Numerical model for macroscopic quantum superpositions based on phase-covariant quantum cloning

A. Buraczewski \(^a\), M. Stobińska \(^b,*\)

\(^a\) Faculty of Electronics and Information Technology, Warsaw University of Technology, Warsaw, Poland
\(^b\) Institute of Physics, Polish Academy of Sciences, Warsaw, Poland

ABSTRACT

Macroscopically populated quantum superpositions pose a question to what extent the macroscopic world obeys quantum mechanical laws. Recently, such superpositions for light, generated by an optimal quantum cloner, have been demonstrated. They are of fundamental and technological interest. We present numerical methods useful for modeling of these states. Their properties are governed by a Gaussian hypergeometric function, which cannot be reduced to either elementary or easily tractable functions. We discuss the method of efficient computation of this function for half-integer parameters and a moderate value of its argument. We show how to dynamically estimate a cutoff for infinite sums involving this function performed over its parameters. Our algorithm exceeds double precision and is parallelizable. Depending on the experimental parameters it chooses one of the several ways of summation to achieve the best efficiency. The methods presented here can be adjusted for analysis of similar experimental schemes.

Program summary

Program title: MQSVIS
Catalogue identifier: AEMR_v1_0
Program summary URL: http://cpc.cs.qub.ac.uk/summaries/AEMR_v1_0.html
Program obtainable from: CPC Program Library, Queen’s University, Belfast, N. Ireland
No. of lines in distributed program, including test data, etc.: 1643
No. of bytes in distributed program, including test data, etc.: 13212
Distribution format: tar.gz
Programming language: C with OpenMP extensions (main numerical program), Python (helper scripts).
Computer: Modern PC (tested on AMD and Intel processors), HP BL2x220.
Operating system: Unix/Linux.
Has the code been vectorized or parallelized?: Yes (OpenMP).
RAM: 200 MB for single run for 1000 × 1000 tile
Classification: 4.15, 18.
External routines: OpenMP
Nature of problem:

Recently, macroscopically populated quantum superpositions for light, generated by an optimal quantum cloner, have been demonstrated. They are of fundamental and technological interest. Their properties are governed by Gaussian hypergeometric function \(2F1\) of half-integer parameters, which cannot be reduced to either elementary or easily tractable functions. Computation of the photon number distribution, visibility, and mean number of photons, necessary for characterization of these states, requires evaluation of infinite sums involving this function performed over its parameters.
1. Introduction

Macroscopic quantum superposition (MQS) and entanglement were pointed out by Schrödinger in 1935. He posed a gedanken experiment in which the quantum formalism was applied to a macroscopic object – a cat and a microscopic object – a two-level radioactive atom, describing them in a joint quantum superposition [1]. These objects were closed in a box, and if the atom decayed at a random time, an additional mechanism installed inside was activated to kill the cat. The conclusion was striking: the cat was dead and alive at the same time, as long as no one opened the box. This revealed the phenomenon of quantum entanglement: the state of the cat is random (dead or alive) but perfectly correlated with the state of the atom (decayed or not). It also planted a question to what extent macroscopic objects obey laws of quantum mechanics.

Small MQS have been produced in nanoscale magnets, laser-cooled trapped ions, photons in a microwave cavity, C\textsubscript{60} molecules, superconducting devices [2] and macroscopic-size diamonds, both cooled [3] and at room temperature [4]. In 2007, macroscopically populated polarization entangled superpositions of light were generated in room conditions by optimal quantum cloning (OQC). This is based on parametric frequency down conversion (PDC) [5,6], where a higher energetic blue photon in a given polarization is turned, with some probability, into two lower energetic red photons by a nonlinear crystal preserving the polarization. The more the crystal is pumped with the blue photon beam, the higher the probability of conversion, resulting in multi-photon output. OQC does not violate the no-cloning theorem [7], which states that an unknown quantum state (here: polarization) cannot be copied perfectly. It makes imperfect copies, characterized by a cloning fidelity lower than 1 [8]. There are two kinds of MQS of light produced by OQC: the micro–macro singlet state, in which a single photon is entangled with a “macroscopic qubit”, and the bright entangled squeezed vacuum [6,9]. They contain $10^6$–$10^{13}$ photons.

MQSs are a promising alternative for quantum technologies. They allow exploring the quantum-to-classical transition [10,11], efficient interaction with matter and photons [12], studying the principle of quantum measurement, and they form a non-Gaussian class of quantum states, necessary for fault-tolerant quantum computing [13]. Importantly, they give hope for a loophole-free Bell inequality test [14] enabling the ultimate test of quantum theory.

Theoretical and experimental analysis of MQSs is challenging due to their large complexity and fast decoherence [13], both scaling exponentially with the Hilbert space dimension. Commercial numerical tools (Matlab, Wolfram Mathematica) and libraries (BLAS, NetLib, LAPACK) model MQS only for small populations. Intuitions based on these results are often misleading for high populations [6,15]. Here, a slowly convergent multiple infinite hypergeometric series arises, which is intractable for these applications. This results in the lack of a commonly accepted model of MQS.

In this paper, we develop a numerical model for MQS of light and discuss the method of efficient computation of the Gaussian hypergeometric function for half-integer values of its parameters and a moderate value of the argument. We show how to dynamically estimate a cutoff for infinite sums involving this function performed over its parameters. Our algorithm offers precision exceeding double precision by several orders of magnitude. Depending on the experimental parameters, it chooses one of several ways of summation to achieve the best efficiency. It is parallelizable, which allows better utilization of multi-processor computers. We tested our numerical results with a recently developed analytical model for MQS for certain regimes of parameter values [14]. The numerical methods presented here can be adjusted for analysis of similar experimental schemes.

This paper is organized as follows. In Section 2 we introduce MQSs, their properties and show the origins of the hypergeometric function. In Section 3, we discuss methods used in the numerical model of MQS. Section 4 includes examples: computation of MQS photon number distribution and distinguishability.

2. Theoretical background

2.1. Entangled macroscopic quantum superpositions

In our analysis, we focus on the micro–macro singlet state, but our results can be generalized for the bright squeezed vacuum. The singlet is produced by a phase-covariant optimal quantum cloner [6]. This method requires first a pair of linearly polarized photons created in a usual state through parametric frequency down conversion (PDC), $1/\sqrt{2}(|1\psi\rangle_a|1\psi\rangle_b-|1\psi\rangle_a|1\psi\rangle_b)$. Subscripts $a$ and $b$ denote two spatial modes and $H$ and $V$ denote horizontal and vertical polarization; see Fig. 1a. The equatorial states of the Poincaré sphere of all polarization states, see Fig. 1b, are given by the following transformation of the linear polarization: $a^\psi_\psi = 1/\sqrt{2}(e^{i\psi}a^1_\psi + e^{-i\psi}a^\perp_\psi)$, $a^\psi_\perp = i/\sqrt{2}(e^{i\psi}a^1_\perp - e^{-i\psi}a^\perp_\psi)$, where $a^1_\psi$ and $a^\perp_\psi$ are creators for two orthogonal polarizations $\psi$ and $\psi^\perp$. This subspace, parameterized by the polar angle $\psi\in(0, 2\pi)$, is privileged for the phase-covariant cloner. Here, its Hamiltonian reads $H = i\hbar \Gamma \left( (a^1_\psi)^2 + (a^\perp_\psi)^2 \right) + $ h.c., and this shows that all equatorial states are cloned equally well (the form of $H$ is the same for all $\psi$). Due to rotational invariance of the singlet, it can be expressed in this basis: $1/\sqrt{2}(|1\psi\rangle_a|1\psi\rangle_b-|1\psi\rangle_a|1\psi\rangle_b)$. Next,
one of its spatial modes, e.g. b, is amplified by the cloner to create a multi-photon state \(|\Phi\rangle = e^{\sqrt{g} t_H} |1\rangle_b\) or \(|\Phi\rangle = e^{\sqrt{g} t_H} |1\rangle_a\). This unitary evolution leads to the micro–macro singlet

\[ |\psi^-\rangle = 1/\sqrt{2} (|1\rangle_a |\Phi\rangle_b - |1\rangle_b |\Phi\rangle_a). \]  

(1)

The multi-photon states are an infinite superposition of photon number states

\[ |\Phi\rangle = \sum_{i,j=0}^{\infty} \gamma_{ij} (2i+1)_{\psi}, (2j+1)_{\psi}, \]  

\[ |\Phi\rangle = \sum_{i,j=0}^{\infty} \gamma_{ij} (2i+1)_{\psi}, (2j+1)_{\psi}, \]  

(2)

with the real-valued probability amplitude

\[ \gamma_{ij} = C_g^{-2} (T_g/2)^{i+j} \sqrt{(2i+1)_{\psi}! (2j+1)_{\psi}!} / (i! j!) = C_g^{-2} \gamma_0 \gamma_{ij}, \]  

(3)

where \(\sum_{i,j=0}^{\infty} \gamma_{ij}^2 = 1\). Here, \(g = \int dt = \Gamma\) is the amplification gain, \(C_g = \cosh(g), T_g = \tanh(g)\). The notation \(|(2i+1)_{\psi}\rangle\) denotes \(2i\) photons in polarization \(\psi\) and \(2i+1\) in \(\psi^\perp\), which contribute to the superposition \(|\Phi\rangle\), with probability \(\gamma_{ij}^2\). Fig. 2 depicts \(\gamma_0(i)^2\) and \(\gamma_0(j)^2\) for \(g = 4\) and reveals the slow decay of \(\gamma_0(i)^2\) for \(i\) going to infinity. The convergence of \(\gamma_0(i)\) to zero is slower for higher gain.

The states \(|\Phi\rangle\) and \(|\Phi\rangle\) are orthogonal due to different occupation parity in the polarizations, and they are called macroscopic qubit. In experiments they contained \(4m \approx 10^4\) photons on average, where \(m = \sinh^2 g\). However, in this regime detection is not single-photon resolving, and they reveal effective overlap. Their distinguishability is efficiently quantified by photon number distributions \(p_{\Phi}(k, l) = |\langle k, l |\Phi\rangle|^2\) and \(p_{\Phi}(k, l)\) giving the probability of having \(k\) photons in polarization \(\psi\) and \(l\) in \(\psi^\perp\) simultaneously:

\[ p_{\Phi}(k, l) = p_{\Phi}(k, l) = \begin{cases} \gamma_{k-1,2l/2}^2 & \text{for even } k, \text{ even } l, \text{ odd } l, \text{ odd } l, \text{ otherwise}, \\ \gamma_{k,2l/2}^2 & \text{for odd } k, \text{ even } l, \text{ odd } l, \text{ otherwise}, \end{cases} \]  

(4)

where \(\sum_{k,l=0}^{\infty} p_{\Phi}(k, l) = 1\). The distinguishability is

\[ v = 1 - \sum_{k,l=0}^{\infty} \sqrt{p_{\Phi}(k, l) p_{\Phi}(k, l)}, \]  

(5)

2.2. Additional operations performed on MQS

Partial indistinguishability of \(|\Phi\rangle\) and \(|\Phi\rangle\) is a drawback to any quantum protocol and Bell inequality test, and needs to be fixed by special quantum state engineering. In [14,16,17], a filtering method was suggested to increase the distinguishability of a macro-qubit. It is described by the projective measurement

\[ p_{\Phi}^{(\text{th})}(k, l) = \sum_{k,l=\sigma}^{\infty} |k, l)(k, l|, \]  

(6)

It cuts off the low photon number contributions below a threshold \(\sigma\) in the initial superposition so that the preselected macro states contain \(\sigma\) photons at least distributed over two polarization modes. We will call this operation theoretical preselection. Photon number distributions for MQS subjected to theoretical preselection read \(p_{\Phi}^{(\text{th})}(k, l) = |\langle k, l |p_{\Phi}^{(\text{th})}(\Phi)\rangle|^2\):

\[ p_{\Phi}^{(\text{th})}(k, l) = \begin{cases} |\langle k, l |p_{\Phi}^{(\text{th})}(\Phi)\rangle|^2 & \text{for odd } k, \text{ even } l, \text{ odd } l, \text{ odd } l, \text{ otherwise}, \\ 0 & \text{for odd } k, \text{ even } l, \text{ even } l, \text{ odd } l, \text{ otherwise}, \end{cases} \]  

(7)

For a large class of useful observables, the mean values evaluated for preselected multi-photon states will involve sums of the form \(\sum_{k,l=\sigma}^{\infty} p_{\Phi}^{(\text{th})}(k, l)\), where \(f(p, q)\) is a polynomial. For example, for the mean number of photons one sets \(f(p, q) = p + q\) and \(f(p, q) = p + q^2\) for its variance.

In experiments, theoretical preselection is implemented by a weak measurement. The state is split into two beams, reflected and transmitted, by a beamsplitter (BS) with high transmittivity. Quantum operation \(p_{\Phi}^{(\text{th})}\) is performed only on the reflected part, and the result is fed forward to the transmitted beam (i.e., the summation constraint on occupations is shifted from the reflected to the transmitted part). This is the beam-splitter preselection.
The action of a BS with reflectivity $R$ (transmittivity $T = 1 - R$) on two orthogonal polarizations $\varphi$ and $\varphi_\perp$ is independent. For a given polarization, if one BS input is a vacuum $|0\rangle$ and the other is photon number state $|N\rangle$, its action is given by $\mathcal{U}_{BS}|0\rangle|N\rangle = \sum_{k=0}^{N} c_k^{(N)} |N - k\rangle |k\rangle$, with probability amplitude $c_k^{(N)} = \binom{N}{k} R^k T^{N-k}$, where $t(r)$ denotes the mode where $N - K(k)$ photons were transmitted (reflected). For the multi-photon states we get $\mathcal{U}_{BS}\Phi = \sum_{j=0}^{\infty} \sum_{n=0}^{\infty} |\gamma_j n\rangle \sum_{m=0}^{j} c_m^{(2j)} \left(|2j+1-n\rangle_{\varphi}, (2j-m)_{\varphi_\perp}\right) |n\rangle_{\varphi}, |m\rangle_{\varphi_\perp}$, and an expression for $\mathcal{U}_{BS}\Phi_L$.

The coefficients may be grouped as follows: $c_m^{(2j+1)} |\gamma_j n\rangle = c_m^{(2j)} |\gamma_j n\rangle$. Evaluation of any physical quantity will lead to the following subexpressions:

$$f_i(n, i) = c_\xi^{2j} \gamma_j^{\frac{1}{2}} (\gamma_j^{2j+1})^2$$

$$= c_\xi^{2j} \left(\frac{T_s}{2}\right)^{2i} \frac{(2i+1)^2}{2i!} T^{2i+1+n}$$

and $f_j(m, j) = c_\xi^{2j} \gamma_j^{\frac{1}{2}} (\gamma_j^{2j+1})^2$, where $i, j, n, m$ are non-negative integers. For the beam-splitter preselection, the photon number distributions read

$$p_{\varphi}^{(BS)}(k, l) = |N_{BS}|^2 \sum_{i=0}^{\infty} \sum_{m=0}^{2i+1} f_i(n, i) \cdot f_j(m, j) \cdot \delta_{2i+1+n, 2j+m}$$

with $p_{\varphi}^{(BS)}(k, l) = p_{\varphi}^{(BS)}(l, k)$, where normalization constant $|N_{BS}|^2 = \sum_{i=0}^{\infty} \sum_{m=0}^{2i+1} \sum_{n=0, n+m \geq 0} \sum_{m=0, m \geq 0} c_m^{(2j)} f_i(n, i) \cdot f_j(m, j)$ and $\delta_{i,j}$ is a Kronecker delta function equal to 1 if $i = b$ and 0 otherwise. Here, important mean values are computed by $|N_{BS}|^2 = \sum_{i=0}^{\infty} \sum_{m=0}^{2i+1} \sum_{n=0, n+m \geq 0} c_m^{(2j)} f_i(n, i) \cdot f_j(m, j) \tilde{f}(2i+1+n, 2j+m)$. A beam splitter is useful for many other basic quantum operations, e.g., losses, homodyne detection, inefficient detection, and entanglement distillation.

### 2.3. Hypergeometric function

We notice that the ratios $(\gamma_{j+1,0}/\gamma_{j,0})^2$ and $f_i(n, i+1)/f_i(n, i)$ are rational functions of $i$ and $n$, respectively. According to the definition of a hypergeometric term [18], $\gamma_j^{2j}, f_i, f_j$ are double hypergeometric terms, and their infinite sums over these parameters are hypergeometric functions.

The sum of the probability governing MQS is given by the Gaussian hypergeometric function

$$G(n) = \sum_{j=0}^{\infty} \gamma_j^{2j} = x_n \sum_{i=0}^{\infty} z^{i} F(1, 1 + i, z),$$

$$\tilde{G}(m) = \sum_{j=0}^{\infty} \gamma_j^{2j} = y_m \sum_{j=0}^{\infty} z^{j} F(1, 1 + j, z),$$

where $x_n = (2n+1)! T_{2n} / (C_n^2)^2 T_{2n}^2$, $y_m = (2m)! T_{2m} / (C_m^2)^2 T_{2m}^2$, $z = T_s^2$, $a = n + \frac{1}{2}$, and $b = m + \frac{1}{2}$. The situation is similar for the probability governing the MQS after passing the BS. Here, the sum $A(n) = \sum_{i=0}^{\infty} \sum_{j=0}^{i} f_i(n, i)$ and $B(m) = \sum_{j=0}^{\infty} \sum_{i=0}^{\infty} f_i(n, i)$, where $|x|$ is the floor function $|x| = \max \{m \in \mathbb{Z} | m \leq x\}$, have to be written for odd and even $n$ and $m$ separately.

$$A(2n) = x_n (2n+1)! H_{2n} T_{2n} F(1, 1 + n, z),$$

$$A(2n+1) = x_n (2n+2)! H_{2n+1} T_{2n+1} F(1, 1 + n+1, z),$$

$$B(2m) = y_m (2m)! H_{2m} T_{2m} F(1, 1 + m, z),$$

$$B(2m+1) = y_m (2m+2)! H_{2m+1} T_{2m+1} F(1, 1 + m+1, z).$$

We checked that no closed form of $G(n), \tilde{G}(m), A(n)$, and $B(m)$ exist using Gosper’s, Zeilberger’s, and Petkovšek’s algorithms [18]. These algorithms constitute the standard mathematical tools for hypergeometric series analysis. A closed form, if it existed, would eliminate sums over $i$ and $j$ completely and reduce the computation time.

In our case, these sums have to be computed iteratively. This prevents efficient simulation of MQS properties and evolution for realistic parameters, since it is neither possible to bring this function to elementary ones nor to other special functions (e.g., Bessel functions) more easily computable. The function $f_j$ arises directly from probability governing the MQS $Y_j^{2j}$, which $\gamma_j^{2j}$ are half-integers. This class of Gaussian hypergeometric functions is the least known in the literature. There are only a few known identities simplifying $f_j$ with half-integer parameters [18,19], but they cannot be used here. Moreover, recurrence transformations lead to other half-integer forms of $f_j$. Unconditional convergence of the hypergeometric sum is guaranteed by $z, T_s < 1$. (0, 1).

It is difficult to find the appropriate cutoffs for the sums over $i$ and $j$. High cutoffs make the computation long and error prone, whereas too low values result in omitting significant terms. The fast computation of $\tilde{f}_j$ in a non-asymptotic regime of its arguments and parameters is challenging [20,21]. In the literature, there are algorithms applicable for some special cases of relations between its parameters [19,22,23], but they are not useful here. Moreover, in our intermediate regime they become unstable. Neither Gaussian quadrature [24] nor Padé [25] approximations exist.

To be able to compute Eq. (9) and similar ones of the form

$$S = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} |f_i(n, i) f_j(m, j)|,$$

we aim at partial factorization with respect to $(i, n)$ and $(j, m)$ to compute the sums over $i$ and $j$ separately. Full factorization cannot be achieved due to the preselection condition $n + m \geq \sigma'$. We change the variable order $\sum_{i=0}^{\infty} \sum_{j=0}^{\infty} |f_i(n, i)| = \sum_{n=0}^{\infty} \sum_{m=0}^{\infty} |f_i(n, i)|$ (similarly for $f_j(m, j)$) in Eq. (12):

$$S = \sum_{n=0}^{\infty} \sum_{m=0}^{\infty} A(n) B(m).$$

We avoid four nested sums and get a double sum of independent series. However, it becomes clear that the hypergeometric functions have to be additionally summed up in infinite sums over their parameters.

In the case of computation of mean values $\tilde{f}_j(p, q)$ has first to be split into a sum of monomials $\tilde{f}_j(p, q) = \sum_{n, m} a_{nm} p^n q^m$, where, importantly, $\alpha_{nm}$ are real coefficients. Then hypergeometric terms take the form $\tilde{f}_j(n, i) = f_i(n, i) \cdot (2i + 1 + n)\sigma$ and $\tilde{f}_j(m, j) = f_j(m, j) \cdot (2j - m)\sigma$, where $p$ and $q$ are integers. The analysis presented above for $f_j(m, j)$ still holds for $\tilde{f}_j$ and $\tilde{f}_j$.
3. Numerical methods

3.1. Calculation of hypergeometric terms with high precision

The factorials in the numerators and denominators of hypergeometric terms \( y_0, y_0, f_i \) and \( f_j \) take large values, unavailable for the standard machine arithmetic. IEEE 754 double-precision numbers are capable of storing 15 significant decimal digits and magnitudes \( 10^{-308} - 10^{308} \) [26]. For \( i, j \geq 50 \) the magnitudes exceed \( 10^{308} \), but the values of terms are between 0 and 1 and need to be computed precisely.

One way to compute \( \gamma^2, \gamma^0, f_i \), and \( f_j \) is to use libraries offering enhanced precision arithmetic. For example, the GNU Multiple Precision Arithmetic Library (GMP) [27] and the GNU Multiple Precision Floating-Point Reliable Library (MPFR) [28] are available for C and C++. The Class Library for Numbers (CLN) [29] works with C++ and mpmath [30] with Python. These libraries offer low speed compared to the double-precision arithmetic.

The other possible way of computing hypergeometric terms is to use their logarithmic forms: \( \log y_0, \log y_0, \log f_i \), and \( \log f_j \). The number magnitudes are within the range of double precision for large \( i \) and \( j \). Moreover, both standard C/C++ and MPFR offer the \( \log(\cdot) \) function, which calculates \( \log(\Gamma(\cdot)) = \log((n - 1)!) \) with hardware acceleration.

In order to find the best method of computation of the subexpressions we compared all these mentioned solutions. The relative error estimation was based on the values obtained symbolically with the computer algebra system Wolfram Mathematica. We showed superiority of C/C++ computation with double precision. It is not only several orders of magnitude faster but also consumes less memory. The reason for that is that IEEE 754 arithmetic is built into modern computer processors and does not need any additional memory structures. The average error of double-precision calculation is of the order of \( 10^{-15} \), low enough for numerical simulations. The error can be further decreased at the significant cost of speed with the MPFR library, which is the second fastest solution.

3.2. Convergence rate

Computation of \( G(n), \tilde{G}(m), A(n), \) and \( B(m) \) requires finding proper cutoffs for infinite sums over \( i \) and \( j \) in Eqs. (10) and (11). First, we tested the convergences of these sums, since their acceleration allows setting cutoffs earlier. Several convergence rate acceleration methods were tested: the Aitken delta-squared process [31,32], the Shankstransformation [33], and the Richardson extrapolation [34,35]. None of them significantly improve the computation time. They raise the complexity of the algorithm instead.

3.3. Efficient computation of the Gaussian hypergeometric function

Next, we worked out algorithm for finding the cutoffs for sums \( A(n) \) and \( B(m) \), but a simplified procedure holds for \( G(n) \) and \( \tilde{G}(m) \). The hypergeometric terms \( \gamma^2, \gamma^0, f_i(n, i), \) and \( f_j(m, j) \) converge slowly. Function \( f_i \) is depicted in Fig. 3 (see Fig. 2 for \( \gamma(n) \)). The greater \( n \) gets, the slower the \( f_i \) converge over variable \( i \) and more terms have to be included in the summation (similarly for \( f_j \) and variable \( j \)). One may apply high, constant sum cutoffs. But even for large \( i \) (\( j \)) there exists \( n(m) \) for which the fixed range is too small.

As a solution to this problem, we applied a dynamic cutoff with an adjustable precision, e.g. \( 10^{-15} \). The summation range was divided into two intervals. In the first one, the terms \( f_i \) and \( f_j \) grow monotonically to achieve the global maximum. The second interval is infinite, and the terms slowly decrease asymptotically to 0. The value \( i = 1 \) for which \( f_i \) achieves its maximum can be found by solving the differential equation \( \frac{d f_i}{d n} = 0 \), which leads to the equation \( \psi_0(2i + 1 - n + 1) = \psi_0(n + 1) + \log R - \log T \), where \( \psi_0(\cdot) \) denotes the digamma function (the first derivative of \( \log \Gamma(\cdot) \)). The inverse of \( \psi_0(2i + 1 - n + 1) \) can be computed numerically with the Paul Fackler method [36]. A similar method may be used for finding maximum of \( f_j \).

Alternatively, the algorithm starts from \( i = \lfloor n/2 \rfloor \) \( j = \lfloor (m + 1)/2 \rfloor \), and for each \( i(j) \) evaluates \( f_i(n, i) \) \( f_j(m, i) \). It adds this term to the sum and compares \( f_i(n, i) \) to the previous term \( f_i(n, i - 1) \). If \( f_i(n, i) > f_i(n, i - 1) \), the algorithm is still in the first interval and it continues summation. Otherwise, the summation is performed over the second interval, the algorithm checks if \( f_i \) is smaller than the desired precision, and, if yes, it stops. Error is minimized by adding the terms in increasing order in the first interval. In practice, the algorithm uses logarithmic forms (they allow finer comparison of \( f_i(n, i) \) and \( f_i(n, i - 1) \)).

We compare the fixed and dynamic methods of assigning cutoffs for the sum \( A(n) \) in Table 1 (the numbers for \( B(m) \) are similar). The latter method decreases the number of terms for small \( n \) and \( m \) and achieves a good precision for large values of them. It significantly improves the speed and accuracy of the computation; for example, for \( g = 4 \), if \( n = 100 \), the summation includes 1000 terms whereas if \( n = 0 \) it includes 150 terms to get the same accuracy with relative error \( 10^{-15} \).

\( f_i \) and \( f_j \) have the same properties as \( f_i \) and \( f_j \) only they converge more slowly. Therefore, the algorithm presented is capable of finding cutoffs of the infinite hypergeometric sums required in the computation of mean values.

3.4. Summation of hypergeometric functions

Computation of the sum \( S \) in Eq. (13) involves infinite summation of the product of two Gaussian hypergeometric functions \( A(n) \) and \( B(m) \) over \( n \) and \( m \), where \( n + m \geq \sigma^2 \). The solution is to precompute the values of \( A(n) \) and \( B(m) \) with the method presented in Section 3.3 and store them in the computer memory. The next step is to perform the final summation. The cutoffs \( N \) and \( M \) over variables \( n \) and \( m \), respectively, are found separately during the precomputation stage, so that \( A(N)/A(\sigma^2) \) and \( B(M)/B(\sigma^2) \) are smaller than the desired relative computation error.

Additional speedup of computations is achieved by changing the order of summation and eliminating one of the infinite sums:

\[
S = \sum_{n=0}^{\infty} \sum_{m=0}^{n} A(n - m) \cdot B(m).
\] (14)

The computation time of this formula is polynomial: \( O(N^2) \).

Further optimization is possible by noting the fact that the hypergeometric terms \( \gamma^2, \gamma^0, f_i \) and \( f_j \) are probability distributions and \( \sum_{i=0}^{\infty} \gamma^2 = 1 \), \( \sum_{j=0}^{\infty} \gamma^2 = 1 \), \( \sum_{n=0}^{m} f_i(n, j) f_j(m, j) = 1 \), which
implies that \( \sum_{n=0}^{\infty} A(n) B(m) = 1 \). These properties allow us to rewrite Eq. (14) into a form involving a finite summation range:

\[
S = 1 - \sum'_{n=0} A(n) \sum_{m=0}^{n'} B(m). 
\tag{15}
\]

Here, the computation time is also polynomial: \( O(\sigma'^2) \).

Eq. (15) has an advantage over Eq. (14) for small values of \( \sigma' \), for which the sum intervals are relatively short and the result is obtained quickly. However, the larger the value of \( \sigma' \) is, the more terms in Eq. (15) have to be taken into account. As a result, the summation takes longer and errors accumulate significantly. At some point Eq. (14) gives more reliable results and performs faster than Eq. (15). Thus, the routine computing \( S \) decides, depending on \( \sigma' \), which method to apply.

### 4. Examples and applications

#### 4.1. Computation of MQS normalization and photon number distribution

Formulae for photon number distributions, e.g., Eq. (7) or Eq. (9), require prior computation of normalizations \( |N_m|^2 \) and \( |N_{B0}|^2 \), respectively. In principle, it is possible to take advantage of the fact that \( \sum_{k=0}^{\infty} p_0(k,l) = 1 \), and compute unnormalized distribution values for all pairs \( (k,l) \), sum them up to obtain the inverse of its normalization constant, and later renormalize the result. This method is useful in cases where all meaningful values of \( p_0(k,l) \) have to be computed anyway. However, in general, the normalization constant should be computed separately.

For the theoretical preselection, according to Eqs. (14) and (15), the normalization constants are

\[
\frac{1}{|N_m|^2} = C_g \Gamma \left( \frac{\sigma'-1}{2} \right) \gamma \sum_{i=0}^{\infty} \gamma(i-1/2) \gamma(i+1/2). 
\tag{16}
\]

\[
\frac{1}{|N_{B0}|^2} = 1 - C_g \Gamma \left( \frac{\sigma'-1}{2} \right) \gamma \sum_{i=0}^{\infty} \gamma(i+1/2) \gamma(i-1/2). 
\tag{17}
\]

For \( g = 4 \), we empirically found \( \sigma \approx 5000 \) to be a good threshold for switching from Eq. (17) to Eq. (16). According to Eq. (7), computation of \( |N_m|^2 \) directly gives \( p_g^{(B0)}(k,l) \).

\( |N_{B0}|^2 \) is computed directly with Eqs. (14) and (15). We found that for \( g = 4 \) and \( \sigma' \approx 500 \) the computation time required by Eqs. (14) and (15) is similar, and for \( \sigma' > 500 \) the former is faster. In order to compute the photon number distribution \( p_g^{(B0)}(k,l) \) we turn Eq. (9) into a simpler form:

\[
p_g^{(B0)}(k,l) = |N_{B0}|^2 \times \sum_{n=0}^{\infty} \sum_{m=0}^{n} \frac{\Gamma(n+1/2)}{\Gamma(n+1)} \frac{\Gamma(m+1/2)}{\Gamma(m+1)} f_i \left( \frac{k+n-m-1}{2}, \frac{n-m}{2} \right) \cdot f_j \left( \frac{l+m}{2}, m \right) \tag{18}\]

It is similar to Eq. (16) and is evaluated likewise. The main difference is that the sums in Eq. (18) traverse only even or odd values of \( n \) and \( m \), depending on the parity of \( k \) and \( l \). The computation time is polynomial: \( O(N^2) \), where \( N \) is the cutoff of the infinite sum over \( n \).

#### 4.2. Computation of MQS distinguishability

The distinguishability given in Eq. (5) is not optimal for computation, since it does not minimize the summation errors, and for large values of the cutoffs it is difficult to keep the coefficients of the photon number distributions in memory.

##### 4.2.1. Optimization of mathematical formulae

Eq. (5) may be optimized by a change of variables:

\[
v = 1 - 2 \sum_{k=0}^{K} \sum_{l=0}^{l-1} \sqrt{p_0(k,l)} p_0(l,k) - \sum_{k=0}^{K} p_0(k,k), 
\tag{19}
\]

where \( K \) is the cutoff of the infinite sum over variable \( k \). The computation time of Eq. (19) is polynomial, \( O(K^2) \), and the equation has the same advantages and disadvantages as Eq. (5), but requires only half of the calculations.

A more complex idea is to first divide the \( (k,l) \) plane into rectangles, compute the photon number distributions for each rectangle separately, and save the data to a disk. Next, we calculate the partial overlaps for each rectangle and sum up the contributions [37]. Our approach uses tiles of a size \( K' \times L' \) and performs a two-step summation:

\[
v = 1 - \sum_{k=0}^{K/K'} \sum_{l=0}^{L/L'} \left( \sum_{k=0}^{K/K'-1} \sum_{l=0}^{L/L'-1} \sqrt{p_0(x,y)} p_0(y,x) \right),
\tag{20}\]

where \( x = K'k + K', y = L'l + L', \) and \( K \) and \( L \) are the cutoffs of the sums over \( k \) and \( l \), respectively. The computation complexity of Eq. (20) is the same as that of Eq. (5), but this method offers several advantages. The values of \( K' \) and \( L' \) can be selected in such a way that the precomputed values of coefficients (e.g. \( A(n) \) and \( B(m) \)) are stored in memory and are used for all points of the single tile. Additionally, different tiles can be computed in parallel with separate processors. This significantly speeds up the
4.2.2. Approximation: cutoffs of infinite sums

The plot of Fig. 4. shows three main intervals of \( k \) where the photon number distribution behaves differently.

calculations and leads to a better usage of the computer resources and minimizes errors by grouping summed terms in smaller sets, but it requires proper estimation of the cutoffs \( K \) and \( L \) and of the tile size \( K' \times L' \). The computation gets half as long if Eq. (20) is rewritten in the following form:

\[
v = 1 - 2 \sum_{k=0}^{K' \backslash K} \sum_{l=0}^{K' \backslash l} \left( \sum_{k'=0}^{K'-1} \sum_{l'=0}^{L'-1} \sqrt{p_{\Phi}(x, w) p_{\Phi}(w, x)} \right) \]

\[
- \sum_{k=0}^{K'} \sum_{l=0}^{L'} p_{\Phi}(x, w) + 2 \sum_{l=0}^{L'-1} \sqrt{p_{\Phi}(x, w) p_{\Phi}(w, x)}.
\]

where \( w = K' + l' \).

4.2.2. Approximation: cutoffs of infinite sums

In order to estimate the cutoffs \( K \) and \( L \) in Eqs. (20) and (21), we tried the method presented in Section 3.3. However, unlike in the computation of \( A(n) \) and \( B(m) \), Eq. (20) involves a lot of near-zero terms for small \( k \) and \( l \) before entering the more significant regions, and we had to adjust this algorithm accordingly. The new solution analyzes the shape of photon number distributions. The distribution cut for a given \( l \) spans three different intervals of \( k \); see Fig. 4. The first interval (A) starts at \( k = 0 \) and ends at \( k \approx 10 \sigma \) for the beam-splitter preselection (10 \( \sigma \) for the theoretical). Here, the values of the distributions are small and their input is negligible. The interval \( B \) for \( k \in (10 \sigma', 30m) \) \( (k \in (10 \sigma, 20m)) \) gives the main input. In region C, the values are slowly fading out for \( k \) going to infinity.

The proposed algorithm analyzes the values of \( p_{\Phi}(k, 0) \) to find the start of region C. To speed up the search, it begins at \( k = 30m \) for the beam-splitter preselection \( (k = 20m \) for the theoretical), which is the approximate end of region B, and takes samples of the photon number distribution for \( k = 30m, 30m + 100, 30m + 200, \ldots \) to the point where the values of the distribution are not distinguishable from 0 with the desired precision. This point lies inside region C and can be used as a good cutoff values \( K \) and \( L \). This method gives results which confirm the values obtained analytically.

4.3. Simulation of finite detector resolution

For the theoretical preselection, we model the resolution of detectors measuring photon number \( \pm 150 \) with the Weierstrass transform, i.e. we apply a low-pass filter with a Gaussian distribution for which \( 3\sigma \approx 150 \). The photon number distribution \( p_{\Phi}^{(lb)}(k, l) \) takes the form

\[
p_{\Phi}^{(lb)}(k, l) = \frac{1}{2\sqrt{\pi} \sigma} \sum_{q=-3\delta}^{3\delta} \sum_{p=-q}^{p} p_{\Phi}^{(lb)}(k - p, l - q) e^{-\left(\frac{q^2 + p^2}{2\sigma^2}\right)}.
\]

where \( \delta \) is the standard deviation. The Gaussian properties of the filter allow for replacing the infinite summation over \( p \) and \( q \) with finite ranges.

In case of the beam-splitter preselection, applying a Gaussian filter to the photon number distribution values does not influence the result in a significant way. Since the filtering slows the computations, which in the case of the beam-splitter preselection are already time consuming, there is no reason for the Weierstrass transform.

Application of Eqs. (22) to (21) means that each tile of size \( K' \times L' \) has to be increased by a 3\sigma margin. The computation time for each tile is increased approximately \( (1 + 6\delta/K')^2 \) times and requires more computer memory, but is still quite easily tractable.

5. MQSVIS program suite

MQVIS is a program suite developed for the computation of various quantum indicators for MQS of light. These indicators include:

- photon number distributions for MQS of light,
- visibility,
- visibility computed from the overlap,
- visibility computed for the photon number distributions transformed with the Weierstrass transform for different values of \( 3\sigma \in \{1, 1.5, 15, 150\} \) (Gaussian blur),
- mean number of photons in both polarizations (jointly and separately for each polarization),
- variance of number of photons in both polarizations (jointly and separately for each polarization).

5.1. Compilation of the programs

In order to compile and test the MQVIS program suite for Linux/Unix operating systems with GNU utilities (GNU make, GNU compiler collection) it is enough to run the following commands in a directory containing unpacked source code:

```
make
make check
```

In the case of Linux/Unix with GNU make and an Intel C compiler, one has to modify Makefile, replacing gcc with icc and -fopenmp option with -openmp.

For other platforms and compilers, one should compile mqsvis_norm.c with a standard C compiler, and mqsvis_tile.c with a C compiler and OpenMP extensions turned on. If the compiler does not offer OpenMP extensions, the program will still work but will not utilize multiple cores or processors.

5.2. Description of the program suite

5.2.1. MQVIS_norm

This program computes squared normalization for preselected MQSs of light. This value is required by the MQVIS_tile program.

Program parameters:
- \( m \)—average number of photons, \( Dth \)—preselection threshold.

5.2.2. MQVIS_tile

This program computes the photon number distribution for preselected MQSs of light for a single tile. It produces partial results of quantum indicators: visibility, mean number of photons, variance, to be gathered by the MQVIS_gather script. Additionally, it saves the photon number distribution for a single MQS state to a file.

Program parameters:
- \( m \)—average number of photons, \( Dth \)—preselection threshold, \( R \) — amount of losses, \( N2 \) — squared normalization, computed by the MQVIS_norm program, \( tilesize \)—width and height of a single tile, \( tile \)—tile column (counted from 0), \( tiley \)—tile row (counted from 0), \( plotstep \)—step used for saving photon number distributions, \( plotfname1 \)—file to save photon number distributions.
distribution for tile (tilex, tiley), plotfname2—file to save photon number distribution for tile (tilex, tiley).

Environment variables (used only when compiled with OpenMP extensions): OMP_NUM_THREADS—maximum number of cores or processors utilized by the program.

5.2.3. MQVIS_gather.py (Python script)

This is a script developed to gather partial results computed by the MQVIS_tile program. It computes the final quantum indicators for MQSs of light.

The program requires partial results of the MQVIS_tile program to be saved in text files with the names in the form

\[ \text{MQS}\_\text{Dth}\_\text{R} \_\text{R} \_\text{R} \_\text{d} \_\text{d} \_\text{d} \_\text{d} \_\text{d} \_\text{d} . \text{txt} \]

where the first %s is replaced with the average number of photons, the second %s with the preselection threshold, the third %s with losses, and %d, %d are comma-separated tile coordinates numbered from 0.

Program parameters: \( m \)—average number of photons, \( Dth \)—preselection threshold, \( R \)—losses, \( tiles \)—number of tiles in a row/column.

5.3. Examples and testing

Below we present a sample session with the above programs. The sequence of commands could be used for example for testing the suite.

1. Compute normalization for \( m = 5 \) and \( Dth = 2 \) and save it to a file

\[ \text{mqsvis_norm 5 2 > normalization.txt} \]

2. Compute tile (0, 0) for \( R = 0 \) (no losses), tile size \( 10 \times 10 \)

\[ \text{mqsvis_tile 5 2 0 $(< normalization.txt) 10 0 0 1 \ \text{plot-0,0.txt /dev/null > M5_Dth2_r0-0,0.txt} \]

3. Compute tile (1, 0), and simultaneously tile (0, 1) for the same parameters

\[ \text{mqsvis_tile 5 2 0 $(< normalization.txt) 10 1 0 1 \ \text{plot-1,0.txt /dev/null > M5_Dth2_r0-1,0.txt} \]

4. Compute tile (1, 1)

\[ \text{mqsvis_tile 5 2 0 $(< normalization.txt) 10 1 1 \ \text{plot-1,1.txt /dev/null > M5_Dth2_r0-1,1.txt} \]

5. Gather the results

\[ \text{python mqsvis_gather.py 5 2 0} \]

The computed results for the above session, printed out in the last step, are as follows:

- total probability sum=0.626948016783147
- visibility=0.683
- visibility prom=0.683
- simple visibility computed from the overlap=1.000
- visibility with Gaussian blur (3sigma=1)=0.968
- visibility with Gaussian blur (3sigma=1.5)=0.746
- visibility with Gaussian blur (3sigma=2)=0.491
- visibility with Gaussian blur (3sigma=3)=0.912
- mean=8.362
- mean k=6.269
- mean l=2.092
- variance=68.303
- variance k=40.241
- variance l=15.715
- maximal value=0.0408525598455265

6. Conclusions

We have discussed numerical methods that are useful for the modeling of macroscopic quantum superpositions of light generated by a phase-covariant quantum cloner. The properties of these MQSs are governed by a Gaussian hypergeometric function, whose parameters are half-integers and whose argument takes moderate values. No simplifications of this function are possible. It is given by a slowly convergent infinite sum, which is impossible to accelerate. The algorithm dynamically finds the cutoff as well as the cutoffs of sums involving hypergeometric functions. We achieved a precision of \( 10^{-10} \).

The model allows the simulation of experimental schemes involving linear optical elements and MQSs, as well as the computation of the expectation values of some observables. It is not possible to obtain such results by analytical calculation. Our model takes into account the values of the experimental parameters. Depending on them, it optimizes the computation by choice of the more efficient parallelizable algorithm for evaluation of the indicators for these states, e.g. the photon number distribution or distinguishability. The numerical methods presented here can be adjusted for analysis of decoherence and similar experimental schemes, different filtering techniques, and modified MQSs. As an example, we included a model of a realistic detector which proved the applicability of our model for scientific research. We verified our numerical results with those from a recently developed analytical model for MQS, available only for certain regimes of parameter values [14].

Finally, the MQVIS program suite was developed and implemented in C and Python programming languages. It utilizes the developed numerical model and allows one to compute various quantum indicators for MQSs of light: photon number distributions, visibility in case of limited resolution of photodetectors, and the mean and variance of the number of photons (jointly and separately for each polarization). Parallelization was achieved by splitting the computing tasks into tiles and utilization of OpenMP compiler extensions.

Acknowledgments

A.B. thanks J. Arabas and R.W. Chhajlany for discussions. Calculations were carried out at CI TASK (Galera cluster) and Cyfronet (Zeus cluster). This work was partially supported by the Polish Ministry of Science and Higher Education Grant No. 2619/B/H03/2010/38.

References

[37] This approach has been discussed with R.W. Chhajlany.