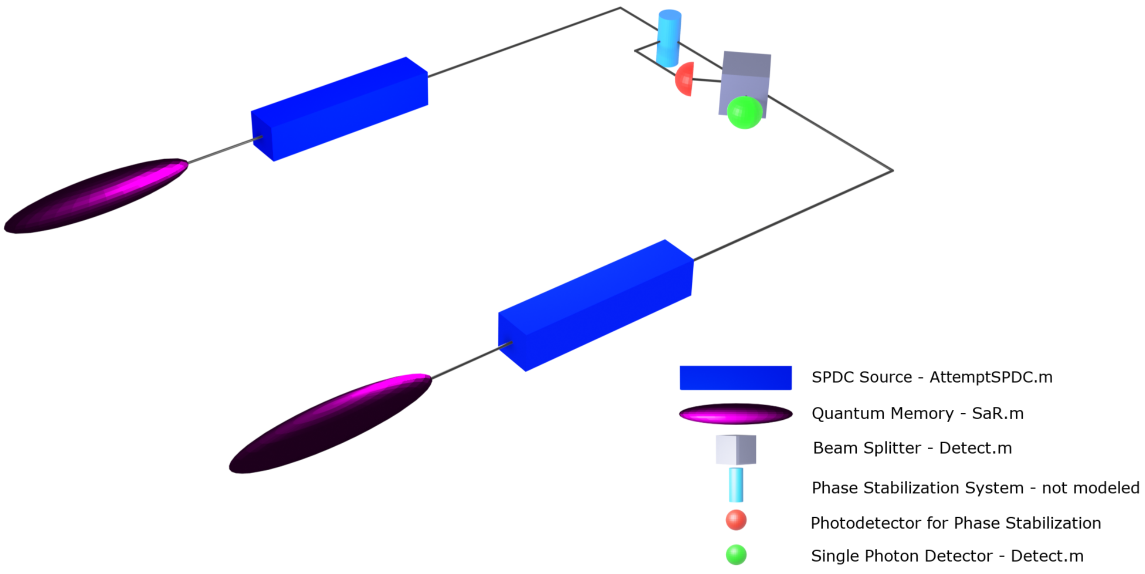

SPDC Source - AttemptSPDC.m
Quantum Memory - SaR.m
Beam Splitter - Detect.m
Phase Stabilization System - not modeled
Photodetector for Phase Stabilization
Single Photon Detector - Detect.m

Figure 2.1: The envisioned setup and its correspondence with the simulation building blocks. The phase stabilization is simulated through the duty cycle parameter in the main code. The detectors for memory readout are not pictured, although they are modeled inside `SaR.m`.

The simulation is implemented with a discrete time axis in mind, i.e. performing a fixed number of communication attempts to which a given time duration is associated. To comment on the physical meaning of the time step parameter, one may think in terms of temporal mode, that is in turn related to the length of the employed single photon pulses: in [8], a communication trial of 25 µs encapsulates 1 to 62 temporal modes and the duration of the single temporal mode is 400 ns. Thus, the relevant time window of emission for SPDC will be the number of active modes times the duration of a single mode.
On the other hand, in the EIT case there is no temporal multiplexing: after considering memory preparation, it is reasonable to say that the communication attempt coincides with the single temporal mode, that in turn is linked to the duration of the employed single photons. The time window is therefore set to 200 ns.

- `Simulator.m` is the main script. It has the purpose of coordinating all the components and provide access to the setup-wide parameters. As there is no physical modeling at this level, the code for the main script is provided in the appendices.

- `AttemptSPDC_WD.m` models the SPDC source at each node: it returns two flags, modelling single and double emission. The inputs to this function are the pump power and the time window on which to verify if SPDC has occurred, while the outputs are two logical flags `emitted` and `DE`. This function will:

  - Calculate the mean number of emitted photons in the experimental time step:

```
em_rate = 2.6607e5*pump_power + 7.1429e4; % data from [10]
%...
p1 = em_rate*time_window;
```

- Obtain the probability of double emission by inverting the conditional autocorrelation expression from [10]:

```
g = 0.0276*pump_power + 0.006;
%...
Δ = (4*p1-2/g)^2 - 16*p1^2;
p2 = (2/g-4*p1 - sqrt(Δ))/8; %Probability to emit two pairs

%p2_1 = (2/g-4*p1 + sqrt(Δ))/8;  %Other solution to the p2 equation,
%consistently > 1, discarded because unphysical
```

- Perform a random check against the probability of double emission: if it is positive, the `DE` flag is set to one. The `rand(m)` function just generates a $m \times m$ array of pseudorandom elements. Called as `rand(1)` it generates a single random number. All checks throughout the present document are handled by calling said function and checking whether its output is below some threshold:

```
DE = (rand(1) < p2);
```

- If the check against P_2 failed (i.e. in the vast majority of cases), an additional check is performed against P_1 to see if single emission occurred, in which case only the single emission flag is risen:

```
if (not(DE))
    emitted = (rand(1) < p1);
end
```

- `SaR.m` models storage and retrieval from one mode of the memory. Its inputs are the number of the active mode (in case the two readout detectors have different specifications) and the simulation time window (to calculate the dark count probability). The experimental parameters defined inside the memory are the storage-and-retrieval efficiency, the specifications of the detectors **and the heralding efficiency of the SPDC source**. The latter is an exception to the modular design adopted throughout the development of the code that simplifies its structure. It is to be stressed that this function is only ever called if a signal photon has been emitted, and hence there is no need to explicitly check for the presence of photons. In the event that no signal photon has been emitted, the `SaR_DC.m` function is called to check for dark counts. The outputs of this function are stored in the `stored` array, whose entries may take a value among 0, 1 and 2, each representing how many photons were stored and retrieved from the memory.

```
if (rand(1) < H\_efficiency)
    success = (rand(1) < SaR\_efficiency)*(rand(1) < D\_efficiency) +...
    (rand(1) < D\_DCR(node));
else
    success = (rand(1) < D\_DCR(node));
end
```

- `Transmit.m` accounts for propagation in the fiber: it takes as inputs the length of the fiber and its attenuation at the idler wavelength, computes the probability for a photon to get through and performs a check against said probability, returning 1 upon success. The outcome of this check is registered in the `arrived` array. Note that, analogously to `SaR.m`, this function is only called when emission has occurred and it is called twice in case of double emission. This implies that any given entry of `arrived` may take 0, 1 or 2 as a value.

- `Detect.m` handles detection of the photons that arrive to the mixing station. Its inputs are the entries of the `arrived` as set by the `Transmit.m` function, the specifics of the detectors and the simulation time step. The output is one entry of the N_a long vector `click`, that logs the number of clicks registered in the active communication attempt.

  Detection is modeled in an iterative way, on the assumption that any repetition of the detection cycle may have to detect a single photon, a pair or vacuum. The cycle is repeated for as many times as the maximum number of photons arriving at either input of the mixing station (e.g. if one photon arrives at input A and two arrive at input B, the code is run twice), and inside every iteration:

  - The two variables `c1` and `c2` keep track of how many photons are still present at the respective output (arr_N1 and arr_N2 being the two inputs mentioned above):

```
1  c1 = (arr_N1 - i);
2  c1 = c1*(c1 > 0); %must remain ≥0
3  c2 = (arr_N2 - i);
4  c2 = c2*(c2 > 0);
```

  - `c1` and `c2` are casted to Boolean flags[1] `f1` and `f2`
  - The actual detection is modeled:

```
if (rand(1) < 0.5)
    click = click + f1*(rand(1)<detector_efficiency) + f2*(rand(1)<detector_efficiency);
end
```

  Notice how the detecting operation runs with 0.5 probability every time. This is due to the fact that only one output of the beam splitter is being monitored. The same check is valid for both single photons and photon pairs because a pair of indistinguishable photons entering a beam splitter from two different inputs will always exit from the same output.

  As a last step before returning, the function checks that no dark counts have occurred and adds them to the click total.

Communication between functions is managed through $2 \times N_A$ arrays of variables, where N_A is the number of communication attempts. The full flowchart of the algorithm is provided in fig. 2.2. Additionally, the complete commented source code is provided in appendix A.

[1] Lines 2 and 4 in the previous step are due to how MATLAB handles casting from integer to logical: any nonzero element is treated as logical one, thus `c1` and `c2` need to be capped at zero to avoid them going negative, but still casting to logical one.

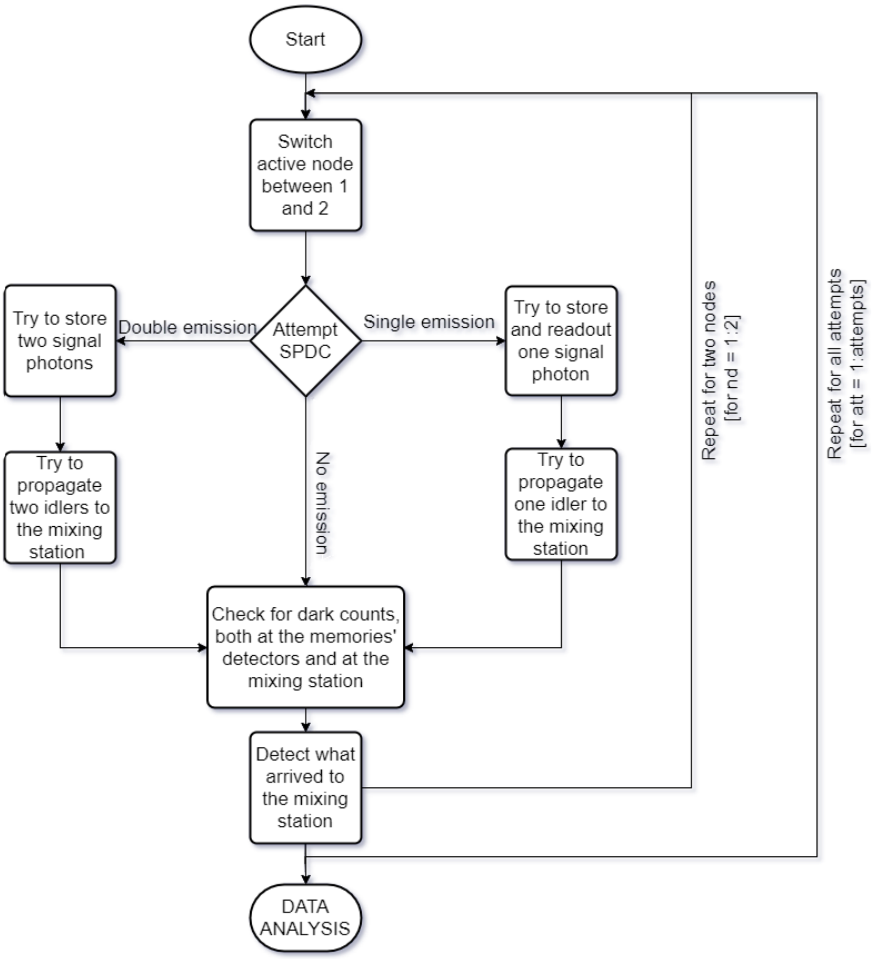

Start
Switch active node between 1 and 2
Attempt SPDC
Double emission
Single emission
No emission
Try to store two signal photons
Try to store and readout one signal photon
Try to propagate two idlers to the mixing station
Try to propagate one idler to the mixing station
Check for dark counts, both at the memories' detectors and at the mixing station
Detect what arrived to the mixing station
Repeat for two nodes [for nd = 1:2]
Repeat for all attempts [for att = 1:attempts]
DATA ANALYSIS

Figure 2.2: Flowchart for the simulation algorithm

Notes on the data analysis step

The main outputs of the code are the concurrence, the multiphoton suppression, the fidelity of the distributed state and the heralding rate. Since the envisioned application is a DLCZ-like setup, the expression for the effective fidelity from [8] is exploited. Therefore, except for the heralding rate, all outputs require the interferometric visibility, that cannot be simulated with this simple of an implementation. In order to still yield results, the interferometric visibility was assumed 85% for all the results.

All terms of the density matrix, which is required for the calculations, are extracted by taking the vectors that store the readouts of the two memories and intersecting various logical combinations of their contents. counting the nonzero elements of said intersections and dividing by the total attempts, thus obtaining an estimate of probabilities. Concerning the heralding rate, The calculation works on the assumption that both SPDC sources emit at the same rate. This is a reasonable assumption because it is equivalent to assuming identical sources with the same pump power at both nodes. The heralding rate is obtained by multiplying the SPDC pair emission rate by an efficiency factor. This efficiency is obtained by calculating the ratio between heralding events and emission events at one node, multiplied by the measurement duty cycle (40% in this case).

Chapter 3

Results

3.1 Validation against experimental data

The first "real world" application of the simulator was the reproduction of the results from [8]: to collect accurate data, a batch version of the simulator code was engineered for two main reasons: first, the computational complexity of the code increases linearly with the number of attempts, but it is not necessary for all the outcomes to be kept in memory at all times. Second, SPDC is an extremely inefficient process, thus having both sources emit in the same communication attempt is even less efficient. This makes all measurements that are dependent on the p_{11} term of the density matrix of the heralded state extremely difficult to measure: out of 1×10^8 communication attempts, it was common not to find any counts at all. As a solution, another piece of code was designed that calls the main script, extracts relevant results and clears the memory for an allotted number of repetitions. This allows to simulate an experiment running for an arbitrary amount of time while keeping in memory only a smaller number of attempts (1×10^8 in this case. The input parameters for this section are summarized in table 3.1.

Table 3.1: Inputs for the code validation section

SPDC Emission Rate, Hz/mW	1513	**Memory Efficiency**	25%
$\mathbf{g^{(2)}(0)}$	0.06	**Fiber length, km**	5
Source Heralding Efficiency	25%	**Fiber attenuation, dB/km**	0.3
Detector Efficiency (Memory)	61%	**Detector Efficiency (Mixing Station)**	80 %
Detector Dark Count Rate (Memory), Hz	10 at node 1, 150 at node 2	**Detector Dark Count Rate (Mixing Station), Hz**	50
Pump Power, mW	2.2	**Interferometric Visibility**	85%
Communication Attempts	270×10^8	**Temporal Mode Length, ns**	400
Active Modes	1	**Communication Trial Duration, µs**	25

Table 3.2 reports the simulation outcome for a three-hour long experimental run. Additionally, the tomography of the distributed state is compared with the one reported in the reference in figure 3.1. Since in the reference work communication is carried out in 25 µs attempts containing a tunable number of 400 ns temporal modes, the length of the temporal mode is used to calculate the SPDC probability of emission, while the length of the attempt is used to compute the total duration of the experimental run (accounting for duty cycle), by which the total number of heralding events will be divided.

Table 3.2: Benchmark of the code against the results from [8]

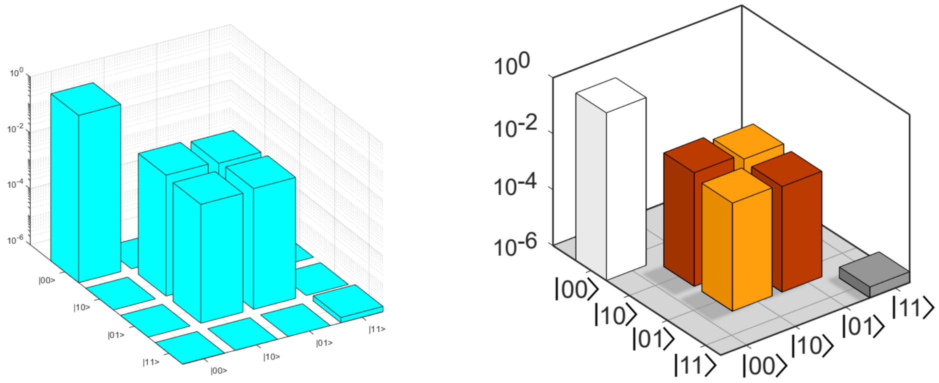
Quantity	**Reference value**	**Simulator output**	**Relative error**
Effective Fidelity	0.92	0.925	0.5%
Heralding Rate [1] (Hz/mode)	13.51	12.38	8%
Concurrence(Measured)	1.15×10^{-2}	2.89×10^{-2}	-
Multiphoton Suppression	0.0036	0.0055	-

At first sight, the reported values for both concurrence and multiphoton suppression suffer from large inaccuracies: this was however expected, since their measurement is extremely difficult in real experimental implementations as well. Managing to reproduce the correct order of magnitude is already a satisfactory result, knowing that simulating an experimental run of the proper length (hundreds of hours) would improve the accuracy. Such a long simulation was impossible to carry out on the available hardware.

The difficulty of measurement is mostly due to the multiphoton component of the heralded state, calculated by looking at the cases in which there was an heralding event, but the readout found two photons per memory. Such an event requires that:

- Both the sources emit in the same communication attempt;
- Only one idler manage to reach the mixing station and be correctly detected;
- Both the memories correctly store and retrieve their excitation.

In an experimental run of 270×10^8 attempts, this event was registered 40 times.



(a) Simulated tomography;

(b) Experimental value, figure adapted from [8];

Figure 3.1: Comparison of the simulated tomography with the experimental one: despite the accuracy not being perfect, the presence of an entangled state is evident.

3.2 Simulation of the true setup

The simulator has been proven accurate against real experimental data. It can therefore be used to estimate how a setup that is similar to [8], but with different parameters, would perform. Such parameters are taken from previous LKB works ([16] and [1] for the EIT memory, [10] for the SPDC source) and feature some differences, the most relevant being the approach to multiplexing: the paper discussed in the previous section employs 62 temporal modes at a rather low efficiency. The EIT memories at LKB on the other hand rely on fewer spatial modes (to be boosted with the aid of spatial light multiplexers/demultiplexers in the near future) whose efficiency is significantly higher. To have some reference to gauge the accuracy of the results, comparisons against their corresponding values in [8] are reported.

3.2.1 Effect of the memory

The memory parameter in the setup of section 3.1 were replaced with the following:

Table 3.3: Specifics of LKB's EIT memory vs. the AFC one used in [8]

Memory	**Efficiency (Single Mode**	**Maximum Number of Modes**
AFC	25%	62
EIT	85%	2, up to 15 in the near future

This case is not expected to show any change in terms of heralding rate, for which the limiting factor is the source. On the other hand, given the much higher efficiency of the memory, the state quality should be greatly improved. Running a three-hour long simulation (long simulations are preferable when measuring concurrence and multiphoton suppression), the results in table 3.4 are obtained, validating the prediction:

Table 3.4: Variation in the state quality parameters when the AFC memory is replaced with the EIT one.

Memory	**Concurrence**	**Multiphoton Suppression**
AFC	2.89×10^{-2}	0.0055
EIT	0.1007	0.0033

The concurrence increased by a factor 3.5 and the multiphoton suppression was almost halved, signaling an entangled state of higher purity.

3.2.2 Effect of the source

The SPDC source available at LKB is brighter than the one employed in the previously presented experiment, featuring a much higher pair emission rate with similar photon statistics. Such an improvement should not affect the quality of the entangled state, although it is expected to noticeably boost the heralding rate. To verify such a prediction, the simulation was run with the same parameters as in the validation step, except for the SPDC source, that was raised to the specifics reported in table 3.5[2]:

Table 3.5: Specifics of LKB's SPDC source vs. the one used in [8]

	Emission Rate	$\mathbf{g^{(2)}(0)}$	**Heralding Efficiency**
[8]	1513 Hz/mW	0.06	25%
LKB	$266\ kHz/mW \cdot P + 71\ kHz$	$0.0276\ (1/mW) \cdot P + 0.006$	70%

The results from this simulation were not completely in line with predictions. The simulation of a three-hour long run yielded a noticeable increase in heralding rate from 12.38 Hz/mode to 738 Hz/mode. However, such a drastic change in the source also degraded the state quality. Despite the measurements still being in the acceptable range for a proof of principle (i.e. concurrence higher than zero, multiphoton suppression below one), such a setup would not be suitable in a DLCZ-like dual chain scheme, that requires the multiphoton suppression times four to be lower than one[9].

The first solution when dealing with a multiphoton component that is too high is always to lower the SPDC pump power. A second simulation of this setup was run with the lowest pump power registered in the source data. The quality of the state was in fact improved, despite the heralding rate being substantially lower. The state distributed with lower pump power was inside the constraints for extension to a dual chain setup.

[2] The emission rate and autocorrelation for the LKB source were obtained via linear interpolation of fig.4 of [10], hence the requirement to multiply by the pump power

3.2.3 Complete setup

After gauging the effect of the individual components, the actual feasibility study can be performed by replacing realistic, state-of-the-art parameters inside a code embodying the bright SPDC source and the EIT memories. To maintain a fair comparison, all parameters that are unrelated to sources or memories have been kept the same across the two implementations. Nevertheless, the complete set of input parameters for this section is reported in table 3.6:

Table 3.6: Complete parameters for the simulation of the LKB setup.

SPDC Emission Rate, Hz	$266kHz/mW \cdot P + 71kHz$	**Memory Efficiency (single mode)**	85%
$\mathbf{g^{(2)}(0)}$	0.06	**Fiber length, km**	5
Source Heralding Efficiency	70%	**Fiber attenuation, dB/km**	0.3
Detector Efficiency (Memory)	61%	**Detector Efficiency (Mixing Station)**	80 %
Detector Dark Count Rate (Memory), Hz	10 at node 1, 150 at node 2	**Detector Dark Count Rate (Mixing Station), Hz**	50
Pump Power, mW	0.5	**Interferometric Visibility**	85%
Communication Attempts	270×10^8	**Temporal Mode Length, ns**	200
Active Modes	1	**Communication Trial Duration, ns**	200

This last simulation is expected to provide an high heralding rate and the state quality of 3.2.1. The output results are reported in tab. 3.7.

Table 3.7: Results of the complete setup simulation

Heralding Rate	**Concurrence**	**Multiphoton Suppression**
25.714 kHz/mode	0.2321	0.0722

The concurrence is satisfactory, but the multiphoton suppression looks high with respect to expectations. Investigating even lower pump powers would surely improve the result, but it would require additional data from the experimental setup. Notice that such a high Heralding Rate per mode is not surprising, since the AFC memory had 62 modes, while the EIT one has only 2. The total heralding rate, as expected, was observed to be equal to section 3.2.2

Chapter 4

Extension to a dual chain link

As previously mentioned, the true application potential of quantum repeater links lies outside the scope of Fock state entanglement: this section deals with the simulation of two parallel links in terms of heralding rate, concurrence and multiphoton suppression.

The first idea would be to take any one of the simulations reported in the previous sections and modify it so that it runs twice; However, this would double the computational weight of the code and consequently the running time. A different solution is to follow the analysis detailed in [9], that demonstrates how to derive the reduced density matrix of a double chain entangled state starting from the density matrix of the single chain states. The price to pay for this analysis to be feasible is that both the single chain states must be described by the same density matrix: this is not only a reasonable assumption for the developed model, but also the same assumption that would be made if the full computation with two separate simulations was performed. Other than the state quality parameters, the fundamental figure of merit here is how often the two links herald entanglement in the same communication attempt. Once that happens, two of the four memories can be read out through a Bell state measurement to obtain a DLCZ Effective Maximally Entangled state, i.e. a state that is a 50/50 mixture of a polarization-like entangled state and the vacuum (in case both excitations are read out), but will be purified to a maximally entangled state due to the built-in purification capabilities of the DLCZ scheme[4].

Since all the required outputs can be extracted from a single simulation, DLCZ analysis was implemented directly inside the main simulation script. If DLCZ analysis is desired, the user will set the `DLCZ_Visibility` parameter (the visibility of the interference fringes when performing the Bell state readout). If they do not require such analysis, setting the variable to zero will make the program skip it.

```
DLCZ_Visibility = 0.85; % set to 0 to skip DLCZ analysis
%...
if (Perform_DLCZ_analysis)
    %Construct reduced density matrix
    DLCZ_tom = 1/2*diag(diag(tom)); %Assigning p10 and p01
    DLCZ_tom(4,4) = tom(4,4);
    DLCZ_tom(1,1) = 0; %Zeroing out the first entry to simplify the next instruction
    DLCZ_tom(1,1) = 1 - trace(DLCZ_tom);
    DLCZ_tom(2,3) = DLCZ_Visibility/2*(DLCZ_tom(2,2) + DLCZ_tom(3,3));
    DLCZ_tom(3,2) = DLCZ_tom(2,3);
    DLCZ_d = DLCZ_tom(2,3); %just to lighten the following line

    %Assess state quality
    DLCZ_C_nonzero =  2*(abs(DLCZ_d) - sqrt(DLCZ_tom(1,1)*DLCZ_tom(4,4)));
    DLCZ_C_vec = [0 DLCZ_C_nonzero];
    DLCZ_C = max(DLCZ_C_vec);

    DLCZ_h = DLCZ_tom(4,4)/(DLCZ_tom(2,2)*DLCZ_tom(3,3));

    %Determine double heralding rate
    click_P = nnz(click == 1)/attempts; %probability for one link to herald entanglement
    DLCZ_P = click_P^2; %Probability for two links to herald entanglement in the same time step,
                        % Required to calculate rate


    %%%%% Last step (rate estimate) redacted due to errors%%%%%
end
```

4.1 Feasibility and expected performance of the link

Once all the building blocks have been completely fleshed out, the simulation of the actual link becomes possible. The physical distance between the two laboratories places the total length of the setup at around 1 km, thus length sweeps were performed from 300 m to 750 m per half-link. Concerning the multimodality aspect, given the assumption of linearity for the heralding rate and independence for the state quality, one can obtain the results for an arbitrary number of modes, real or effective, just by rescaling the monomode results. The parameters presented in the previous sections were realistic enough for an implementation, therefore the data reported refers to the same parameters as table 3.6, except for the fiber length being swept over six points in the aforementioned interval. Furthermore, the code for this part was run with the `DLCZ_Visibility` set to 0.85 to perform DLCZ analysis as well.

The outputs of the simulation are reported in figure 4.1:

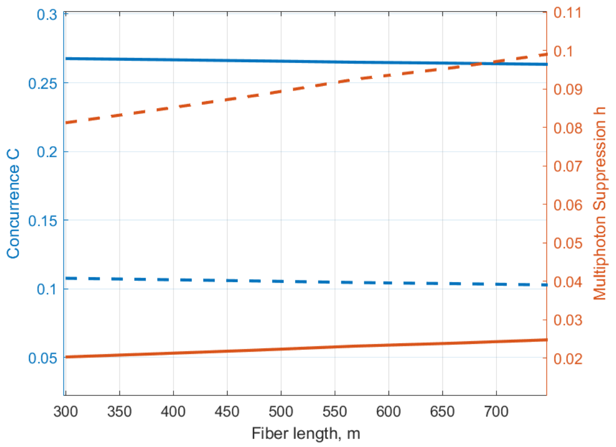


Figure 4.1: Outputs of the final length sweep. Continuous lines are associated with the single-chain state, while dashed ones correspond to the dual-chain one.

As shown in the figure, both the concurrence and the multiphoton suppression are well in the acceptable range for all the length values that were examined, entailing the success of the feasibility study: it is possible to establish a repeater link between LIP6 and LKB with current technology, expecting performance levels that are similar to the ones detailed in the present document.

4.2 Conclusions and outlook

The implemented code has been proven to be able to reproduce detailed enough predictions for a preliminary feasibility assessment, thus the remaining work will be primarily targeted towards optimization and accuracy. Even if it could intuitively look like a trade-off (higher accuracy meaning more memory requirements), the presented implementation has abundant space for optimization in terms of overhead: it is estimated that merely storing the intermediate variables such as `emitted` as `sparse`[1] will reduce memory requirements by more than 90 % and is therefore the first short-term goal.

Beyond that, additional work could be made to try and make the code faster (see Vectorized Code appendix) or improve its accuracy by implementing more refined models for some of the physical steps of the process.

[1] When storing arrays as sparse, only the memory address of non-zero elements is stored. This is particularly efficient for arrays whose non-zero elements are a small percentage of the size, making it a dramatic improvement in the present case.

This internship was carried out almost completely in remote due to COVID restrictions. Nonetheless, it was a deeply fulfilling opportunity to work inside such a challenging and interesting, but also warm and welcoming environment. I wish to thank all the QI team at LIP6, in particular my advisor prof. E.Diamanti, for the guidance and continued support. I also would like to thank all the researchers from LKB, in particular prof. J. Laurat, who also offered their support and help throughout the whole internship, both remotely and in person. Despite the trying time this project was carried out in, it was an honor and a pleasure to work with you all.

Bibliography

[1] Mingtao Cao et al. “Efficient reversible entanglement transfer between light and quantum memories”. In: *Optica* 7.10 (Oct. 2020), pp. 1440–1444. DOI: 10.1364/OPTICA.400695. URL: http://www.osapublishing.org/optica/abstract.cfm?URI=optica-7-10-1440.

[2] Christophe Couteau. “Spontaneous parametric down-conversion”. In: *Contemporary Physics* 59.3 (July 2018), pp. 291–304. DOI: 10.1080/00107514.2018.1488463. arXiv: 1809.00127 [quant-ph].

[3] Jean Dalibard and Claude Cohen-Tannoudji. “Laser cooling and trapping of neutral atoms”. In: (). URL: http://pro.college-de-france.fr/jean.dalibard/publi2/laser_cooling.pdf.

[4] L.-M. Duan et al. “Long-distance quantum communication with atomic ensembles and linear optics”. In: *Nature* 414.6862 (Nov. 2001), pp. 413–418. ISSN: 1476-4687. DOI: 10.1038/35106500. URL: https://doi.org/10.1038/35106500.

[5] Frédéric Grosshans and Philippe Grangier. “Quantum cloning and teleportation criteria for continuous quantum variables”. In: *Phys. Rev. A* 64 (1 June 2001), p. 010301. DOI: 10.1103/PhysRevA.64.010301. URL: https://link.aps.org/doi/10.1103/PhysRevA.64.010301.

[6] Stephen E. Harris. “Electromagnetically Induced Transparency”. In: *Physics Today* 50.7 (1997), pp. 36–42. DOI: 10.1063/1.881806. eprint: https://doi.org/10.1063/1.881806. URL: https://doi.org/10.1063/1.881806.

[7] Khabat Heshami et al. “Quantum memories: emerging applications and recent advances”. In: *Journal of Modern Optics* 63.20 (2016). PMID: 27695198, pp. 2005–2028. DOI: 10.1080/09500340.2016.1148212. eprint: https://doi.org/10.1080/09500340.2016.1148212. URL: https://doi.org/10.1080/09500340.2016.1148212.

[8] Dario Lago-Rivera et al. “Telecom-heralded entanglement between multimode solid-state quantum memories”. In: *Nature* 594.7861 (June 2021), pp. 37–40. ISSN: 1476-4687. DOI: 10.1038/s41586-021-03481-8. URL: https://doi.org/10.1038/s41586-021-03481-8.

[9] Julien Laurat et al. “Towards experimental entanglement connection with atomic ensembles in the single excitation regime”. In: *New Journal of Physics* 9.6 (June 2007), pp. 207–207. DOI: 10.1088/1367-2630/9/6/207. URL: https://doi.org/10.1088/1367-2630/9/6/207.

[10] Hanna Le Jeannic et al. “High-efficiency WSi superconducting nanowire single-photon detectors for quantum state engineering in the near infrared”. In: *Optics Letters* 41.22 (2016), pp. 5341–5344. DOI: 10.1364/OL.41.005341. URL: https://hal.sorbonne-universite.fr/hal-01420824.

[11] M. Pompili et al. "Realization of a multinode quantum network of remote solid-state qubits". In: *Science* 372.6539 (2021), pp. 259–264. ISSN: 0036-8075. DOI: 10.1126/science.abg1919. eprint: https://science.sciencemag.org/content/372/6539/259.full.pdf. URL: https://science.sciencemag.org/content/372/6539/259.

[12] Daniel Rieländer et al. "Cavity enhanced telecom heralded single photons for spin-wave solid state quantum memories". In: *New Journal of Physics* 18.12 (Dec. 2016), p. 123013. DOI: 10.1088/1367-2630/aa4f38. URL: https://doi.org/10.1088/1367-2630/aa4f38.

[13] Nicolas Sangouard et al. "Quantum repeaters based on atomic ensembles and linear optics". In: *Rev. Mod. Phys.* 83 (1 Mar. 2011), pp. 33–80. DOI: 10.1103/RevModPhys.83.33. URL: https://link.aps.org/doi/10.1103/RevModPhys.83.33.

[14] James Schneeloch et al. "Introduction to the absolute brightness and number statistics in spontaneous parametric down-conversion". In: *Journal of Optics* 21.4 (Feb. 2019), p. 043501. DOI: 10.1088/2040-8986/ab05a8. URL: https://doi.org/10.1088/2040-8986/ab05a8.

[15] Oliver Slattery et al. *Background and Review of Cavity-Enhanced Spontaneous Parametric Down-Conversion*. en. 2019-08-22 2019. DOI: https://doi.org/10.6028/jres.124.019.

[16] Pierre Vernaz-Gris et al. "Highly-efficient quantum memory for polarization qubits in a spatially-multiplexed cold atomic ensemble". In: *Nature Communications* 9.1 (Jan. 2018), p. 363. ISSN: 2041-1723. DOI: 10.1038/s41467-017-02775-8. URL: https://doi.org/10.1038/s41467-017-02775-8.

[17] Stephanie Wehner, David Elkouss, and Ronald Hanson. "Quantum internet: A vision for the road ahead". In: *Science* 362.6412 (2018). ISSN: 0036-8075. DOI: 10.1126/science.aam9288. eprint: https://science.sciencemag.org/content/362/6412/eaam9288.full.pdf. URL: https://science.sciencemag.org/content/362/6412/eaam9288.

Appendix A

Complete Source Code of the Simulator

BatchSimulator.m

```matlab
%This block's purpose is to iteratively run the simulation and perform data analysis at the end.
% DLCZ analysis reference: [9]
clear
close all
clc
%INPUTS
N = 135; % number of repetitions of the simulation
V = 0.85; % Interferometric visibility when mixing memory readouts
DLCZ_Visibility = 0.85; % set to 0 to skip DLCZ analysis
lengths = linspace(0.300,0.750,6); % Fiber lengths to sweep
activemodes = 1; % number of modes in the simulation
%%%%%%%%
attempts = 1e8; % Communication attempts to perform per single call to the Simulator
                %function, only really dependent on the host machine's memory, reduce
                %this and increase N to lower memory requirements while keeping the same
                %data set
WaitTime = 0.0474; % s, this is the duration of the preparation of the memory, which
                   % must be performed at a 20 Hz rate as per reference.
time_window = 200e-9; % Time step for the simulation
prep_rate = 20 * time_window; %A 20 Hz rate corresponds to a rate of prep_rate time_window^-1
prep_number = 2.5*attempts*prep_rate; % The preparation cycle must be repeated prep_number
                                      % times during the whole simulation
%Preallocation of outputs
freq_Herald = zeros(1,length(lengths));
C = zeros(1,length(lengths));
F = zeros(1,length(lengths));
h = zeros(1,length(lengths));
if (DLCZ_Visibility) % Preallocation of DLCZ outputs only performed if DLCZ active
    DLCZ_C = zeros(1,length(lengths));
    DLCZ_h = zeros(1,length(lengths));
    DLCZ_freq_Herald = zeros(1,length(lengths));
end
```

```
%%% MAIN CODE %%%%%%%%%
for i = 1:length(lengths)
    len = lengths(i); % fiber length for this attempt
    counts = zeros(4,4);
    click = 0;
    for run = 1:N
        [cnts, clk, attempts] = Simulator(attempts, time_window, V, len);
        counts = counts + cnts; % Iteratively adding up the simulation outcomes
        click = click + nnz(clk == 1);
    end
% Data analysis at each step
    % Frequency estimation code redacted
    d = V*(tom(2,2) + tom(3,3))/2; % Coherence term
    tom(2,3) = d;
    tom(3,2) = d;
    %Concurrence:
    C_nonzero =  2*(d - sqrt(tom(1,1)*tom(4,4)));
    C_vec = [0 C_nonzero];
    C(i) = max(C_vec);
    %Fidelity:
    F(i) = 1/2*(tom(2,2)+tom(3,3))*(1+V)/(tom(2,2)+tom(3,3)+tom(4,4));
    %Multiphoton Suppression:
    h(i) = tom(4,4)/(tom(2,2)*tom(3,3));
    %DLCZ:
    if (DLCZ_Visibility)
        %Construct reduced density matrix as per reference
        DLCZ_tom = 1/2*diag(diag(tom)); %Assigning p10 and p01
        DLCZ_tom(4,4) = tom(4,4);
        DLCZ_tom(1,1) = 0; %Zeroing out the first entry to simplify the next instruction
        DLCZ_tom(1,1) = 1 - trace(DLCZ_tom);
        DLCZ_tom(2,3) = DLCZ_Visibility/2*(DLCZ_tom(2,2) + DLCZ_tom(3,3));
        DLCZ_tom(3,2) = DLCZ_tom(2,3);
        DLCZ_d = DLCZ_tom(2,3); %just to lighten the following line

        %Assess state quality
        %Concurrence:
        DLCZ_C_nonzero =  2*(abs(DLCZ_d) - sqrt(DLCZ_tom(1,1)*DLCZ_tom(4,4)));
        % Same calculations as before with a different matrix
        DLCZ_C_vec = [0 DLCZ_C_nonzero];
        DLCZ_C(i) = max(DLCZ_C_vec);
        %Multiphoton suppression:
        DLCZ_h(i) = DLCZ_tom(4,4)/(DLCZ_tom(2,2)*DLCZ_tom(3,3));
        %Determine double heralding rate
        click_P = click/(N*attempts); %probability for one link to herald entanglement
        DLCZ_P = click_P^2; % Probability for two links to herald entanglement in
                            % the same time step
        %Frequency estimation code redacted
    end
end
```

Simulator.m

```
% Main Simulator Block
% Reference : [1]
function [counts, click, attempts] = Simulator(attempts, time_window, V, fiber_length)
% PARAMETERS
fiber_attenuation = 0.3; % dB/km, attenuation of the fibers at the idler wavelength
detector_efficiency = 0.9; % Efficiency of the SPD at the mixing station
detector_DC = 50; % Dark Count rate of the SPD at the MS, Hz
PumpPower = 0.5; % mW, pump power for the SPDC pump.

% Intermediate variables
emitted = zeros(2,attempts); % Single emission happened at [node,attempt]
DE = zeros(2,attempts); % Double emission happened at [node,attempt]
stored = zeros(2,attempts); % Photon successfully stored and retrieved [node,attempt]
arrived = zeros(2,attempts); % idler arrived at [node,attempt]
click = zeros(1,attempts); %Clicks at the mixing station detector [attempt]

for att = 1:attempts
    for nd = 1:2 %Two separate nodes
        [emitted(nd,att), DE(nd,att)] = AttemptSPDC(PumpPower,time_window);
        if ((emitted(nd,att) == 1) && (DE(nd,att) == 0))  % If SPDC went correctly and
                                                          % no double emission, try to
                                                          % store the signal photon and
                                                          % to propagate the idler.
            stored(nd,att) = SaR(nd,time_window);
            arrived(nd,att) = Transmit(fiber_attenuation,fiber_length);
        elseif (emitted(nd,att) == 1) % If SPDC went correctly BUT there was double
                                      % emission,
            stored(nd,att) = SaR(nd,time_window) + SaR(nd,time_window); % Store and read
                                                                        % two photons
            arrived(nd,att) = Transmit(fiber_attenuation,fiber_length) + ...
            Transmit(fiber_attenuation,fiber_length); %Transmit two idlers
        else % If there was no emission, only check for dark counts in the memory blocks.
             % This will play a role only in the extremely unlikely case of dark count both
             % at one memory and at the mixing station.
            stored(nd,att) = SaR_DCO(nd,time_window);
        end

    end
  % Detect the photons that arrive at the mixing station
  click(1,att) = Detect(arrived(1,att),arrived(2,att),detector_efficiency,detector_DC,time_window);
end

% DATA CONDITIONING

mem1 = stored(1,:); % Reading out the memories, just to make the following steps more readable.
mem2 = stored(2,:);

% Assembling the counts in a matrix
```

```
n11 = (mem1 == 1).*(mem2 == 1).*(click == 1);
n01 = (mem1 == 0).*(mem2 == 1).*(click == 1);
n10 = (mem1 == 1).*(mem2 == 0).*(click == 1);
n00 = (mem1 == 0).*(mem2 == 0).*(click == 1);

counts = nnz(n10)*diag(ones(4,1)); % Building (part of) the reduced density matrix.
                                   % Using a matrix instead of a vector is a leftover
                                   % from the old implementation.
counts(1,1) = nnz(n00);
counts(4,4) = nnz(n11);
counts(2,2) = nnz(n01);
end
```

AttemptSPDC.m

```
% Check if SPDC single emission and double emission have appened in the time_window.
% Reference: [10]
function [emitted, DE] = AttemptSPDC(pump_power,time_window)
%%% Experimental parameters of the source
em_rate = 2.6607e5*pump_power + 7.1429e4; % Obtained by linearly interpolating the
                                          % data in fig. 4 of the reference
g = 0.0276*pump_power + 0.006;
%%% Default values for the output flags
DE = 0;
emitted = 1;

p1 = em_rate*time_window; %Probability to emit one pair

Δ = (4*p1-2/g)^2 - 16*p1^2;
p2 = (2/g-4*p1 - sqrt(Δ))/8; % Probability to emit two pairs obtained by
                                 % inverting the g expression from the reference

    %p2_1 = (2/g-4*p1 + sqrt(Δ))/8;  %Other solution to the p2 equation,
    %consistently > 1, discarded because unphysical

DE = rand(1) < p2; % Check if double emission has occurred. If not, check for
                   % single emission. Notice that if it has occurred, single
                   % emission keeps its default value of 1.
if (not(DE))
    emitted = rand(1) < p1;
end
end
```

SaR.m

```
% Store and retrieve a photon from the quantum memory. Note that this block
% also models the heralding efficiency of the source.
% Reference :  [8]
% This function is called as many times as there are emission events by AttemptSPDC.
function success = SaR(node,time_window)
% Experimental parameters
SaR_efficiency = 0.85; % Storage and retrieval efficiency
D_efficiency = 0.61; % Detector efficiency, from reference
D_DCR = time_window*[10 150]; % Dark count rate, from reference
H_efficiency = 0.70; % From [10], since it is actually a source
                     % parameter (Heralding efficiency)
success = 0;
if (rand(1) < H_efficiency) % Checking heralding efficiency here makes sense
                            % because the only case that will actually influence
                            % the results is if there is only idler and no signal.
                            % If there is only signal and no idler, the state is not
                            % heralded.
    success = (rand(1) < SaR_efficiency)*(rand(1) < D_efficiency) + ...
        (rand(1) < D_DCR(node)); % Check against both memory and detector efficiency
else
    success = (rand(1) < D_DCR(node)); % If the heralding efficiency check failed,
                                       % check for dark counts alone
end
end
```

SaR_DC.m

```
function DC = SaR_DCO(node,time_window)
DC = 0;
D_DCR = time_window*[10 150]; % DC probability from [8]
DC = (rand(1) < D_DCR(node)); % Check for dark counts
end
```

Transmit.m

```
% This function checks whether a photon manages to cross a fiber of given
% length and attenuation.
function arrived = Transmit(fiber_attenuation,fiber_length)
arrived = 0;
att_dB = fiber_attenuation*fiber_length; % Total attenuation
prob_T = 10^(-att_dB/10); % Cast to probability
arrived = (rand(1) < prob_T); % Check if arrived
end
```

Detect.m

```
%The two branches arrive at the mixing station: this function evaluate
%whether they come out in the right detecting mode and whether they are
%detected. arr_N1 and arr_N2 are the number of arrived photons at each
%beam splitter input (0,1,2). DCR is the Dark Count Rate of the detector.
function  out = Detect(arr_N1,arr_N2,detector_efficiency,DCR,gate_time)
click = 0;
for i = 0:max(arr_N1,arr_N2) % The cycle is repeated as many times as
                             % the highest number of photons at either input.

    detector_DC = DCR*gate_time; % Probability to register a dark count in the time window

    c1 = (arr_N1 - i); % These two variables count the photons that still have to be
    c1 = c1*(c1 ≥ 0); % detected at port 1 or 2.
    c2 = (arr_N2 - i);
    c2 = c2*(c2 ≥ 0);

    f1 = logical(c1); % Casting to boolean to check whether *something* to detect is still
    f2 = logical(c2); % there, be it one or two photons. This simplifies the following check.

    if (rand(1) < 0.5) % The following line is executed with 1/2 probability, same probability
                       % whether detecting single photon or indistinguishable pair.

        % The check for detection is carried out only if there is still something
        % waiting at the input.
        click = click + f1*(rand(1)<detector_efficiency) + f2*(rand(1)<detector_efficiency);
    end
end
click = click + (rand(1) < detector_DC); % Final check for dark counts
out = click;
end
```

A.1 Vectorized Implementation

When looking at the presented implementation of the simulator, one may argue that it does not follow the best MATLAB practices, in that it uses a large `for` loop instead of one-shot calls to functions that operate on vectors. Such an implementation was also attempted, and it yielded a marginal speed-up (about 5%) on the machine on which all the tests were performed, while greatly reducing the readability of the code to a potential user. Despite the sequential version having been used for all the results in the present document, the vector version is presented in the following, together with an example output on the parameters of section 3.1 to demonstrate it produces the same results. Notice that the scaling of this vector implementation should be better than the sequential one, making it more viable to run simulations on more powerful machines. For brevity, only the parts of the various blocks that actually differ from the sequential code have been reported[1]:

Simulator.m

[1] This version of the code presents extremely long lines. To preserve readability, some lines have been forcibly broken and marked by "...", despite it not always being in the proper place for MATLAB syntax

```
%Since the preamble is the same as the sequential code, only the main part is reported:
%...
for nd = 1:2 %Two separate nodes
    [emitted(nd,:), DE(nd,:)] = AttemptSPDC_WD(PumpPower,time_window,attempts);
    %All SPDC outcomes are handled in a single logical expression:
    stored(nd,:) = (emitted(nd,:) == 1).*(SaR(nd,attempts).*(DE(nd,:) == 0) + ...
    (SaR(nd,attempts) + SaR(nd,attempts)).*(DE(nd,:) == 1)) + ...
    (emitted(nd,:) == 0).*SaR_DC(nd,attempts);
    arrived(nd,:) =  (emitted(nd,:) == 1).*(Transmit(fiber_attenuation,...
    ...fiber_length,attempts).*(DE(nd,:) == 0) + ...
    (Transmit(fiber_attenuation,fiber_length,attempts) + ...
    Transmit(fiber_attenuation,fiber_length,attempts)).*(DE(nd,:) == 1));
end
click = Detect_new(arrived(1,:),arrived(2,:),detector_efficiency,...
...detector_DC,time_window,attempts); % Clicks at the detecting station
```

AttemptSPDC.m

```
function [emitted, DE] = AttemptSPDC(pump_power,time_window,attempts)
    %Same definition of parameters and calculation of probabilities
    %p1 and p2 as sequential case
    %...
    DE = rand(1,attempts) < p2;
    emitted = (not(DE).*(rand(1,attempts) < p1)) + DE;
end
```

SaR.m

```
function success = SaR(node,attempts)
    %Same parameters as sequential case
    success = (rand(1,attempts) < H_efficiency).*(rand(1,attempts) <...
    ...SaR_efficiency).*(rand(1,attempts) < D_efficiency) + ...
    (rand(1,attempts) < D_DCR(node));
end
```

Transmit.m

```
function arrived = Transmit(fiber_attenuation,fiber_length,attempts)
    % Same parameters as sequential case
    arrived = (rand(1,attempts) < prob_T);
end
```

Detect.m

```
function  out = Detect_new(arr_N1,arr_N2,detector_efficiency,detector_DC,...
gate_time,attempts)
    out = (rand(1,attempts) < 0.5).*((arr_N1 == 1).*(rand(1,attempts) < ...
    detector_efficiency) + (arr_N2 == 1).*(rand(1,attempts) < ...
    detector_efficiency) + (arr_N1 == 2).*((rand(1,attempts) < ...
    detector_efficiency) + (rand(1,attempts) < detector_efficiency)) + ...
     (arr_N2 == 2).*((rand(1,attempts) < detector_efficiency) + ...
     (rand(1,attempts) < detector_efficiency))) + (rand(1,attempts) <...
    detector_DC*gate_time);
end
```

This vectorial implementation reproduces the same results seen in section 3.1.