Quantum Information Processing (2021) 20:91
https://doi.org/10.1007/s11128-020-02976-z

Mermin polynomials for non-locality and entanglement detection in Grover's algorithm and Quantum Fourier Transform

Henri de Boutray[1,2] · Hamza Jaffali[1,2] · Frédéric Holweck[1,3] · Alain Giorgetti[1,2] · Pierre-Alain Masson[1,2]

Received: 16 January 2020 / Accepted: 18 December 2020 / Published online: 5 March 2021

Abstract
The non-locality and thus the presence of entanglement of a quantum system can be detected using Mermin polynomials. This gives us a means to study non-locality evolution during the execution of quantum algorithms. We first consider Grover's quantum search algorithm, noticing that states during the execution of the algorithm reach a maximum for an entanglement measure when close to a predetermined state, which allows us to search for a single optimal Mermin operator and use it to evaluate non-locality through the whole execution of Grover's algorithm. Then the Quantum Fourier Transform is also studied with Mermin polynomials. A different optimal Mermin operator is searched for at each execution step, since in this case nothing hints us at finding a predetermined state maximally violating the Mermin inequality. The results for the Quantum Fourier Transform are compared to results from a previous study of entanglement with Cayley hyperdeterminant. All our computations can be repeated thanks to a structured and documented open-source code that we provide.

Keywords Mermin polynomials · MABK violation · Quantum programs · entanglement · Non-locality · Grover's quantum search algorithm · Quantum Fourier Transform

✉ Henri de Boutray
henri.de_boutray@univ-fcomte.fr

1 Univ. Bourgogne Franche-Comté (UBFC), Besançon, France

2 Institut FEMTO-ST (UMR 6174 - CNRS/UBFC/UFC/ENSMM/UTBM), Besançon, France

3 Laboratoire Interdisciplinaire Carnot de Bourgogne (ICB, UMR 6303 - CNRS/UB/UTBM), Dijon, France

1 Introduction

Quantum entanglement has been identified as a key ingredient in the speed-up of quantum algorithms [18], when compared to their classical counterparts. Our work is in line with previous work on a deeper understanding of the role of entanglement in this speed-up [6,7,10,19].

We focus on Grover's algorithm [11] and the Quantum Fourier Transform (QFT) [25, Chap. II-Sec. 5] which plays a key role in Shor's algorithm [29]. We choose these two examples because they both provide quantum speed-up (quadratic for Grover's algorithm and exponential for the QFT) and are well understood and described in the literature [25]. Previous work tackled entanglement in Grover's algorithm and the QFT from two perspectives: quantitatively, with the Geometric Measure of Entanglement (GME) [4,28,33], separately for Grover's algorithm [27] and the QFT [30], and qualitatively, by observing the different entanglement SLOCC classes traversed by an execution, for both algorithms [17].

Instead of directly measuring entanglement we use Mermin polynomials [1,2,22] to demonstrate the non-locality (breaking of an upper bound holding for all classical states) of some states generated by these algorithms. Knowing that a state exhibits non-local properties allows us to conclude that the state is entangled. In this respect one uses Mermin polynomials as entanglement witnesses as suggested in [12,31]. Batle et al. [5] previously investigated non-local properties during Grover's algorithm using Mermin polynomials. However they concluded to the absence of non-locality. In the present work we setup the Mermin polynomials in such a way that we exhibit, on the contrary, violation of the classical inequalities in Grover's algorithm. Moreover our evaluation techniques are more efficient, allowing us to reach 12 qubits. We also exhibit non-locality during the QFT in the context of Shor's algorithm.

An initial motivation of this study is the verification of quantum programs. Turning a quantum algorithm into an implementation for a quantum computer with scarce resources often requires highly non-trivial optimizations, which may introduce bugs in the resulting programs. Checking state properties is a way to gain more confidence in these implementations. In the present paper we investigate non-locality as a property of entangled quantum states that could be checked for a quantum algorithm and its implementations. In this respect evaluation of Mermin polynomials is of particular interest: violation of the classical bound has a physical meaning and the evaluation of Mermin polynomials can be implemented on a quantum computer, as it was demonstrated by Alsina et al. [2].

In this paper we make two different uses of Mermin polynomials. In our study of Grover's algorithm we build for each number of qubits a specific Mermin polynomial which achieves maximal violation for the quantum state of highest GME that Grover's algorithm is meant to approach during its execution. Doing so we will not only show that the states generated by the algorithm violate the classical bound but also that the valuations of this specific Mermin polynomial behave similarly to the GME. In our study of the QFT, we propose a different approach by choosing at each step of the algorithm a Mermin polynomial whose valuation is maximal for the given state. We show that this quantity is a local unitary invariant that can be compared to other invariants. In the context of Shor's algorithm for four qubits, we also obtain violation of

the Bell-like Mermin inequalities (also called MABK in the literature) during the QFT part of the algorithm. This amount of violation is not constant during the QFT, which shows a qualitative change of the nature of entanglement involved. This differs from the quantitative results obtained with the Groverian's measure of entanglement [30] for which it was proved that the amount of entanglement is nearly constant in Shor's algorithm during the QFT. Without being contradictory the present work illustrates the fact that non-equivalent classes of entanglement under local unitary transformations are achieved during the QFT part of Shor's algorithm, as it was shown in [17].

The paper is organized as follows. After Sect. 2 presenting some background on Grover's algorithm, the QFT and Mermin polynomials, Sect. 3 presents our method and results concerning the detection of entanglement in Grover's algorithm and the QFT. In particular we exhibit Mermin inequalities violations in both algorithms. In this section we also compare the results obtained with the Mermin polynomials to previous results [17] using the Cayley hyperdeterminant. Finally, Sect. 4 documents the code developed for this evaluation, in order to make it reusable by anyone wishing to[1]. In addition, Appendix A recalls known properties of the states in Grover's algorithm and Appendix B recalls the definition of the Cayley hyperdeterminant.

2 Background

This paper relies on pure state formalism: each considered state is a normalized vector of the Hilbert space $\mathcal{H} = \mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2$. A *separable* state $|\varphi\rangle$ is a rank-one tensor, i.e., $|\varphi\rangle = |\varphi_1\rangle \otimes |\varphi_2\rangle \otimes \cdots \otimes |\varphi_k\rangle$, where $|\varphi_i\rangle$ are single-qubit states. A tensor/state $|\varphi\rangle$ is said to be of rank r if there are r rank-one tensors $|\varphi_i\rangle = |\varphi_1^i\rangle \otimes |\varphi_2^i\rangle \otimes \cdots \otimes |\varphi_k^i\rangle$, with $i = 1, \ldots, r$, such that $|\varphi\rangle = \sum_{i=1}^{r} \alpha_i |\varphi_i\rangle$ with $\alpha_i \in \mathbb{C}$, and r is minimal for this property. An *entangled* state is a tensor of rank higher than 1.

The remainder of this section provides necessary background to the reader, regarding Grover's algorithm (2.1), some properties of the states during its execution (2.2), the Quantum Fourier Transform (2.3) and the Mermin operators (2.4).

2.1 Grover's algorithm

We summarize here Grover's algorithm, widely described in the literature ([11,20] and [25, chapter 6]).

Grover's algorithm aims to find objects satisfying a given condition in an unsorted database of 2^n objects, *i.e.* to solve the following problem.

Given a positive integer n, $N = 2^n$, $\Omega = \{0, \ldots, N-1\}$ and the characteristic function $f : \Omega \to \{0, 1\}$ of some subset S of Ω ($f(x) = 1$ iff $x \in S$), find in Ω an element of S only by applying f to some elements of Ω.

Grover's algorithm provides a quadratic speedup over its classical counterparts. Indeed, assuming that each application of f is done in one step, it runs in $\mathcal{O}(\sqrt{N})$ instead of $\mathcal{O}(N)$.

[1] The source code is available at https://quantcert.github.io/Mermin-eval.

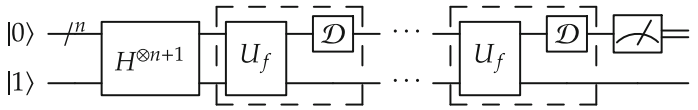


Fig. 1 Grover's algorithm in circuit formalism

Figure 1 shows this algorithm as a circuit composed of several gates that we now describe. $H^{\otimes n+1}$ is simply the Hadamard gate on each wire. When applied on the n first registers initialized at $|0\rangle$, it computes the superposition of all states, *i.e.*,

$$H^{\otimes n}|0\rangle = \frac{1}{\sqrt{N}} \sum_{\mathbf{x}=0}^{N-1} |\mathbf{x}\rangle.$$

After $H^{\otimes n+1}$, the dashed box (hereafter called $\mathcal{L}$) is repeated $k_{opt} = \left\lfloor \frac{\pi}{4}\sqrt{\frac{N}{|S|}} \right\rfloor$ times.

The circuit $\mathcal{L}$ is composed of the *oracle* U_f and the *diffusion operator* $\mathcal{D}$. The gate U_f computes the classical function f. It has the following effect on states:

$$\forall(\mathbf{x}, y) \in \{0, \ldots, N\} \times \{0, 1\},\ U_f\,(|\mathbf{x}\rangle \otimes |y\rangle) = |\mathbf{x}\rangle \otimes |y \oplus f(\mathbf{x})\rangle.$$

On the circuit of Fig. 1 one can show that the last register remains unchanged when applying the U_f gate. Indeed after the Hadamard gate H, this last register becomes $H|1\rangle = \dfrac{|0\rangle - |1\rangle}{\sqrt{2}}$. Now consider a state $|\mathbf{x}\rangle \otimes \dfrac{|0\rangle - |1\rangle}{\sqrt{2}}$. Then

$$U_f\ \left(|\mathbf{x}\rangle \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) = \begin{cases} |\mathbf{x}\rangle \otimes \dfrac{|1\rangle - |0\rangle}{\sqrt{2}} & \text{if } f(\mathbf{x}) = 1 \\ |\mathbf{x}\rangle \otimes \dfrac{|0\rangle - |1\rangle}{\sqrt{2}} & \text{otherwise.} \end{cases}$$

In other words,

$$U_f\ \left(|\mathbf{x}\rangle \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) = (-1)^{f(\mathbf{x})}\ \left(|\mathbf{x}\rangle \otimes \frac{|0\rangle - |1\rangle}{\sqrt{|2\rangle}}\right).$$

One says that the oracle U_f marks the solutions of the problem by changing their phase to -1. To emphasize this, we adopt the usual convention which consists of ignoring the last register and considering that U_f has the following effect:

$$\begin{cases} U_f\ |\mathbf{x}\rangle = -|\mathbf{x}\rangle, \forall \mathbf{x} \in S \\ U_f\ |\mathbf{x}\rangle = |\mathbf{x}\rangle, \forall \mathbf{x} \notin S \end{cases}.$$

The diffusion operator $\mathcal{D} = 2(|+\rangle\langle+|)^{\otimes n} - I_{2^n}$ performs the inversion about the mean. Indeed if $|\varphi\rangle = \sum_{\mathbf{i}=0}^{N-1} \alpha_{\mathbf{i}}|\mathbf{i}\rangle$ and $\bar{\alpha} = \frac{1}{N}\sum_{\mathbf{i}=0}^{N-1} \alpha_{\mathbf{i}}$ denotes the mean value of the amplitudes of $|\varphi\rangle$, then $\mathcal{D}\,|\varphi\rangle = \sum_{\mathbf{i}=0}^{N-1} \alpha'_{\mathbf{i}}|\mathbf{i}\rangle$ with $\alpha'_{\mathbf{i}} - \bar{\alpha} = \bar{\alpha} - \alpha_{\mathbf{i}}$.

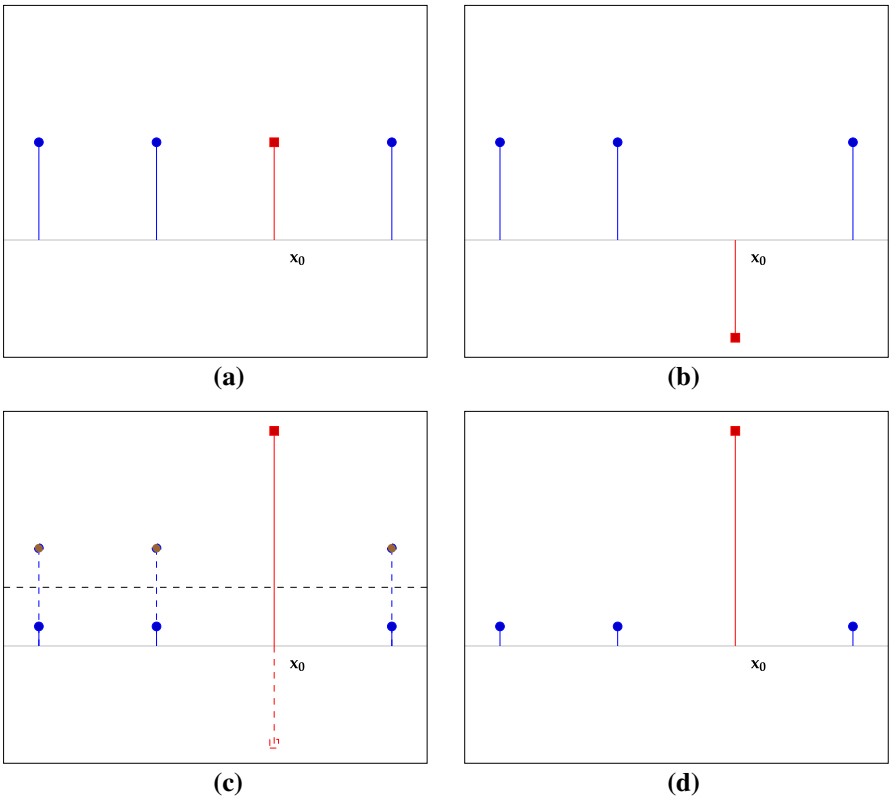


Fig. 2 First iteration of loop $\mathcal{L}$ in Grover's algorithm: the combs represent the amplitude of each element

Figure 2 provides a visualization of the effect of the beginning of the algorithm on the amplitudes of $|\varphi\rangle$. For readability purposes, only 4 amplitudes are represented, and only one element is searched ($S = \{\mathbf{x}_0\}$), shown with a square instead of a bullet. The state is initialized to $|\mathbf{0}\rangle$. The state resulting of applying $H^{\otimes n}$ is the superposition of all states $|+\rangle^{\otimes n}$ (Fig. 2a). Then the oracle U_f flips the searched element (Fig. 2b), and the diffusion operator $\mathcal{D}$ performs the inversion about the mean (Fig. 2c, d).

The final measure yields the index of an element from S with high probability.

2.2 Properties of states in Grover's algorithm

The evolution of the amplitudes of the state $|\varphi\rangle$ during the execution of the algorithm is well known [25]. If we denote by θ the real number such that $\sin(\theta/2) = \sqrt{|S|/N}$, then after k iterations (*i.e.*, after applying k times the circuit $\mathcal{L}$), the state is:

$$|\varphi_k\rangle = \alpha_k \sum_{\mathbf{x}\in S} |\mathbf{x}\rangle + \beta_k \sum_{\mathbf{x}\notin S} |\mathbf{x}\rangle \tag{1}$$

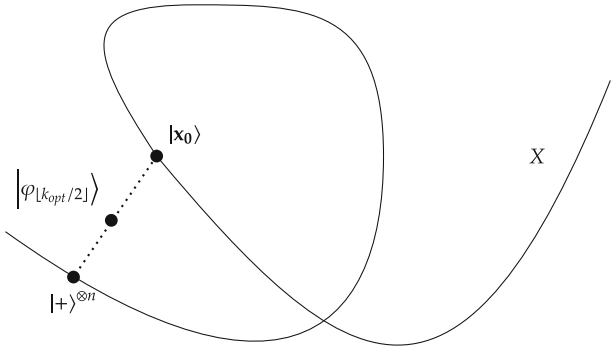


Fig. 3 States path (dotted) in relation with the variety of separable states X during Grover's algorithm execution in the space of pure states [14, Figure 2]

with $\alpha_k = \frac{1}{\sqrt{|S|}} \sin(\frac{2k+1}{2}\theta)$ and $\beta_k = \frac{1}{\sqrt{N-|S|}} \cos(\frac{2k+1}{2}\theta)$. The sequences $(\alpha_k)_k$ and $(\beta_k)_k$ are two real sequences, respectively, increasing and decreasing when k varies between 0 and $k_{opt} = \left\lfloor \frac{\pi}{4}\sqrt{\frac{N}{|S|}} \right\rfloor$.

An alternative representation of the evolution of the states during the execution of Grover's algorithm is proposed in [14]. An elementary algebra calculation (See Appendix A, Proposition 2) shows that

$$|\varphi_k\rangle = \tilde{\alpha}_k \sum_{\mathbf{x}\in S} |\mathbf{x}\rangle + \tilde{\beta}_k |+\rangle^{\otimes n} \tag{2}$$

with $\tilde{\alpha}_k = \alpha_k - \beta_k$ and $\tilde{\beta}_k = 2^{n/2}\beta_k$. The sequences $(\tilde{\alpha}_k)$ and $(\tilde{\beta}_k)$ are, respectively, increasing and decreasing on $\{0, \ldots, k_{opt}\}$ (see Appendix A, Proposition 3).

In particular, if one considers the case of one searched element $|\mathbf{x}_0\rangle$, *i.e.* $S = \{\mathbf{x}_0\}$, then Equation (2) becomes

$$|\varphi_k\rangle = \tilde{\alpha}_k |\mathbf{x}_0\rangle + \tilde{\beta}_k |+\rangle^{\otimes n}. \tag{3}$$

Figure 3 provides a graphical interpretation of Equation (3). The "curve" X represents the variety (set defined by algebraic equations) of separable states. This figure illustrates the fact that during the execution of Grover's algorithm, the quantum state $|\varphi_k\rangle$ evolves as follows: it starts from the separable state $|+\rangle^{\otimes n}$ and moves on the dotted secant line until it gets close to the searched state $|x_0\rangle$ when $k = k_{opt}$. All states on the secant line are entangled (rank-two tensors).

In [14], it is proven that for states in superposition $\alpha|\mathbf{x}_0\rangle + \beta|+\rangle^{\otimes n}$ with $\alpha, \beta \in \mathbb{R}_+$, the GME is maximal when $\alpha = \beta$. Let $|\varphi_{ent}\rangle$ hereafter denote the state $(|\mathbf{x}_0\rangle + |+\rangle^{\otimes n})/K$, normalized with the factor K. Figure 3 suggests that the search should come close to the state $|\varphi_{ent}\rangle$, around the step $k_{opt}/2$. Thus, a maximum of entanglement is expected close to this pivot step.

2.3 Quantum Fourier Transform (QFT)

The quantum analogous of the Discrete Fourier Transform (DFT) is the Quantum Fourier Transform (QFT). It acts linearly on quantum registers and is a key step in Shor's algorithm, permitting to reveal the period of the function defining the factorization problem [25,29].

In the context of Shor's algorithm, the QFT is used to transform a periodic state into another one to obtain its period. The periodic state $|\varphi^{l,r}\rangle$ of n qubits with shift l and period r is defined by

$$|\varphi^{l,r}\rangle = \frac{1}{\sqrt{A}} \sum_{i=0}^{A-1} |l + ir\rangle \quad \text{with } A = \left\lceil \frac{N-l}{r} \right\rceil \text{ and } N = 2^n,$$

for $0 \leq l \leq N-1$ and $1 \leq r \leq N-l-1$ [30, Eq. 5].

For example, for the periodic 4-qubit states, with shift $l = 1$ and period $r = 5$, there are $A = \lceil \frac{16-1}{5} \rceil = 3$ basis elements, so:

$$|\varphi^{1,5}\rangle = \frac{1}{\sqrt{3}}\big(|\mathbf{1}\rangle + |\mathbf{6}\rangle + |\mathbf{11}\rangle\big) = \frac{1}{\sqrt{3}}\big(|0001\rangle + |0110\rangle + |1011\rangle\big)\,.$$

When applied to one of the computational basis states $|k\rangle \in \{|0\rangle, |1\rangle, \ldots, |N-1\rangle\}$ (expressed here in decimal notation), the result of the QFT can be expressed by

$$QFT\ |k\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \omega^{kj} |j\rangle,$$

where $\omega = e^{\frac{2i\pi}{N}}$ is the primitive N-th root of unity. Then, for any n-qubit state $|\psi\rangle = \sum_{j=0}^{N-1} x_j |j\rangle$, we get

$$QFT\ |\psi\rangle = \sum_{k=0}^{N-1} y_k |k\rangle \quad \text{with } y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \cdot \omega^{kj}. \tag{4}$$

The corresponding matrix is

$$QFT_N = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^1 & \omega^2 & \omega^3 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{pmatrix}.$$

In the circuit representation, the QFT can be decomposed into several one-qubit or two-qubit operators. To obtain this decomposition three different kinds of gates are

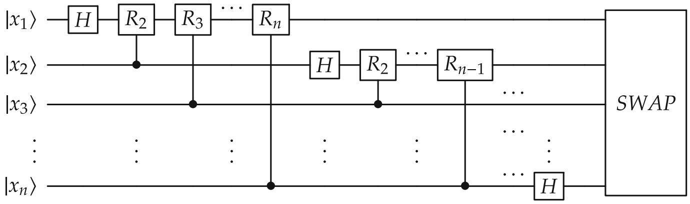
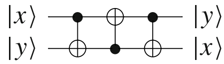


Fig. 4 Quantum circuit representation of the Quantum Fourier Transform for a n-qubit register

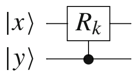
used: the Hadamard gate, the SWAP gate and the *controlled*-R_k gates, defined by the matrices and circuits

$$SWAP = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



and

$$cR_k = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{\frac{2i\pi}{2^k}} \end{pmatrix}$$



The complete circuit of the QFT is provided in Fig. 4, where the n-qubit SWAP operation consists of swapping $|x_1\rangle$ with $|x_n\rangle$, $|x_2\rangle$ with $|x_{n-1}\rangle$, and so on.

Remark 1 One of the reasons that explain the exponential speed-up in Shor's quantum algorithm, is the complexity of the QFT which is quadratic with respect to the number of registers. By comparison, classically, the complexity of the Fast Fourier Transform algorithm that computes the DFT of a vector with 2^n entries is in $\mathcal{O}(n2^n)$.

2.4 Mermin polynomials and Mermin inequalities

Entanglement variations during the execution of Grover's algorithm have been studied either by computing the evolution of the Geometric Measure of Entanglement [27,33], or by computing other measures of entanglement like the concurrence or measures based on invariants [5,14,33]. Similarly, for Shor's algorithm and in particular to study the variation of entanglement within the QFT, numerical computation of the Geometric Measure of Entanglement was carried out in [30]. Let us also mention [17] where the evolution of entanglement in Grover's and Shor's algorithms is studied qualitatively by considering the classes of entanglement reached during the execution of the algorithms.

The authors of [5] proposed to exhibit the non-local behavior of the states generated by Grover's algorithm by testing a generalization of Bell's inequalities known as Mermin's inequalities, based on Mermin polynomials [1,8].

Definition 1 (Mermin polynomials, [1]) Let $\left(a_j\right)_{j\geq 1}$ and $\left(a'_j\right)_{j\geq 1}$ be two families of one-qubit observables with eigenvalues in $\{-1,+1\}$. The *Mermin polynomial* M_n is inductively defined by:

$$\begin{cases} M_1 = a_1 \\ \forall n \geq 2,\ M_n = \frac{1}{2} M_{n-1} \otimes (a_n + a'_n) + \frac{1}{2} M'_{n-1} \otimes (a_n - a'_n) \end{cases} \tag{5}$$

where, in (5), M'_k is obtained from M_k by interchanging operators with and without the prime symbol.

Example 1 For $n = 2$, the Mermin polynomial is $M_2 = \frac{1}{2}(a_1 \otimes a_2 + a_1 \otimes a'_2 + a'_1 \otimes a_2 - a'_1 \otimes a'_2)$. The operator M_2 is, up to a factor, the CHSH operator used to prove Bell's Theorem [9].

One can note that a one-qubit observable a with eigenvalues in $\{-1,+1\}$ can be written as a normed linear combination $a = \alpha X + \beta Y + \gamma Z$ of the Pauli matrices $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$ and $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$, with the constraint $|\alpha|^2 + |\beta|^2 + |\gamma|^2 = 1$.

Mermin's inequalities

$$\langle M_n \rangle^{LR} \leq 1 \quad \text{and} \quad \langle M_n \rangle^{QM} \leq 2^{\frac{n-1}{2}} \tag{6}$$

respectively, formalize that the expectation value $\langle M_n \rangle$ of M_n is bounded by 1 under the hypothesis LR of local realism, while it is bounded by $2^{\frac{n-1}{2}}$ in quantum mechanics (QM).

The violation of the first Mermin inequality shows non-locality which is only possible under the hypothesis of quantum mechanics and if the quantum state is entangled. More precisely the maximal violation of Mermin's inequalities occurs for GHZ-like states [1,8,22], *i.e.* states equivalent to $|GHZ\rangle = \frac{1}{\sqrt{2}}(|0\rangle^{\otimes n} + |1\rangle^{\otimes n})$ by local transformations.

One of the advantages of Mermin's inequalities is that they can be tested by a physical experiment. Recently the violation of Mermin's inequalities was tested for $n \leq 5$ qubits on a small quantum computer [2].

3 Method and results

In this section we present the main results of this study, obtained by evaluating Mermin polynomials on states generated at different steps of Grover's algorithm and the QFT. As explained in the introduction our goal is to exhibit quantum properties of those states that can be experimentally checked. When it violates the classical bound, a Mermin polynomial detects entanglement – a resource that has been proved several times to appear in those algorithms. We obtain those violations in both algorithms. It is also known that the amount of violation of Mermin's inequalities is not in one-to-one correspondence with the quantity of entanglement involved [3]. The question

of measuring the quantity of entanglement is also a difficult question, as it is known that the notion of absolutely maximally entangled states does not exist already in the four-qubit case [16]. Here we compare evaluation of Mermin polynomials to different types of entanglement measures. In Grover's algorithm one uses a specific Mermin polynomial, which is fixed once for all the algorithm. By carefully choosing this polynomial one shows that its evaluation behaves like the GME. In the QFT algorithm, previous work [30] concluded to small variations of the GME. Here, by choosing differently which Mermin polynomial we evaluate at each state, we show that the entanglement classes change during the QFT, as it was already observed in [17].

Once two families $(a_j)_{1\leq j\leq n}$ and $(a'_j)_{1\leq j\leq n}$ of observables are chosen, one can define the *Mermin test function* f_{M_n} by $f_{M_n}(\varphi) = \langle\varphi|M_n|\varphi\rangle$. Inequalities (6) tell that $f_{M_n}(\varphi) > 1$ implies that $|\varphi\rangle$ is non-local. We present in this section two approaches to choose the parameters $(a_j)_{1\leq j\leq n}$ and $(a'_j)_{1\leq j\leq n}$ of M_n to satisfy the previous inequality for some states generated by the quantum algorithm of choice.

The first approach evaluates each state that the algorithm goes through with the same function f_{M_n}, with a unique polynomial M_n chosen prior to state computation. This approach has the advantage of providing a fast calculation ($(a_j)_{1\leq j\leq n}$ and $(a'_j)_{1\leq j\leq n}$ are computed only once), but the function f_{M_n} is not a measure of entanglement, since it is not invariant by local unitary transformations, *i.e.*, we do not have $f_{M_n}(\varphi) = f_{M_n}(g.\varphi)$ for all transformations $g \in LU = U_2(\mathbb{C})^n$ and all quantum states $|\varphi\rangle$ ($|g.\varphi\rangle$ is defined as such: for $g = (g_1, \ldots, g_n)$ and $G = g_1 \otimes \ldots \otimes g_n$, $|g.\varphi\rangle = G\,|\varphi\rangle$).

The second approach is to choose a different M_n for each state $|\varphi\rangle$, by optimizing $f_{M_n}(\varphi)$ for each state traversed by the algorithm. This means that we are finding values for (a_j) and (a'_j) many times for a single run. This approach was for example used in [5]. We use it in Sect. 3.2.1 to define a quantity $\mu(\varphi)$, invariant under the group LU of local unitary transformations (see Proposition 1).

3.1 Grover's algorithm properties

Hereafter we simplify the calculations by taking $S = \{\mathbf{x}_0\}$, *i.e.*, by considering that Grover's algorithm is only searching for a single element $\mathbf{x}_0$. We want to show two properties:

1. Grover's algorithm exhibits non-locality.
2. Parameters of the Mermin test function can be computed so that the function values increase and then decrease for the successive states $|\varphi_k\rangle$ in Grover's algorithm. The maximum is reached at an integer k_{max} in $\{\lfloor k_{opt}/2\rfloor, \lceil k_{opt}/2\rceil\}$.

Property **1** is in contradiction with [5], a detailed explanation is given in Remark 2. Property **2** makes the chosen Mermin test function behave like the Geometric Measure of Entanglement.

Next section details the method we followed for finding a good Mermin polynomial establishing these properties.

3.1.1 Method

The definition of Mermin polynomials provides degrees of freedom in the choice of $(a_j)_{j\geq 1}$ and $(a'_j)_{j\geq 1}$ (an infinite number of parameters). We reduce that choice by imposing that the two sequences $(a_j)_{j\geq 1}$ and $(a'_j)_{j\geq 1}$ are constant, *i.e.* $\forall j, a_j = a$ and $a'_j = a'$. This restriction strongly reduces calculations, and it will be sufficient to achieve our objectives.

Let us denote by a and a' the two one-qubit observables that will be used to write our Mermin polynomial. We have $a = \alpha X + \beta Y + \gamma Z$ and $a' = \alpha' X + \beta' Y + \gamma' Z$ with the constraints $|\alpha|^2 + |\beta|^2 + |\gamma|^2 = 1$ and $|\alpha'|^2 + |\beta'|^2 + |\gamma'|^2 = 1$.

The degrees of freedom are the 6 complex numbers α, β, δ, α', β' and δ' with the two normalization constraints. Let $A = (\alpha, \beta, \delta, \alpha', \beta', \delta')$ be the six-tuple of these variables.

In order to satisfy Property **2**, we search for a six-tuple of parameters A such that f_{M_n} reaches its maximum for the state $\varphi_{k_{opt}/2}$. We also would like this choice of A to be independent of the states generated by the algorithm. According to the geometric interpretation presented in Sect. 2.2, the state $\varphi_{k_{opt}/2}$ should tend to the state $|\varphi_{ent}\rangle = \frac{1}{K}(|\mathbf{x}_0\rangle + |+\rangle^{\otimes n})$ when n tends to infinity (the approximation improves as n increases). Moreover the state $|\varphi_{ent}\rangle$ is a tensor of rank two with an overlap $\langle \mathbf{x}_0|+\rangle^{\otimes n} = 1/\sqrt{2^n}$ between the states $|\mathbf{x}_0\rangle$ and $|+\rangle^{\otimes n}$ which tends to 0 as n increases, *i.e.*, we expect the state $|\varphi_{ent}\rangle$ to behave like a GHZ-like state when n is large (by definition the GHZ state is SLOCC equivalent to any non-biseparable rank-two tensor). This point is important because GHZ-like states are the ones that maximize the violation of classical inequalities by Mermin polynomials [1,8,22]. Therefore by choosing a tuple of parameters A maximizing $f_{M_n}(\varphi_{ent})$ we expect to satisfy Properties **1.** and **2.**.

We use a random walk in $\mathbb{R}^6$ to maximize $f_{M_n}(\varphi_{ent})$. We operate the walk for a fixed number of steps, starting from an arbitrary point. At each step, we choose a random direction, and move toward it to a new point. If the value of $f_{M_n}(\varphi_{ent})$ at that new point is higher than at the previous one, then that point is the start point for the next step, otherwise a new point is chosen.

Once the proper coefficient for M_n found, we compute the values of each $f_{M_n}(\varphi_k)$ for k in $\{0, \ldots, k_{opt}\}$ to validate Properties **1.** and **2.**.

Example 2 When searching the state $|0000\rangle$, the highest value of $f_{M_4}(\varphi_{ent})$ obtained by this random walk is for $A = (-0.7, -0.3, -0.7, -0.5, 0.7, -0.5)$. Then, A is used to compute M_4, and then $f_{M_4}(\varphi_k)$, $\forall k \in \{0, \ldots, k_{opt}\}$.

Remark 2 Some comments are in order at this point to compare our approach with the work of [5]. First in [5] all calculations are done using the density matrices formalism instead of the vector/tensor approach we use here. But this difference is meaningless, because we are only considering pure states, so, every computation in the density matrix formalism can be done equivalently within the vector state formalism. Moreover in [5] the optimization is done at each step of the algorithm with respect to the state computed by the algorithm, while we compute the parameters only once with respect to a targeted state $|\varphi_{ent}\rangle$. Finally, as mentioned at the beginning of Sect. 3.1.1, we also restrict ourselves to two operators a and a' and thus all optimizations are performed

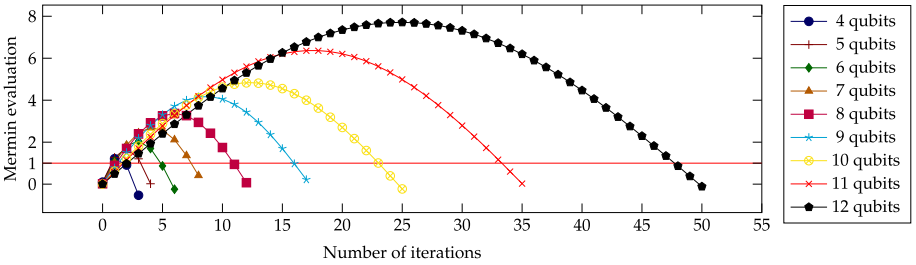


Fig. 5 Violation of Mermin's inequalities during Grover's algorithm execution for $4 \leq n \leq 12$ qubits

n	4	5	6	7	8	9	10	11	12
$\lfloor k_{opt}/2 \rfloor$	1	1	2	3	5	8	12	17	24
k_{max}	1	2	3	4	6	9	12	18	25
$f_{M_n}(\varphi_{k_{max}})$	1.21	1.72	2.05	2.69	3.37	4.17	4.83	6.36	7.71

Fig. 6 Maximums of $f_{M_n}(\varphi_k)$ for $4 \leq n \leq 12$ qubits

on six parameters instead of $6n$. This allows us to perform the calculation for a larger number of qubits (up to 12).

3.1.2 Results

Thanks to our implementation of this method in SageMath, described in Sect. 4, we obtain the values depicted in Fig. 5, for n from 4 up to 12 qubits. The searched element $\mathbf{x}_0$ is always the first element $|\mathbf{0}\rangle$ of the canonical basis, but other searched elements would give similar results, by symmetry of the problem.

The lower bound for the number n of qubits is set to 4 because for $n \leq 3$ the algorithm has no time to show any advantage, is not very reliable and does not exhibit non-locality. The upper bound is set to 12 because of technological limitations: computations for 13 qubits or more become too expensive.

We see that the two expected properties hold for all values of n: the classical limit is violated and the Mermin evaluation increases up to the middle of the executions, and then decreases (the maximal values are given in Fig. 6).

Remark 3 In [5] similar curves (Figure 3) were obtained for $n \in \{2, 4, 6, 8\}$ qubits showing the increasing-decreasing behavior, but the violation of Mermin's inequalities – the non-locality – was not established for $n = 6$ and $n = 8$, whereas it is obtained in our calculation. Recall from Remark 2 that the calculation of [5] is not exactly the same as the one performed in this paper. The curves of [5] are obtained by maximizing $f_{M_n}(\varphi_k)$ at each step of the algorithm with a larger number of parameters. Therefore as we obtain violation of Mermin's inequalities via a restricted calculation, the authors of [5] should also have observed it. We suspect errors in the implementation of the calculation of Equations (19) of [5] as we have redone this calculation for $n = 6$ based on Equations (18) and (20) of [5], and we have obtained the violation of Mermin's inequalities shown in Fig. 7.

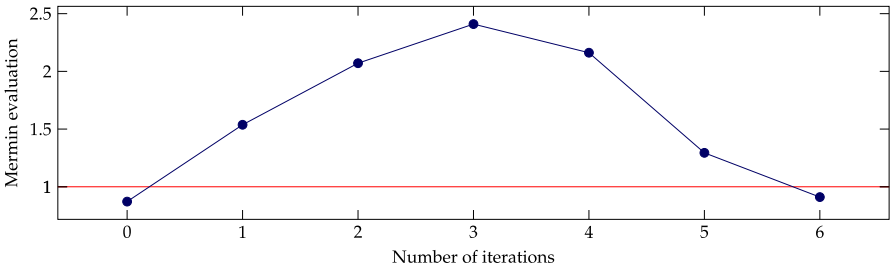


Fig. 7 Violation of Mermin's inequalities during Grover's algorithm execution for 6 qubits using [5] method

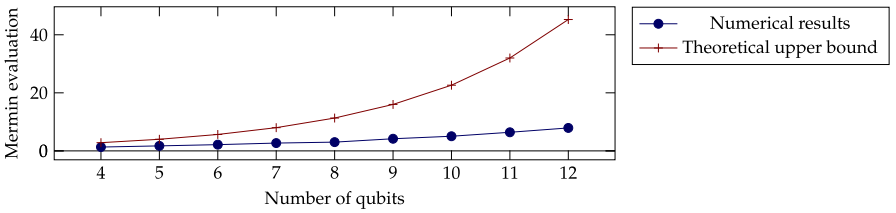


Fig. 8 Comparison between results of the computations and theoretical Mermin boundary. The curve with points as dots corresponds to the evaluation of $f_{M_n}(\varphi_{ent})$ and the curve with points as crosses corresponds to the theoretical upper bound for the violation of the Mermin inequality defined by M_n

Remark 4 The curve for $n = 12$ in Fig. 5 should be compared to the curve of Figure 1 of [27] where the evolution of the GME of the states generated by Grover's algorithm is given for $n = 12$ qubits. In our setting it is not a surprise that both curves are similar because in all of our calculations the function f_{M_n} is defined by the set of parameters that maximizes its value for $|\varphi_{ent}\rangle$. Similar behavior for other invariants in the context of Grover's algorithm have also been observed in [7,14,23].

Figure 8 provides another argument explaining why we expected violation of Mermin's inequalities in Grover's algorithm when n increases. It can be deduced from the geometric description of the algorithm (Sect. 2.2) that the quantum state $|\varphi_{\lceil k_{opt}/2 \rceil}\rangle$ should be close to $|\varphi_{ent}\rangle$ and thus behave like it with respect to the Mermin polynomial. Despite the fact that $f_{M_n}(\varphi_{ent})$ does not reach the theoretical upper bound that is obtained for states LOCC equivalent to $|GHZ_n\rangle$, one sees that the difference between $f_{M_n}(\varphi_{ent})$ and the classical bound 1 increases as a function of n.

3.2 Quantum Fourier Transform

To exhibit non-local behavior of states generated at each step of the Quantum Fourier Transform we restrict ourselves to periodic four-qubit states for the following reasons:

1. as explained in Sect. 2.3, the QFT in Shor's algorithm is applied to periodic states [25];
2. as we will see in Sect. 3.2.2 the four-qubit case is sufficient to obtain violation of Mermin's inequalities;

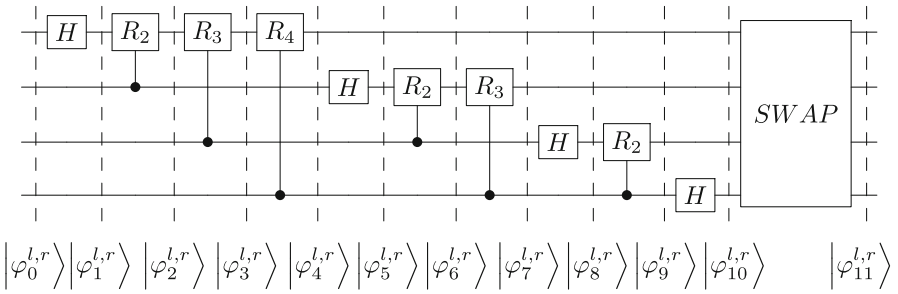


Fig. 9 Quantum circuit representation of the Quantum Fourier Transform for a 4-qubit register

3. we want to compare the present approach with a recent study of entanglement in Shor's algorithm in the four-qubit case, proposed by two of the authors of the present paper [17].

3.2.1 Method

When we apply the QFT to periodic states we have no *a priori* geometric information about the type of states that will be generated. In fact it depends on two initial parameters that define the periodic state $|\varphi^{l,r}\rangle$: its shift l and its period r. Therefore there are no reasons for restricting the choice of parameters in the calculation of $f_{M_n}(\varphi^{l,r})$. For the four-qubit case this implies that our optimization will be carried over the 24 parameters defining M_4, hereafter denoted $\alpha_1, \ldots, \alpha_{24}$ (such that $a_1 = \alpha_1 X + \alpha_2 Y + \alpha_3 Z, \ldots, a_4 = \alpha_{10} X + \alpha_{11} Y + \alpha_{12} Z, a'_1 = \alpha_{13} X + \alpha_{14} Y + \alpha_{15} Z$, …, and $a'_4 = \alpha_{22} X + \alpha_{23} Y + \alpha_{24} Z$), and this, for each state generated, in opposition to Sect. 3.1.

For $k \geq 0$, let $|\varphi^{l,r}\rangle_k$ denote the state reached after the first k gates in the QFT (Fig. 9) initialized with the periodic state $|\varphi^{l,r}\rangle$ with shift l and period r.

We are interested by the evolution of the function q defined for $k \geq 0$ by

$$q(k) = \max_{\alpha_1,\ldots,\alpha_{24}} f_{M_4}\left(\varphi_k^{l,r}\right). \tag{7}$$

In [17] two of the authors of the present paper have studied the evolution of entanglement for periodic four-qubit states through QFT by computing the absolute value of an algebraic invariant called the Cayley hyperdeterminant and denoted by Δ_{2222}. This polynomial of degree 24 in 16 variables is a well-known invariant in quantum information theory and its absolute value is known to be a measure of entanglement [13,21,24,26]. We provide the definition of Δ_{2222} in Appendix B.

Surprisingly, the two approaches, which are of different natures – an algebraic definition for the hyperdeterminant and an operator-based construction for Mermin evaluation – would sometimes present similar behavior (see Fig. 10).

In [17] it was observed that the evolution of entanglement for four-qubit periodic states through QFT shows three different behaviors with respect to Δ_{2222}.

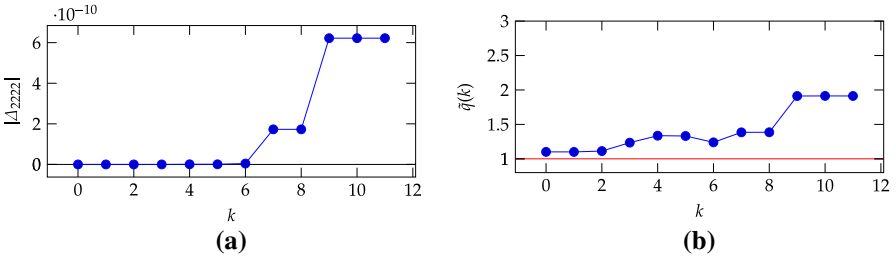


Fig. 10 Comparison of entanglement evaluation through the QFT for periodic state $(l, r) = (9, 1)$ using the measures given by the absolute value of the hyperdeterminant and the Mermin evaluation

- Case 1. The polynomial Δ_{2222} is nonzero when evaluated on $|\varphi^{l,r}\rangle$ and does not vanish during the transformation. In terms of four-qubit classification [32] it means that the transformed states remain in the so-called G_{abcd} class. This happens for $(l, r) \in \{(1, 3), (2, 3)\}$.
- Case 2. The polynomial Δ_{2222} is zero for the periodic state $|\varphi^{l,r}\rangle$ and is nonzero during the QFT. This happens for $(l, r) \in \{(0, 3), (0, 5), (2, 1), (3, 1), (3, 3), (4, 1), (4, 3), (5, 1), (5, 3), (6, 1), (6, 3), (7, 1), (9, 1)(10, 1), (11, 1), (12, 1)\}$.
- Case 3. The polynomial Δ_{2222} is zero for the periodic state $|\varphi^{l,r}\rangle$ and it remains equal to zero all along the QFT for all the other (l, r) configurations (in $\{0, \ldots, N-1\} \times \{1, \ldots, N-r\}$).

Before presenting the results let us point out that now the calculated quantity is invariant under local unitary transformations, *i.e.* under the group $LU = U_2(\mathbb{C})^n$.

Proposition 1 *Let $|\varphi\rangle \in (\mathbb{C}^2)^{\otimes n}$ be a n-qubit state and (a_i) and (a'_i) be families of one-qubit observables that define a Mermin polynomial M_n according to Definition 1. Let*

$$\mu(\varphi) = \max_{a_i, a_{i'}} \langle\varphi|M_n|\varphi\rangle. \tag{8}$$

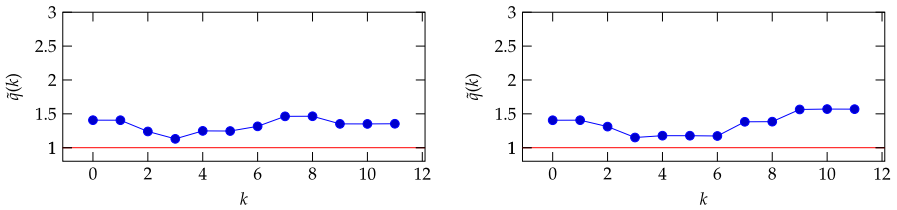
Then $\mu(\varphi)$ is LU-invariant.

Proof First one recalls that a one-qubit observable A such that $Sp(A) = \{-1, 1\}$ can always be written as $A = \alpha X + \beta Y + \gamma Z$ with $\alpha, \beta, \gamma \in \mathbb{R}$ and $\alpha^2 + \beta^2 + \gamma^2 = 1$. For the action $g.A = g^\dagger A g$ on A by conjugation with a unitary matrix $g \in U_2(\mathbb{C})$, one has $g.A = \tilde{A} = \tilde{\alpha} X + \tilde{\beta} Y + \tilde{\gamma} Z$ with $\tilde{\alpha}, \tilde{\beta}, \tilde{\gamma}$ reals such that $\tilde{\alpha}^2 + \tilde{\beta}^2 + \tilde{\gamma}^2 = 1$. Indeed $\tilde{A}$ is also a one-qubit observable such that $Sp(\tilde{A}) = \{-1, 1\}$.

Let us denote by $\lambda = (\alpha_1, \beta_1, \gamma_1, \alpha'_1, \beta'_1, \gamma'_1, \ldots, \alpha_n, \beta_n, \gamma_n, \alpha'_n, \beta'_n, \gamma'_n)$ a tuple of $6n$ parameters that define a Mermin polynomial $M_n(\lambda)$. Then

$$\mu(\varphi) = \max_{\lambda \in \mathbb{R}^{6n}, \alpha_i^2 + \beta_i^2 + \gamma_i^2 = 1, \alpha'^2_i + \beta'^2_i + \gamma'^2_i = 1} \langle\varphi|M_n(\lambda)|\varphi\rangle$$

exists, because it is the maximum of a degree n polynomial in (at most) $6n$ variables under the constraints $\alpha_i^2 + \beta_i^2 + \gamma_i^2 = 1$ and $\alpha'^2_i + \beta'^2_i + \gamma'^2_i = 1$. Let us denote by



(a) $(l, r) = (1, 3)$ **(b)** $(l, r) = (2, 3)$

Fig. 11 Evolution of the maximal values of Mermin operators in the QFT steps. Examples of input $|\varphi^{(l,r)}\rangle$ in Case 1 from [17]

λ' a tuple of parameters that maximizes $\langle\varphi|M_n(\lambda)|\varphi\rangle$, *i.e.*,

$$\mu(\varphi) = \langle\varphi|M_n(\lambda')|\varphi\rangle.$$

Let $|\psi\rangle$ be a n-qubit state LU-equivalent to $|\varphi\rangle$. Thus there exists $g = (g_1, \ldots, g_n) \in LU$ such that $|\psi\rangle = |g.\varphi\rangle = G\,|\varphi\rangle$ with $G = g_1 \otimes \ldots \otimes g_n$. Then $\langle\varphi|M_n(\lambda')|\varphi\rangle = \left(\langle\varphi|\,G^\dagger\right)\,G\,M_n(\lambda')\,G^\dagger\,\left(G\,|\varphi\rangle\right) = \langle\psi|M_n(\lambda'')|\psi\rangle$ for some tuple of parameters λ''. Therefore

$$\mu(\varphi) \leq \mu(\psi).$$

But $|\varphi\rangle = G^\dagger|\psi\rangle$ also holds, so a similar reasoning provides the inequality $\mu(\varphi) \geq \mu(\psi)$ and thus the equality. □

In the next section we plot and analyze different curves of the approximation $\tilde{q}$ of q in the four-qubit case for different choices of (l, r).

3.2.2 Results

In order to compute the values of the function q defined by (7), we optimize its parameters to approximate it and denote by $\tilde{q}$ the approximation resulting from this optimization. Curves of $\tilde{q}(k)$ are shown on Figs. 11, 12 and 13, for $k \in \{0, \ldots, 11\}$ and for different choices of shift l and period r, respectively, in Cases 1, 2 and 3.

Let us start with general comments.

- All examples in Figs. 11, 12 and 13 present violations of the Mermin inequality, and the amount of violation evolves during the algorithm. This contrasts with [30] where the authors found almost no evolution of the GME during the QFT. Those statements are not contradictory as entanglement and non-locality are not the same resource but it shows that the Mermin polynomials detect variations of the nature of the states that are not measured by the GME.
- The sets $\{0, 1\}$, $\{4, 5\}$, $\{7, 8\}$ and $\{9, 10, 11\}$ for k correspond to states before and after gates of the QFT that do not modify entanglement (Hadamard, SWAP). That explains why the function is constant on those intervals, as it was already the case for the curves $k \mapsto |\Delta_{2222}(\varphi_k^{l,r})|$ in [17].

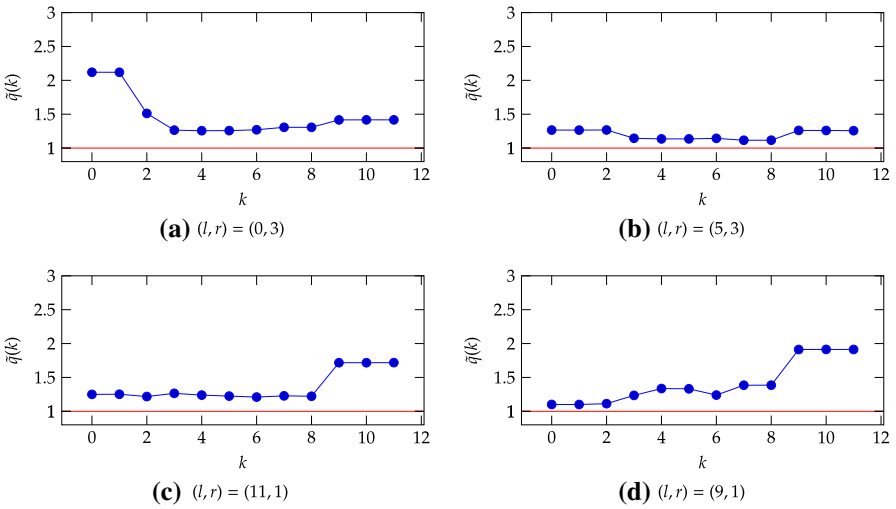


Fig. 12 Evolution of the maximal values of Mermin operators in the QFT steps. Examples of input $|\varphi^{(l,r)}\rangle$ in Case 2 from [17]

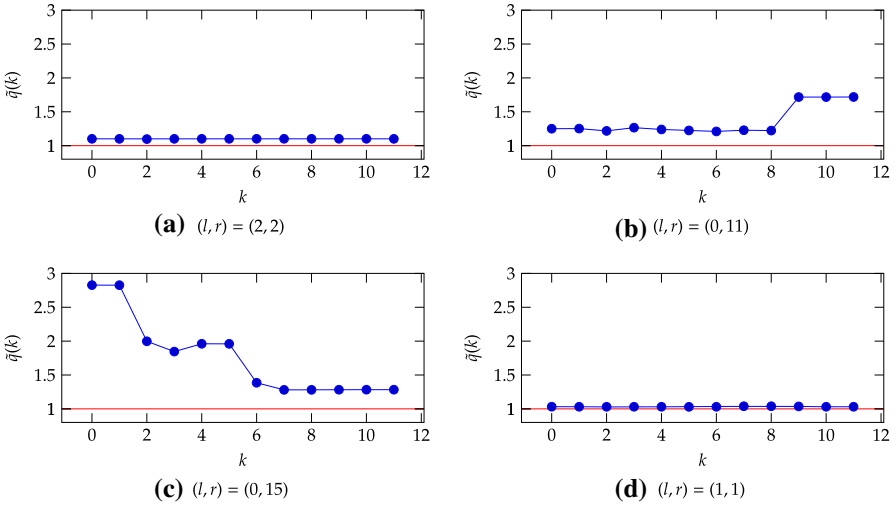


Fig. 13 Evolution of the maximal values of Mermin operators in the QFT steps. Examples of input $|\varphi^{(l,r)}\rangle$ in Case 3 from [17]

– States corresponding to Cases 1 and 2 of [17] violate the classical bound during the execution of the QFT. Only some states corresponding to Case 3 produce constant curves with some of them equal to the classical bound (not drawn). It is for instance the case for $(l, r) = (2, 4)$ which is a separable state that remains separable during the algorithm. Figure 13 illustrates different possible behaviors of the states in Case 3. These variations were not detected in [17] by the evaluation of $|\Delta_{2222}|$.

The amount of violation of non-locality measured during the QFT is not connected to the change of SLOCC classes computed in [17] for the same algorithm and input state. Indeed, states in the same SLOCC class reach different values of the maximal violation of the Mermin inequality. For instance, if one considers the periodic states $|\varphi^{l,r}\rangle$ for $(l,r) = (2,2)$ and $(l,r) = (0,11)$ (Fig. 13a, b), it is shown in [17] that these two states are SLOCC equivalent (*i.e.* can be inter-converted by a reversible local operation), but their evolution during the QFT is quite different. The value of $\tilde{q}(k)$ fluctuates around 1.10 for $(l,r) = (2,2)$, whereas it is in the interval $[1.65, 2.18]$ for $(l,r) = (0,11)$.

Similarly the cases $(l,r) = (0,15)$ and $(1,1)$ (Fig. 13c, d) correspond to two states SLOCC equivalent to $|GHZ_4\rangle$ at the beginning of the algorithm. It is clear for $(l,r) = (0,15)$ because $|\varphi^{0,15}\rangle = |GHZ_4\rangle$ and $\tilde{q}(k)$ reaches the maximal possible value at the beginning of the algorithm. The maximal violation of Mermin inequality for four qubits is $2\sqrt{2} \approx 2.81$ ($2^{\frac{n-1}{2}}$ for $n = 4$), but this value is nowhere to be approached for $(l,r) = (1,1)$ where the value of $\tilde{q}(k)$ is close to 1 at all steps of the run. In fact the state

$$|\varphi^{1,1}\rangle = \sqrt{\frac{16}{15}}|{+}{+}{+}{+}\rangle - \frac{1}{\sqrt{15}}|0000\rangle \tag{9}$$

is a state on the secant line joining $|{+}{+}{+}{+}\rangle$ and $|0000\rangle$, as described in Sect. 2.2. This state is indeed SLOCC equivalent to $|GHZ_4\rangle$ but it is closer to a separable state if one considers the GME.

4 Implementation

This section explains the code developed for this article and relates it to the notations from Sect. 2. This code can be found at https://quantcert.github.io/Mermin-eval. It uses the open-source mathematics software system SageMath[2] based on Python. The code is a module named `mermin_eval`, and usage examples can be found in the GitHub repository. Note that all the results of this article have been double checked, by first being obtained on Maple[3] and then only being generalized on SageMath.

The code is provided and presented for several reasons: so the readers can see how we obtained the results presented in Sect. 3.1.2, and they can reproduce our computations by running the code. But the code can also be extended to other evaluation methods of Grover algorithm, or adapted to other quantum algorithms, since it is structured in several well-documented functions.

This section is divided in two parts: we first explain the code used for Grover's algorithm in Sect. 4.1, and then the code used for the Quantum Fourier Transform in Sect. 4.2.

[2] http://www.sagemath.org.

[3] https://www.maplesoft.com/.

4.1 Grover's algorithm implementation

For Grover's algorithm, the main function `grover` is reproduced in Listing 1. The parameter `target_state_vector` is the searched state $|\varphi_0\rangle$. The function first executes an implementation `grover_run` of the Grover algorithm, detailed in Sect. 4.1.1, and stores in the list `end_loop_states` the states after each iteration of the loop $\mathcal{L}$. Then but independently, a call to the function `grover_optimize` (Sect. 4.1.2) optimizes Mermin operator. The result is stored in the matrix `M_opt`. Finally both these results are used to evaluate entanglement after each iteration of $\mathcal{L}$ with a call to the function `grover_evaluate` (Sect. 4.1.3), also responsible of printing the evaluations at each step.

```
def grover(target_state_vector):
  end_loop_states = grover_run(target_state_vector)

  M_opt = grover_optimize(target_state_vector)

  grover_evaluate(end_loop_states, M_opt)
```

Listing 1 Main function for Grover's entanglement study

4.1.1 Execution

The function `grover_run` given in Listing 2 takes as input the target state and returns a list of states composed of the states at the end of each loop iteration.

```
def grover_run(target_state_vector):
  layers, k_opt = grover_layers_kopt(target_state_vector)
  N = len(target_state_vector)
  V0 = vector([0, 1] + [0]*(2*N-2))

  states = run(layers, V0)
  end_loop_states = states[0]
  for i in range(k_opt):
    end_loop_states.append(states[2*i+1])

  return end_loop_states
```

Listing 2 Function running Grover's algorithm

This function operates in two steps. The first step is to build the circuit for Grover algorithm, which is achieved by the function `grover_layers_kopt`. The circuit format is a list of layers: each layer being a list of matrices (all the operations performed at a given time) and each matrix representing an operation performed on one or more wires. For example, if `H` is the Hadamard matrix, `I2` and `I4` are the identity matrix (in dimensions 2 and 4) and `X` is the first Pauli operator, then the circuit in Fig. 14 is represented by the list `[[H,I4], [X,X,I2], [I4,H], [H,H,H]]`.

The next step is to run the circuit, which is achieved by `run` that returns the list of the states after each layer. The function `run` takes as input the circuit (`layers`) and the initial state (`V0`). This function both allows us to separate syntax and semantics, and is reusable in any future context involving circuits.

The `for`-loop then filters out all the intermediate states which are not at the end of a loop iteration. For example, if we consider Grover's algorithm on three qubits

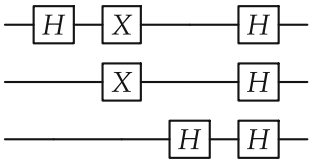


Fig. 14 Example for the circuit formalism in `grover_ent`

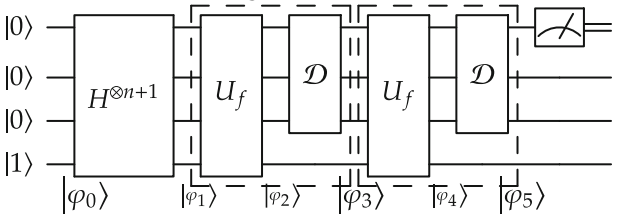


Fig. 15 End loop counting example

shown in Fig. 15, we would have the first state $|\varphi_0\rangle$, and the states $|\varphi_3\rangle$ and $|\varphi_5\rangle$ in `end_loop_states`.

This implementation of the simulation of Grover's algorithm has its limits though. It is computationally expensive to multiply matrices beyond a certain number of qubits. To push it a little further, we used another implementation for Grover's algorithm, less versatile but more efficient. This method is presented in Listing 3. In this case, two important differences are first that there is no more use for the ancilla qubit (the last wire in the circuit definition of Grover's algorithm, see Fig. 1), which divides by two the number of elements in a state vector, and second that almost no matrix multiplication is used. Indeed, the loop is now handled by functions operating directly on the state vector. The first function is `oracle_artificial`, and it only flips the correct coefficient in the running state (this is the behavior explained in Sect. 2.1). The second function `diffusion_artificial` performs the inversion about the mean.

```
def grover_run(target_state_vector):
  N = len(target_state_vector)
  n = log(N)/log(2)
  k_opt = round((pi/4)*sqrt(N))
  H = matrix(field, [[1, 1],
                     [1, -1]])/sqrt(2)
  hadamard_layer = kronecker_power(H, n)

  V0 = vector([1]+[0]*(N-1))

  V = hadamard_layer * V0
  end_loop_states = [V]

  for k in range(k_opt):
    V = oracle_artificial(target_state_vector, V)
    V = diffusion_artificial(V)
    end_loop_states.append(V)

  return end_loop_states
```

Listing 3 Optimized implementation of Grover's algorithm

4.1.2 Optimization

The `grover_optimize` function shown in Listing 4 computes an approximation of an optimal Mermin operator, as explained in Sect. 3.1.1. The Mermin operator M_n is an implicit function of $(\alpha, \beta, \delta, \alpha', \beta', \delta')$, here implemented as `(a,b,c,m,p,q)`. Because of this, optimizing the Mermin operator is finding the optimal $(\alpha, \beta, \delta, \alpha', \beta', \delta')$ for our Mermin evaluation.

```
def grover_optimize(target_state):
  n = log(len(target_state))/log(2)
  plus = vector([1,1])/sqrt(2)
  plus_n = kronecker_power(plus, n)
  phi = (target_state + plus_n).normalized()

  def M_phi(a,b,c,m,p,q):
    return M_eval(a,b,c,m,p,q, phi)

  (a,b,c,m,p,q),v = optimize(M_phi, (1,1,1,1,1,1), 5,
   10**(-2), 10**2)

  return M_from_coef(n,a,b,c,m,p,q)
```

Listing 4 Optimization function for Grover's algorithm

To optimize the Mermin operator, first the state $|\varphi_{ent}\rangle = (|\mathbf{x}_0\rangle + |+\rangle^{\otimes n})/K$ (with K the normalizing factor) is computed and stored in `phi`, then f_{M_n} represented by `M_eval` is used to define $f_{M_n}(|\varphi_{ent}\rangle)$ as `M_phi`. Note that in the mathematical notations, $f_{M_n}(|\varphi_{ent}\rangle)$ is an implicit function of $(\alpha, \beta, \delta, \alpha', \beta', \delta')$. This implicit relation is made explicit as `M_phi` is a function of `(a,b,c,m,p,q)`.

The `optimize` function takes as input a function (here `M_phi`), a first point to start the optimization from (here `(1,1,1,1,1,1)`), the step sizes bounds and a maximal number of iterations on a single step (here 10^2). The random walk starts with a step size of 5 and ends with a step size of 10^{-2}.

The optimization function proceeds with a random walk. It iterates until it finds a local maximum (for all points p in a neighborhood around the point found p_{opt}, their evaluation by the function given as the first parameter is less than the evaluation of the point found $f(p) \leq f(p_{opt})$). To find this optimum, the process starts from an arbitrary point (given as an argument) and at each step, an exploration of the space is done around the current point until the evaluation on the argument function increases. If an increase cannot be found before the fixed maximal number of iterations, the step size is reduced, otherwise the same step is repeated with the same step size. The function ends when the step size reaches the fixed minimal size of the steps.

Remark 5 This optimization can be expensive, so to speed up the calculation, a memoization step is hidden here: if `(a,b,c,m,p,q)` has already been computed for `target_state`, this result has been stored on disk at this point and is now loaded.

4.1.3 Evaluation

The function `grover_evaluate` shown in the Listing 5 is the simplest of the three: it computes $f_{M_n}(|\varphi_k\rangle) = \langle\varphi_k|M_n|\varphi_k\rangle$ for each $|\varphi_k\rangle$ in the `end_loop_states` list with M_n here being `M_opt`, and prints them.

```
def grover_evaluate(end_loop_states, M_opt):
  for state in end_loop_states:
    print((state.transpose().conjugate()*M_opt*state))
```

Listing 5 Evaluation function for Grover's algorithm

To overview the code as a whole, we can exhibit the link with Fig. 5. For this figure, each graph has been obtained by using a code line such as in Listing 6 (where `string_to_ket` is a function used to convert a string of a specific format into a vector, in this case the vector $|0000\rangle$). So, for four qubits, we set the target state as $|0000\rangle$, for five qubits as $|00000\rangle$, and so on. This is enough for symmetry reasons (searching for $|1001\rangle$ instead of $|0000\rangle$ yields similar results).

```
>>> grover(string_to_ket("0000"))
0.173154027401573
1.01189404012534
-0.469906068136016
```

Listing 6 Mermin evaluation in Grover algorithm example

4.2 Quantum Fourier Transform implementation

For the QFT, the main function `qft` is reproduced in Listing 7. The parameter `state` is the state ran through the QFT, generally a periodic state $|\varphi^{l,r}\rangle$ generated by the function `periodic_state` (Listing 8). The function `qft` first calls an implementation `qft_run` of the QFT, detailed in Sect. 4.2.1, and stores the computed states in the list `states`. Then the states are directly evaluated. The important difference compared to Grover's algorithm implementation is the fact that we are not using a separate optimization step, the optimization process is included in the evaluation process: each evaluation requires an optimization. The evaluation process is thus performed by the function `qft_evaluate` (Sect. 4.2.2), printing the evaluation as well.

```
def qft_main(state):
  states = qft_run(state)
  return qft_evaluate(states)
```

Listing 7 Main function for QFT entanglement study

```
def periodic_state(l,r,nWires):
  N = 2**nWires
  result = vector(N)
  for i in range(ceil((N-l)/r)):
    result[l+i*r] = 1
  return result.normalized()
```

Listing 8 Function used to generate the periodic state $|\varphi^{l,r}\rangle$

4.2.1 Execution

The function `qft_run` (Listing 9) uses the same circuit format as `grover_run` presented in Sect. 4.1.1. This circuit is built by `qft_layers` (Listing 10) and run by `run`. In this case however, the states do not need to be filtered, resulting in an almost trivial `qft_run` function.

```
def qft_run(state):
  layers = qft_layers(state)
  states, _ = run(layers, state)
  return states
```

Listing 9 Function running the QFT

The `qft_layers` function uses two functions not detailed here. `swap` returns a matrix corresponding to the swap of two wires `wire1` and `wire2` and the identity on the other wires concerned. The `R` method returns the controlled rotation of angle $e^{\frac{2i\pi}{2^k}}$, with the rotation being performed on the wire `target` controlled by the wire `control`. The two matrices built by these functions have a size of `2**size`. With these two functions, `qft_layers` builds the circuit for the QFT using `R` on the whole width of the circuit when a rotation is needed and using `swap` only at the end to build the global swap (in fact, `swap` is also used in `R` and that is the reason why this implementation of swap on two wires have been chosen instead of a more general arbitrary permutation gate).

```
def qft_layers(state):
  def swap(wire1,wire2,size):
    ...
  def R(k,target,control,size):
    ...
  H = matrix(field, [[1,  1],
                     [1, -1]])/sqrt(2)
  I2 = matrix.identity(field, 2)
  nWires = log(len(state))/log(2)
  layers = []

  for wire in range(nWires):
    layers.append([I2]*wire + [H] + [I2]*(nWires-wire-1))
    for k in range(2, nWires-(wire-1)):
      layers.append([R(k, wire, k+(wire-1), nWires)])

  global_swap = matrix.identity(field, 2**nWires)
  for wire in range(nWires/2):
    global_swap *= swap(wire, nWires-1-wire, nWires)
  layers.append([global_swap])

  return layers
```

Listing 10 Function building the circuit of the QFT

4.2.2 Evaluation

In this case again, the evaluation is conceptually simpler than in Grover's algorithm. Indeed, since the optimization needs to be performed for each evaluation, the result

printed at each step is simply the optimal point reached by the `optimize` function (the same as described in Sect. 4.1.2). In this case, a notable difference in the usage of `optimize` is the presence of `3*n*2` coefficients. This is explained by the fact that, this time, we do not want a trend for the evaluation's evolution and a "good enough" M_n. This means that we do not stand satisfied by the constant $a_n = \alpha X + \beta Y + \delta Z$ but we have α, β and δ variable as explained in 3.2.1 (where they become $(\alpha_i)_{1 \leq i \leq 6n}$).

Because of this, the function `M_func` (Listing 11) we optimize is now calling `M_eval_all` instead of `M_eval`. The difference is that `M_eval` took only 3×2 coefficients to compute M_n with fixed $a_i = \alpha X + \beta Y + \delta Z$ and $a'_i = \alpha' X + \beta' Y + \delta' Z$, whereas this time the coefficients of a_i and a'_i are variable, thus `M_eval_all` takes as arguments two lists of triples `_a_coefs` and `_a_prime_coefs` (each triple encoding one a_i or a'_i). The function `coefficients_packing` reshapes as two lists of triples the flat list of reals that `M_func` requires as input.

```
def qft_evaluate(states):
  n = log(len(states[0]))/log(2)
  for state in states:
      rho = matrix(state).transpose()*matrix(state)

     def M_func(_a_a_prime_coefs):
      _a_coefs, _a_prime_coefs = coefficients_packing(
   _a_a_prime_coefs)
      return M_eval_all(n, _a_coefs, _a_prime_coefs, rho)

      _,value = optimize(M_func, [1]*3*n*2, 5, 10**(-2),
   10**2)

      print value
```

Listing 11 Evaluation function for the QFT

4.3 Implementation recap

Finally, to conclude this section, we recall the functions reusable in a general context, the `run` function can be used for general purpose quantum circuit simulation and the Mermin evaluation process can be used for arbitrary state entanglement evaluation. An issue previously mentioned was the correctness between the process and the simulation, and here this issue is tackled by structured and clear code. This structure also helps the code to be more modular, for instance, if the user wants to change the optimization method for more speed or precision, it can be easily achieved.

Remark 6 Note that the actual implemented functions have additional parameters that are ignored here for simplicity's sake. For example, each function has a verbose mode, to display more information about its run.

5 Conclusion

In this paper, we have shown that both Grover's algorithm and the QFT generate states that violate Mermin's inequalities. We provided, for different settings, curves mea-

suring the evolution of the non-local behavior of the states through the algorithms. Evaluation of Mermin polynomials detects entanglement when it violates the classical bound and we compared our numerical results on non-locality evolution with the evolution of values obtained from several measures of entanglement for the same algorithms. Understanding the connection between entanglement and non-local properties of quantum states is a difficult question and we did not intend to provide new theoretical perspectives on this subject. Instead our goal was more to focus on an operational level by studying how specific properties of quantum states generated by those algorithms behave.

This work is a step towards contributions in quantum program verification, by checking state properties, such as entanglement or violation of classical Bell inequality, or temporal properties, such as the increase or decrease of a quantity related to non-locality, during the execution of a quantum program. In the present work we check properties during the execution of the program, for a fixed number of qubits. A promising possibility is to check properties statically, without executing the program and once for all numbers of qubits. A theoretical foundation for this static verification is the quantum Hoare logic [34], an adaptation of the Hoare logic [15] to quantum programs. Mermin polynomials studied in this paper seem promising to check properties during program execution, since Mermin evaluation corresponds to an experimental measurement that could be performed on a quantum computer (see for instance [2] for examples of Mermin evaluation on a 5-qubit computer).

Acknowledgements This project is supported by the French Investissements d'Avenir program, project ISITE-BFC (contract ANR-15-IDEX-03), and by the EIPHI Graduate School (contract ANR-17-EURE-0002). The computations have been performed on the supercomputer facilities of the Mésocentre de calcul de Franche-Comté.
We thank the reviewers of the previous versions of this paper for their valuable comments and remarks, that have helped improving its content.

Appendix A Explicit states for Grover's algorithm

Proposition 2 [14, Observation 1] *The state $|\varphi_k\rangle$ after k iterations of Grover's algorithm can be written as follows:*

$$|\varphi_k\rangle = \tilde{\alpha}_k \sum_{\mathbf{x}\in S} |\mathbf{x}\rangle + \tilde{\beta}_k |+\rangle^{\otimes n} \tag{10}$$

with $\tilde{\alpha}_k = \dfrac{\cos(\frac{2k+1}{2}\theta)}{\sqrt{|S|}} - \dfrac{\sin(\frac{2k+1}{2}\theta)}{\sqrt{N-|S|}}$ *and* $\tilde{\beta}_k = 2^{n/2}\dfrac{\sin(\frac{2k+1}{2}\theta)}{\sqrt{N-|S|}}$.

Proof With $|\varphi_0\rangle = |+\rangle^{\otimes n}$, we can write:

$$|\varphi_k\rangle = \mathcal{L}^k|\varphi_0\rangle = \frac{a_k}{\sqrt{|S|}} \sum_{\mathbf{x}\in S} |\mathbf{x}\rangle + \frac{b_k}{\sqrt{N-|S|}} \sum_{\mathbf{x}\notin S} |\mathbf{x}\rangle$$

where $\mathcal{L}$ is the loop (oracle and diffusion operator) in Grover's algorithm.

The oracle is a reflection about $(\sum_{\mathbf{x}\in S}|\mathbf{x}\rangle)^{\perp} = \sum_{\mathbf{x}\notin S}|\mathbf{x}\rangle$ and the diffusion operator is a reflection about $|+\rangle^{\otimes n}$. The composition of these two symmetries is a rotation whose angle θ is the double of the angle between $\sum_{\mathbf{x}\notin S}|\mathbf{x}\rangle$ and $|+\rangle^{\otimes n}$. So,

$$
\begin{aligned}
|+\rangle^{\otimes n} &= \frac{1}{\sqrt{|S|}}\sin(\frac{\theta}{2})\sum_{\mathbf{x}\in S}|\mathbf{x}\rangle + \frac{1}{\sqrt{N-|S|}}\cos(\frac{\theta}{2})\sum_{\mathbf{x}\notin S}|\mathbf{x}\rangle \\
\frac{1}{\sqrt{N}}\left(\sum_{\mathbf{x}\in S}|\mathbf{x}\rangle + \sum_{\mathbf{x}\notin S}|\mathbf{x}\rangle\right) &= \frac{1}{\sqrt{|S|}}\sin(\frac{\theta}{2})\sum_{\mathbf{x}\in S}|\mathbf{x}\rangle + \frac{1}{\sqrt{N-|S|}}\cos(\frac{\theta}{2})\sum_{\mathbf{x}\notin S}|\mathbf{x}\rangle \\
\frac{1}{\sqrt{N}}\sum_{\mathbf{x}\in S}|\mathbf{x}\rangle &= \frac{1}{\sqrt{|S|}}\sin(\frac{\theta}{2})\sum_{\mathbf{x}\in S}|\mathbf{x}\rangle \\
\frac{1}{\sqrt{N}} &= \frac{1}{\sqrt{|S|}}\sin(\frac{\theta}{2}) \\
\sin(\frac{\theta}{2}) &= \sqrt{\frac{|S|}{N}}.
\end{aligned}
$$

The fact that $\mathcal{L}$ is a rotation of angle θ gives $a_k = \sin(\theta_k)$ and $b_k = \cos(\theta_k)$ with $\theta_k = k\theta + \theta/2$. Equation (1) then comes from $\alpha_k = \frac{1}{\sqrt{|S|}}\sin(\frac{2k+1}{2}\theta)$ and $\beta_k = \frac{1}{\sqrt{N-|S|}}\cos(\frac{2k+1}{2}\theta)$.

With this, we can now take $\tilde{\alpha}_k = \alpha_k - \beta_k$ and $\tilde{\beta}_k = 2^{n/2}\beta_k$ which gives us

$$
\begin{aligned}
|\varphi_k\rangle &= \alpha_k\sum_{\mathbf{x}\in S}|\mathbf{x}\rangle + \beta_k\sum_{\mathbf{x}\notin S}|\mathbf{x}\rangle \\
&= (\alpha_k - \beta_k)\sum_{\mathbf{x}\in S}|\mathbf{x}\rangle + \beta_k\sum_{\mathbf{x}=0}^{N-1}|\mathbf{x}\rangle \\
&= \tilde{\alpha}_k\sum_{\mathbf{x}\in S}|\mathbf{x}\rangle + \tilde{\beta}_k|+\rangle^{\otimes n}
\end{aligned}
$$

since $|+\rangle^{\otimes n} = \left(\frac{1}{\sqrt{2}}\right)^n\sum_{\mathbf{x}=0}^{N-1}|\mathbf{x}\rangle$. □

Proposition 3 *In Proposition 2, $\tilde{\alpha}_k$ increases for k between 0 and $\frac{\pi}{4}\sqrt{\frac{N}{|S|}} - \frac{1}{2}$ and $\tilde{\beta}_k$ decreases on the same interval.*

Proof The optimal number of iterations of the loop $\mathcal{L}$ in Grover's algorithm is the smallest value k_{opt} of k such that $a_k = 1$, *i.e.*, $\theta_{k_{opt}} = \pi/2$. With $|S| \ll N$, $\sin(\theta/2) = \sqrt{|S|/N}$ gives $\theta \approx 2\sqrt{|S|/N}$ and $\theta_k \approx (2k+1)\sqrt{|S|/N}$. Finally $(2k_{opt}+1)\sqrt{|S|/N}$ optimally approximates $\pi/2$ if $k_{opt} = \left\lfloor \frac{\pi}{4}\sqrt{\frac{N}{|S|}} - \frac{1}{2}\right\rceil = \left\lfloor \frac{\pi}{4}\sqrt{\frac{N}{|S|}}\right\rfloor$.

Moreover, $a_k = \sin(\theta_k)$ and $\alpha_k = \frac{1}{\sqrt{|S|}}a_k$ are increasing and $b_k = \cos(\theta_k)$ and $\beta_k = \frac{1}{\sqrt{N-|S|}}b_k$ are decreasing for k from 0 to $\left(\frac{\pi}{4}\sqrt{\frac{N}{|S|}} - \frac{1}{2}\right)$. From the expressions $\tilde{\alpha}_k = \alpha_k - \beta_k$ and $\tilde{\beta}_k = 2^{n/2}\beta_k$, we get the result of the proposition. □

Appendix B Cayley hyperdeterminant Δ_{2222}

Let $|\varphi\rangle = \sum_{i,j,k,l\in\{0,1\}} a_{i,j,k,l}|ijkl\rangle$ be a four-qubit state. The algebra of polynomial invariants for the four-qubit Hilbert space can be generated by the four polynomials H, L, M and D defined as follows [21]:

$$\begin{aligned} H = {} & a_{0000}a_{1111} - a_{1000}a_{0111} - a_{0100}a_{1011} + a_{1100}a_{0011} \\ & - a_{0010}a_{1101} + a_{1010}a_{0101} + a_{0110}a_{1001} - a_{1110}a_{0001} \end{aligned}$$

is an invariant of degree 2.

$$L = \begin{vmatrix} a_{0000} & a_{0010} & a_{0001} & a_{0011} \\ a_{1000} & a_{1010} & a_{1001} & a_{1011} \\ a_{0100} & a_{0110} & a_{0101} & a_{0111} \\ a_{1100} & a_{1110} & a_{1101} & a_{1111} \end{vmatrix} \quad \text{and } M = \begin{vmatrix} a_{0000} & a_{0001} & a_{0100} & a_{0101} \\ a_{1000} & a_{1001} & a_{1100} & a_{1101} \\ a_{0010} & a_{0011} & a_{0110} & a_{0111} \\ a_{1010} & a_{1011} & a_{1110} & a_{1111} \end{vmatrix}$$

are two invariants of degree 4.

Consider the partial derivative

$$b_{xt} := \det\left(\frac{\partial^2 A}{\partial y_i \partial z_j}\right)$$

of the quadrilinear form $A = \sum_{i,j,k,l\in\{0,1\}} a_{i,j,k,l} x_i y_j z_k t_l$ with respect to the variables y and z. This quadratic form with variables x and t can be interpreted as a bilinear form on the three-dimensional space $\mathrm{Sym}^2(\mathbb{C}^2)$, *i.e.*, there is a 3×3 matrix B_{xt} satisfying

$$b_{xt} = [x_0^2, x_0x_1, x_1^2]\, B_{xt} \begin{bmatrix} t_0^2 \\ t_0t_1 \\ t_1^2 \end{bmatrix}.$$

Then $D = \det(B_{xt})$ is an invariant of degree 6.

Let's introduce the invariant polynomials

$$\begin{aligned} & U = H^2 - 4(L - M), \quad V = 12(HD - 2LM), \\ & S = \frac{1}{12}(U^2 - 2V) \quad \text{and} \quad T = \frac{1}{216}(U^3 - 3UV + 216D^2). \end{aligned}$$

Then the Cayley hyperdeterminant is [21]:

$$\Delta_{2222} = S^3 - 27T^2.$$

References

1. Alsina, D., Cervera, A., Goyeneche, D., Latorre, J.I., Życzkowski, K.: Operational approach to Bell inequalities: application to qutrits. Phys. Rev. A **94**(3), 032102 (2016)

2. Alsina, D., Latorre, J.I.: Experimental test of Mermin inequalities on a 5-qubit quantum computer. Phys. Rev. A **94**(1), 012314 (2016)
3. Brunner, N., Gisin, N., Scarani, V.: Entanglement and non-locality are different resources. New J. Phys. **7**, 88 (2005)
4. Biham, O., Nielsen, M.A., Osborne, T.J.: Entanglement monotone derived from Grover's algorithm. Phys. Rev. A **65**(6), 062312 (2002)
5. Batle, J., Ooi, C.H.R., Farouk, A., Alkhambashi, M.S., Abdalla, S.: Global versus local quantum correlations in the Grover search algorithm. Quantum Inf. Process. **15**(2), 833–849 (2016)
6. Braunstein, S.L., Pati, A.K.: Speed-up and entanglement in quantum searching. Quantum Inf. Comput. **2**(5), 399–409 (2002)
7. Chakraborty, S., Banerjee, S., Adhikari, S., Kumar, A.: Entanglement in the Grover's search algorithm. arXiv:1305.4454 [quant-ph] (2013)
8. Collins, D., Gisin, N., Popescu, S., Roberts, D., Scarani, V.: Bell-type inequalities to detect true n-body non-separability. Phys. Rev. Lett. **88**(17), 170405 (2002)
9. Clauser, J.F., Horne, M.A., Shimony, A., Holt, R.A.: Proposed experiment to test local hidden-variable theories. Phys. Rev. Lett. **23**(15), 880–884 (1969)
10. Ekert, A., Jozsa, R.: Quantum algorithms: entanglement-enhanced information processing. Philos. Trans. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci. **356**, 1769–1782 (1998)
11. Grover, L.K.: A Fast quantum mechanical algorithm for database search. In: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, STOC '96. New York, NY, USA. ACM, pp. 212–219 (1996)
12. Gühne, O., Tóth, G.: Entanglement detection. Phys. Rep. **474**(1), 1–75 (2009)
13. Gour, G., Wallach, N.R.: On symmetric SL-invariant polynomials in four qubits. In: Howe, R., Hunziker, M., Willenbring, J.F. (eds.) Symmetry: Representation Theory and Its Applications. Honor of Nolan R. Wallach, Progress in Mathematics, pp. 259–267. Springer, New York, NY (2014)
14. Holweck, F., Jaffali, H., Nounouh, I.: Grover's algorithm and the secant varieties. Quantum Inf. Process. **15**(11), 4391–4413 (2016)
15. Hoare, C.A.R.: An axiomatic basis for computer programming. Commun. ACM **12**(10), 576–580 (1969)
16. Higuchi, A., Sudbery, A.: How entangled can two couples get? Phys. Lett. A **273**(4), 213–217 (2000)
17. Jaffali, H., Holweck, F.: Quantum Entanglement involved in Grover's and Shor's algorithms: the four-qubit case. Quantum Inf. Process. **18**(5), 133 (2019)
18. Jozsa, R., Linden, N.: On the role of entanglement in quantum computational speed-up. Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci. **459**(2036), 2011–2032 (2003)
19. Kendon, V.M., Munro, W.J.: Entanglement and its role in Shor's algorithm. Quantum Inf. Comput. **6**(7), 630–640 (2006)
20. Lavor, C., Manssur, L.R.U., Portugal, R.: Grover's Algorithm: Quantum Database Search. arXiv:quant-ph/0301079 (2003)
21. Luque, J.-G., Thibon, J.-Y.: The polynomial invariants of four qubits. Phys. Rev. A **67**(4), 042303 (2003)
22. Mermin, N.D.: Extreme quantum entanglement in a superposition of macroscopically distinct states. Phys. Rev. Lett. **65**(15), 1838–1840 (1990)
23. Meyer, D.A., Wallach, N.R.: Global entanglement in multiparticle systems. J. Math. Phys. **43**(9), 4273–4278 (2002)
24. Miyake, A., Wadati, M.: Multipartite entanglement and hyperdeterminants. Quantum Inf. Comput. **2**(7), 540–555 (2002)
25. Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information: 10th, Anniversary edn. Cambridge University Press, Cambridge (2010)
26. Osterloh, A., Siewert, J.: Entanglement monotones and maximally entangled states in multipartite qubit systems. Int. J. Quantum Inf. **04**(03), 531–540 (2006)
27. Rossi, M., Bruß, D., Macchiavello, C.: Scale invariance of entanglement dynamics in Grover's quantum search algorithm. Phys. Rev. A Atom. Mol. Opt. Phys. **87**(2), 1–5 (2013)
28. Shimony, A.: Degree of entanglement. Ann. N. Y. Acad. Sci. **755**(1), 675–679 (1995)
29. Shor, P.W., Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th Annual Symposium on Foundations of Computer Science. Santa Fe, NM, USA, : IEEE Comput. Press, Soc, pp. 124–134 (1994)

30. Shimoni, Y., Shapira, D., Biham, O.: Entangled quantum states generated by Shor's factoring algorithm. Phys. Rev. A **72**(6), 062308 (2005)
31. Toth, G., Guehne, O.: Entanglement detection in the stabilizer formalism. Phys. Rev. A **72**(2), 022340 (2005)
32. Verstraete, F., Dehaene, J., De Moor, B., Verschelde, H.: Four qubits can be entangled in nine different ways. Phys. Rev. A **65**(5), 052112 (2002)
33. Wei, T.-C., Goldbart, P.M.: Geometric measure of entanglement and applications to bipartite and multipartite quantum states. Phys. Rev. A **68**(4), 042307 (2003)
34. Ying, M.: Floyd–Hoare logic for quantum programs. ACM Trans. Program. Lang. Syst. **33**(6), 1–49 (2011)