Logical operators of quantum codes

Mark M. Wilde*
Electronic Systems Division, Science Applications International Corporation, 4001 North Fairfax Drive, Arlington, Virginia 22203, USA
(Received 30 March 2009; published 24 June 2009)

I show how applying a symplectic Gram-Schmidt orthogonalization to the normalizer of a quantum code gives a different way of determining the code's logical operators. This approach may be more natural in the setting where we produce a quantum code from classical codes because the generator matrices of the classical codes form the normalizer of the resulting quantum code. This technique is particularly useful in determining the logical operators of an entanglement-assisted code produced from two classical binary codes or from one classical quaternary code. Finally, this approach gives additional formulas for computing the amount of entanglement that an entanglement-assisted code requires.

DOI: 10.1103/PhysRevA.79.062322 PACS number(s): 03.67.Hk, 03.67.Pp

I. INTRODUCTION

The logical operators of a quantum code are important for fault-tolerant quantum computation [1] and in the simulation of the performance of stabilizer quantum error-correcting codes [2]. The typical method for determining the logical operators of a quantum code is to employ Gottesman's algorithm [3]. This algorithm brings the stabilizer of a quantum code into a standard form and extracts the logical operators from this standard form.

In this paper, I provide another technique for determining the logical operators of a quantum code. This technique begins with the *normalizer* of the code and is perhaps more natural in the setting where we produce a (Calderbank-Shor-Steane) CSS quantum code from two classical additive binary codes [4,5] or where we produce a (Calderbank-Rains-Shor-Sloane) CRSS quantum code from one classical additive code over the Galois field of four elements GF(4) [6]. In both of these cases, we possess the full normalizer of the resulting quantum code because the generator matrices of the classical codes form its normalizer. Additionally, I show that this technique is useful in determining the logical operators of an entanglement-assisted code [7,8], a type of quantum code to which Gottesman's algorithm does not apply. The technique is again natural in the setting where we import classical codes because the generator matrices form the full normalizer of the resulting entanglement-assisted code. This technique also gives another way of computing the amount of entanglement that a given entanglement-assisted code requires (in this sense, this paper is a sequel to the findings in Ref. [9]). Specifically, I give a set of formulas, different from those in Ref. [9], for computing the entanglement that a given entanglement-assisted code requires. The computation of the formulas in this paper may be more efficient than those in Ref. [9] and the efficiency depends on the parameters of a given code. The technique for computing the logical operators and the entanglement formulas in this paper should be a useful addition to the quantum code designer's toolbox.

*mark.m.wilde@saic.com

II. REVIEW OF THE SYMPLECTIC GRAM-SCHMIDT ORTHOGONALIZATION PROCEDURE

I first review the symplectic Gram-Schmidt orthogonalization procedure (SGSOP) before proceeding to the main development of this paper. The SGSOP is equivalent to finding a standard symplectic basis for a set of vectors [10]. It has made appearances in the quantum information literature in entanglement-assisted quantum coding [7,8], in finding an entanglement measure for a stabilizer state [11], in the simulation of measurement outcomes of stabilizer codes [12], and in the decomposition of subsystem codes [13].

We begin with a set of m Pauli generators $g_1,\ldots,g_m$ that do not necessarily form a commuting set. Consider generator g_1. There are two possible cases for this generator:

(1) Generator g_1 commutes with all other generators $g_2,\ldots,g_m$. Set it aside in a "set of processed generators," relabel generators $g_2,\ldots,g_m$ as respective generators $g_1,\ldots,g_{m-1}$, and repeat the SGSOP on these remaining generators.

(2) Generator g_1 anticommutes with some other generator g_j where $2\leq j\leq m$. Relabel g_j as g_2 and vice versa. Modify the other generators as follows:

$$\forall i \in \{3,\ldots,m\} \quad g_i = g_i g_1^{f(g_i,g_2)} g_2^{f(g_i,g_1)},$$

where the function f is equal to zero if its two arguments commute and is equal to one if its two arguments anticommute. (Note that we are not concerned with the overall phases of any of the generators in this paper.) The new generators $g_3,\ldots,g_m$ then commute with g_1 and g_2. Place g_1 and g_2 into the set of processed generators, relabel generators $g_3,\ldots,g_m$ as respective generators $g_1,\ldots,g_{m-2}$, and repeat the SGSOP on these remaining generators.

The result of the SGSOP is to produce a set of m processed generators where $2c$ of them form anticommuting pairs (each pair commuting with the generators in the other pairs), and l of them commute with themselves and the c pairs of generators. It is a reversible algorithm, meaning that we can recover the original set of generators by performing all of the steps in reverse, and its complexity is $O(nm^2)$. This complexity is the same as the complexity of Gottesman's algorithm for bringing the generating set into a standard form

and determining the logical operators from this standard form [3].

III. GENERAL STABILIZER CODES

We now consider applying the SGSOP to the normalizer $N(S)$ of a stabilizer code [3] with stabilizer S. The result is that the SGSOP gives a different way for determining the logical operators of a given stabilizer code.

Suppose that we have a set $\langle S\rangle$ of $n-k$ Pauli generators where each Pauli generator in $\langle S\rangle$ is an n-fold tensor product of Pauli matrices. Suppose furthermore that $\langle S\rangle$ generates an Abelian group. Suppose we also have a generating set $\langle N(S)\rangle$ of p elements where $p>n-k$ and $\langle N(S)\rangle$ generates the normalizer $N(S)$ of the stabilizer S. The relation $S\subseteq N(S)$ holds for this case because the set of generators in $\langle S\rangle$ generates an Abelian group.

We can perform the SGSOP on the generating set $\langle N(S)\rangle$. The SGSOP divides $\langle N(S)\rangle$ into two different sets: the *logical generating set* $\langle S_L\rangle$ and the stabilizer generating set $\langle S\rangle$. We can write the latter generating set as $\langle S\rangle$ because it generates the same group that the generating set from the previous paragraph generates. The logical generating set $\langle S_L\rangle$ consists of k anticommuting pairs of generators that correspond to the logical operators for the stabilizer code S. The size of the generating set $\langle N(S)\rangle$ is then $p=n-k+2k=n+k$.

We can also think about the code in terms of binary vectors that represent it [3,14]. Suppose we have the normalizer matrix:

$$N \equiv [N_Z \mid N_X],$$

where N is the binary representation of the generators in $\langle N(S)\rangle$. We can consider the "symplectic product matrix" Ω_N [9] where

$$\Omega_N \equiv N_Z N_X^T + N_X N_Z^T,$$

and addition is binary. It is then straightforward to show by the method of proof of Theorem 1 in Ref. [9] that

$$\frac{1}{2}\mathrm{rank}(\Omega_N) = k.$$

The idea of the proof is that the application of the SGSOP to the normalizer matrix N transforms the matrix Ω_N to the following standard form:

$$\bigoplus_{i=1}^{k} J \oplus \bigoplus_{j=1}^{n-k} [0], \tag{1}$$

where the small and large $\oplus$ correspond to the direct sum operation, J is the matrix

$$J = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \tag{2}$$

and $[0]$ is the one-element zero matrix. Each matrix J in the direct sum corresponds to a *symplectic pair* and has rank of two. Each symplectic pair corresponds to exactly one logical operator in the code. Each matrix $[0]$ has rank of zero and corresponds to an ancilla qubit. See Ref. [9] for more details.

A. CSS codes

One major class of quantum codes is the class of CSS codes [4,5]. These codes allow us to import two "dual-containing" classical codes for use as a quantum code. The resulting quantum code inherits the error-correcting properties and rate from the classical codes.

Here, we show that the SGSOP provides a way to determine the logical operators for a CSS code. It constructs the full code (stabilizer and logical operators) from the generator matrices of the two classical codes rather than exploiting the parity check matrices.

Suppose we have two classical codes C_1 and C_2 that satisfy $C_2^{\perp}\subseteq C_1$. Suppose their respective generator matrices are G_1 and G_2, and their respective parity check matrices are H_1 and H_2. The conditions $H_1G_1^T=0$ and $H_2G_2^T=0$ follow from the definition of a parity check matrix, and the condition $H_1H_2^T=0$ follows from the condition $C_2^{\perp}\subseteq C_1$.

The typical method for constructing a CSS quantum code is to build the following stabilizer matrix:

$$\left[\begin{array}{c|c} H_1 & 0 \\ 0 & H_2 \end{array}\right].$$

The code forms a valid stabilizer matrix because $H_1H_2^T=0$ (this condition corresponds to the requirement for a stabilizer code to consist of a commuting set of generators).

The number of generators in the CSS quantum code is equal to $2n-k_1-k_2$, and it therefore has $n-(2n-k_1-k_2)=k_1+k_2-n$ logical qubits with $2(k_1+k_2-n)$ logical operators. The size of the normalizer is thus $(2n-k_1-k_2)+2(k_1+k_2-n)=k_1+k_2$.

There is another way to build the code by exploiting the generator matrices G_1 and G_2. The below matrix is the normalizer matrix of the code:

$$\left[\begin{array}{c|c} 0 & G_1 \\ G_2 & 0 \end{array}\right]. \tag{3}$$

The conditions $H_1G_1^T=0$ and $H_2G_2^T=0$ guarantee that the above matrix is a valid normalizer matrix. It represents the full normalizer for the code because the rows are linearly independent and the number of generators in the normalizer is k_1+k_2.

Suppose that we run the SGSOP on the rows of the normalizer matrix in Eq. (3). The procedure then produces k_1+k_2-n anticommuting pairs and $n-k_1+n-k_2$ commuting generators. The k_1+k_2-n anticommuting pairs correspond to $2(k_1+k_2-n)$ logical operators for the code, and the $n-k_1+n-k_2$ commuting generators correspond to the stabilizer generators of the code. This technique for obtaining the logical operators and the stabilizer generators in the case of a CSS code is perhaps more natural than applying Gottesman's method in Ref. [3] because we already know the matrices G_1 and G_2 when importing classical codes.

By the same method of proof as Corollary 1 in Ref. [9], the following formula also holds for any CSS code

$$\mathrm{rank}(G_1G_2^T) = k_1 + k_2 - n, \tag{4}$$

because this quantity captures the amount of anticommutativity in the generators in Eq. (3).

B. CRSS quantum codes

A CRSS quantum code is one that we create by importing a classical additive code over GF(4) [6].

Suppose a GF(4) code has parity check matrix H and generator matrix G where $HG^T=0$. Additionally, suppose that the code is dual under the trace product over GF(4), i.e., the following condition holds:

$$\mathrm{tr}\{HH^\dagger\}=0,$$

where the dagger symbol † is the conjugate transpose of matrices over GF(4) and the null matrix on the right has dimension $(n-k)\times(n-k)$. Note that the above trace operation is different from the standard matrix trace but is rather an elementwise computation of the trace product over GF(4) [9].

We can then create a quantum check matrix:

$$\gamma\left(\begin{bmatrix}\omega H\\ \bar{\omega}H\end{bmatrix}\right), \tag{5}$$

where γ is the isomorphism that takes an n-dimensional vector over GF(4) to a n-fold tensor product of Pauli operators [9].

It is straightforward to determine the normalizer of such a code because we can construct it from the generator matrix G:

$$\begin{bmatrix}\omega G^*\\ \bar{\omega}G^*\end{bmatrix}. \tag{6}$$

The above matrix is a valid normalizer for the resulting quantum code because

$$\begin{aligned}\mathrm{tr}\left\{\begin{bmatrix}\omega H\\ \bar{\omega}H\end{bmatrix}\begin{bmatrix}\omega G^*\\ \bar{\omega}G^*\end{bmatrix}^\dagger\right\}&=\mathrm{tr}\left\{\begin{bmatrix}\omega H\\ \bar{\omega}H\end{bmatrix}[\bar{\omega}G^T\ \ \omega G^T]\right\}\\&=\mathrm{tr}\left\{\begin{bmatrix}HG^T & \bar{\omega}HG^T\\ \omega HG^T & HG^T\end{bmatrix}\right\}=\begin{bmatrix}0&0\\0&0\end{bmatrix},\end{aligned} \tag{7}$$

where the first equality is by definition of the conjugate transpose, and the second equality holds because $\omega\bar{\omega}=1$, $\bar{\omega}\bar{\omega}=\omega$, and $\omega\omega=\bar{\omega}$ for elements of GF(4). The last equality follows from the condition $HG^T=0$.

We then can perform the SGSOP on the above normalizer matrix in Eq. (6). The result of the algorithm is to produce $2(2k-n)$ anticommuting pairs and $2(n-k)$ generators that commute with themselves and the pairs. This method again determines the logical operators for the imported code and may be a more natural alternative to Gottesman's method because we already know the matrix G that determines the normalizer for the resulting CRSS quantum code.

By the same method of proof as Corollary 2 in Ref. [9], the following formula holds for a trace-orthogonal code imported from GF(4):

$$\mathrm{rank}(GG^\dagger)=2k-n.$$

IV. ENTANGLEMENT-ASSISTED QUANTUM CODES

We can extend this procedure to an entanglement-assisted code as well. This alternative procedure for determining the logical operators is useful here because Gottesman's method does not apply to a non-Abelian group of Pauli generators.

Suppose that we have a set $\langle S\rangle$ of m Pauli generators that represents a generating set of operators with desirable error-correcting properties. The generating set $\langle S\rangle$ does not necessarily generate an Abelian group. Suppose we also have a generating set $\langle N(S)\rangle$ of p elements. This generating set $\langle N(S)\rangle$ generates the normalizer $N(S)$ of the stabilizer S. In general, the relation $S\subseteq N(S)$ does not necessarily hold for this case because the set of generators in $\langle S\rangle$ is not an Abelian group.

We can perform the SGSOP on the generators in $\langle S\rangle$. This procedure divides $\langle S\rangle$ into two sets of generators: the *entanglement generator set* $\langle S_E\rangle$ and the *isotropic generator set* $\langle S_I\rangle$. The set $\langle S_E\rangle$ contains anticommuting pairs of generators (where generators different pairs commute), and the set $\langle S_I\rangle$ contains a commuting set of generators that furthermore commute with the generators in $\langle S_E\rangle$. The size of the generating set $\langle S_E\rangle$ is $2c$, and the size of the set $\langle S_I\rangle$ is i so that $m=i+2c$.

We can also perform the SGSOP on the generating set $\langle N(S)\rangle$. The SGSOP divides this generating set into two sets: the logical generating set $\langle S_L\rangle$ and the isotropic generating set $\langle S_I\rangle$. We can write this isotropic generating set as $\langle S_I\rangle$ because it generates the same group as the group generated by the isotropic generating set from the previous paragraph. The size of the generating set $\langle N(S)\rangle$ is then $p=i+2l$. The generators in the logical generating set $\langle S_L\rangle$ are the logical operators for the code.

We can again consider this procedure in the binary vector representation. Suppose the following matrix specifies the normalizer matrix:

$$N\equiv[N_Z\,|\,N_X].$$

Consider the matrix Ω_N where

$$\Omega_N\equiv N_ZN_X^T+N_XN_Z^T,$$

and addition is binary. By the same method of proof as Theorem 1 in Ref. [9], we can show that

$$\frac{1}{2}\mathrm{rank}(\Omega_N)=l.$$

A. CSS entanglement-assisted codes

We now consider extending the development in Sec. IV to the formulation of a CSS entanglement-assisted code [7,8]. This extension gives a useful way for determining the logical operators of a CSS entanglement-assisted quantum code. Additionally, we determine a formula, different from that in Corollary 1 of Ref. [9], for computing the amount of entanglement that a given CSS entanglement-assisted quantum code requires.

Consider two arbitrary additive binary codes C_1 and C_2 with respective parity check matrices H_1 and H_2 and respective generator matrices G_1 and G_2. We can form a quantum check matrix as follows:

$$\left[\begin{array}{c|c} H_1 & 0 \\ 0 & H_2 \end{array}\right]. \tag{8}$$

The above matrix is not a valid stabilizer matrix but we can run the SGSOP on it to determine how to add entanglement in order to make this code a valid stabilizer code. After adding ebits to the code, it then becomes an entanglement-assisted stabilizer code. A CSS entanglement-assisted code constructed in this way has $2n-k_1-k_2$ generators ($2c$ of which form anticommuting pairs and $2n-k_1-k_2-2c$ form the isotropic generating set), k_1+k_2-n+c logical qubits, and consumes c ebits [7].

The normalizer of the generators in Eq. (8) is again

$$\left[\begin{array}{c|c} 0 & G_1 \\ G_2 & 0 \end{array}\right],$$

and represents the full normalizer of the generators in Eq. (8) because the number of generators is k_1+k_2 and the rows are linearly independent. This time, when we run the algorithm, we get $2(k_1+k_2-n+c)$ anticommuting pairs and $n-k_1+n-k_2-2c$ commuting generators. The commuting generators form the basis for the isotropic generating set of the code and the anticommuting generators correspond to the logical operators of the code.

By the method of proof of Corollary 1 in Ref. [9], we can show that the following relation holds:

$$\text{rank}(G_1 G_2^T) = k_1 + k_2 - n + c. \tag{9}$$

This formula gives us another way to determine the amount of ebits that a given entanglement-assisted code requires because the parameters n, k_1, and k_2 are known parameters.

The complexity of the computation in Eq. (9) is $O(k_1 k_2 n)$. A similar formula from Ref. [9] computes the amount of entanglement that the code requires:

$$\text{rank}(H_1 H_2^T) = c.$$

The complexity of the above formula is $O[(n-k_1)(n-k_2)n]$. Thus, the formula in Eq. (9) may provide a speedup for low-rate codes. It is up to the quantum code designer to decide which formula to use depending on the parameters of the code.

B. CRSS entanglement-assisted codes

The method for constructing a CRSS entanglement-assisted code is again to import an additive classical code over GF(4). Suppose the imported code has parity check matrix H and generator matrix G so that $HG^T=0$. The imported code does not have to be orthogonal with respect to the trace product. We then create a quantum check matrix:

$$\gamma\left(\left[\begin{array}{c} \omega H \\ \bar{\omega} H \end{array}\right]\right),$$

where γ is the same isomorphism as in Eq. (5).

The normalizer of the code is again the same as in Eq. (6). The normalizer matrix is

$$\gamma\left(\left[\begin{array}{c} \omega G^* \\ \bar{\omega} G^* \end{array}\right]\right),$$

and it is a valid normalizer for the same reasons as in Eq. (7).

Performing the symplectic Gram-Schmidt orthogonalization procedure on the normalizer matrix, we then get $2(2k-n+c)$ anticommuting pairs and $2(n-k-c)$ commuting generators. This method again determines the logical operators for the imported code.

By the same method of proof as Corollary 2 in Ref. [9], we can show that the following formula holds:

$$\text{rank}(GG^\dagger) = 2k - n + c. \tag{10}$$

The formula gives another way for determining the amount of entanglement that an entanglement-assisted quantum code requires because the parameters k and n are known parameters.

The complexity of the computation of rank $(GG^\dagger)$ is $O(k^2 n)$. A similar formula from Ref. [9] computes the amount of entanglement that the codes requires:

$$\text{rank}(HH^\dagger) = c.$$

The complexity of the above formula is $O[(n-k)^2 n]$. Thus, the formula in Eq. (10) may again provide a speedup for low-rate codes.

V. CONCLUSION

I have shown how the normalizer of a quantum code, whether stabilizer or entanglement assisted, gives a useful way for determining its logical operators. It is natural to assume full knowledge of the normalizer in the case where we import classical codes to produce quantum codes. Additionally, this development gives formulas for computing the amount of entanglement that an entanglement-assisted code requires, and these formulas are an alternative to those given in Ref. [9]. It is straightforward to extend these methods and formulas to qudit codes and continuous-variable codes, by glancing at the results in Ref. [9].

Future work includes determining how to apply these findings in the setting of quantum convolutional codes [15,16] and entanglement-assisted quantum convolutional codes [17]. The author and co-workers have made attempts at this problem by devising the algorithm in Sec. IVD of Ref. [18] for the CSS case, and the expansion technique and polynomial symplectic Gram-Schmidt algorithm in, respectively, Secs. IV and V of Ref. [19] for the CRSS case. It is more complicated in the convolutional setting because we would like to maintain the periodic structure of the quantum convolutional code while still maintaining the standard symplectic relations. An algorithm that achieves both tasks would be a boon to the theory of quantum convolutional coding.

ACKNOWLEDGMENTS

I thank Andrew Cross for useful comments on the paper, and acknowledge the support of an internal research and development Grant No. SAIC-1669 of Science Applications International Corporation.

[1] F. Gaitan, *Quantum Error Correction and Fault Tolerant Quantum Computing* (CRC Press/Taylor and Francis Group, Boca Raton, FL, 2008).
[2] A. Cross, D. DiVincenzo, and B. Terhal, Quantum Inf. Comput. **9**, No. 7-8, 0541 (2009).
[3] D. Gottesman, Ph.D. thesis, California Institute of Technology, 1997.
[4] A. R. Calderbank and P. W. Shor, Phys. Rev. A **54**, 1098 (1996).
[5] A. M. Steane, Phys. Rev. Lett. **77**, 793 (1996).
[6] A. R. Calderbank, E. M. Rains, P. W. Shor, and N. J. A. Sloane IEEE Trans. Inf. Theory **44**, 1369 (1998).
[7] T. Brun, I. Devetak, and M. Hsieh, e-print arXiv:quant-ph/0608027.
[8] T. A. Brun, I. Devetak, and M.-H. Hsieh, Science **314**, 436 (2006).
[9] M. M. Wilde and T. A. Brun, Phys. Rev. A **77**, 064302 (2008).
[10] A. Cannas da Silva, *Lectures on Symplectic Geometry* (Springer, New York, 2001).
[11] D. Fattal, T. Cubitt, Y. Yamamoto, S. Bravyi, and I. L. Chuang, e-print arXiv:quant-ph/0406168.
[12] S. Aaronson and D. Gottesman, Phys. Rev. A **70**, 052328 (2004).
[13] D. Poulin, Phys. Rev. Lett. **95**, 230504 (2005).
[14] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, New York, 2000).
[15] M. Grassl and M. Rötteler, *Quantum Convolutional Codes: Encoders and Structural Properties*, in Proceedings of the Forty-Fourth Annual Allerton Conference on Communication, Control, and Computing (UIUC, Monticello, IL, 2006), pp. 510–519.
[16] G. D. Forney, M. Grassl, and S. Guha, IEEE Trans. Inf. Theory **53**, 865 (2007).
[17] M. M. Wilde and T. Brun, e-print arXiv:0712.2223.
[18] M. M. Wilde, H. Krovi, and T. Brun, e-print arXiv:0708.3699.
[19] M. M. Wilde and T. Brun, e-print arXiv:0807.3803.