Examples of minimal-memory, non-catastrophic quantum convolutional encoders

Mark M. Wilde, Monireh Houshmand, and Saied Hosseini-Khayat

Abstract—One of the most important open questions in the theory of quantum convolutional coding is to determine a minimal-memory, non-catastrophic, polynomial-depth convolutional encoder for an arbitrary quantum convolutional code. Here, we present a technique that finds quantum convolutional encoders with such desirable properties for several example quantum convolutional codes (an exposition of our technique in full generality appears elsewhere). We first show how to encode the well-studied Forney-Grassl-Guha (FGG) code with an encoder that exploits just one memory qubit (the former Grassl-Rötteler encoder requires 15 memory qubits). We then show how our technique can find an online decoder corresponding to this encoder, and we also detail the operation of our technique on a different example of a quantum convolutional code. Finally, the reduction in memory for the FGG encoder makes it feasible to simulate the performance of a quantum turbo code employing it, and we present the results of such simulations.

Index Terms—quantum convolutional coding, non-catastrophic, minimal memory, quantum turbo code

I. INTRODUCTION

Quantum convolutional coding is a method for quantum error correction that protects a stream of quantum information from the negative effects of decoherence [1], [2], [3]. This approach is highly beneficial in a quantum communication paradigm where the only decoherence is due to a noisy quantum channel connecting a sender to a receiver. Since the original work on quantum convolutional coding, researchers have contributed a notable literature on several aspects of these codes: methods for encoding them [4], [5], [6], algebraic constructions of them [7], [8], and variations that include resources such as gauge qubits, entanglement shared between sender and receiver [9], [10], and hybrid constructions encoding both classical and quantum bits [11]. Further progress has led to a successful "quantization" of the classical turbo coding theory [12], [13]—quantum serial turbo codes employing constituent quantum convolutional codes appear to be capacity-approaching codes in the standard [14] and entanglement-assisted settings [15].

Despite this progress, an important question concerning the practical implementation of a quantum convolutional code has remained unanswered: *Is there a way to implement a given quantum convolutional code with a non-catastrophic, efficient, minimal-memory encoder?* This question is not only important for reducing the overhead needed to implement these

Mark M. Wilde is with the School of Computer Science, McGill University, Montreal, Quebec, Canada, H3A 2A7. Monireh Houshmand and Saied Hosseini-Khayat are with the Department of Electrical Engineering, Ferdowsi University of Mashhad, Mashhad, Iran. (E-mail: mark.wilde@mcgill.ca; monireh.houshmand@gmail.com; shk@ieee.org)

codes with a coherent quantum device, but it is also important for the algorithm used to decode them—the post-processing time for the decoding algorithm is exponential in the number of memory qubits (specifically, it is $O\left(4^m N\right)$ where m is the number of memory qubits and N is the length of the code [14]). Any reduction in the size of the memory could become especially important in the context of fault-tolerant quantum computing (if these codes are ever exploited for this purpose) because decoding delays can translate into the accumulation of more errors.

Past efforts have not given satisfactory answers to the aforementioned question. The Ollivier-Tillich [1], [16] and Grassl-Rötteler [4], [5], [6] algorithms for encoding quantum convolutional codes are useful methods for accomplishing their intended goals, but they both have no concern for quantum memory consumption. Also, the Grassl-Rötteler algorithm in general can potentially lead to an encoding circuit with exponential depth [4]. We have attempted to answer the minimal-memory question in prior work [17], [18], but our former approaches are suboptimal in general—these approaches begin with a pearl-necklace encoder resulting from the Grassl-Rötteler algorithm, and they then find a particular convolutional encoder that uses the minimal amount of memory for that particular pearl-necklace encoder (thus, they are suboptimal because the original Grassl-Rötteler pearl-necklace encoder might not result in a minimal-memory encoder).

In this paper, we present a technique for finding a minimal-memory, non-catastrophic quantum convolutional encoder for several examples of quantum convolutional codes (a presentation of our technique in full generality appears elsewhere [19]). Our first example encoder is for the Forney-Grassl-Guha (FGG) code [2], [3], and our technique gives a dramatic memory-consumption reduction from 15 memory qubits for the original Grassl-Rötteler encoding [4], [20] to *just one memory qubit*. Our encoding technique simultaneously accomplishes three goals: it finds a minimal-memory encoder for a quantum convolutional code, it can force the resulting encoder to be non-catastrophic [14], [15], and it provides an efficient encoder in the sense that its depth is only $O\left(n^2\right)$ where n is the number of qubits in a frame of the code. Interestingly, the formula for the minimal number of memory qubits [19] bears a close relationship to a formula that gives the minimal number of ebits needed to encode an entanglement-assisted quantum code [21], [22]. This connection to entanglement-assisted quantum coding is perhaps not surprising because the memory qubits of the encoder are entangled for some time with the qubits sent over the channel. We also show how to determine an online decoder corresponding to the encoder for

the FGG code. The technique for finding it is in the same spirit as the technique for finding the online encoder, but the online decoder takes special care to decode both the logical operators and the stabilizer operators of the code. The dramatic reduction in memory consumption for the FGG encoder makes it reasonable to simulate the performance of a quantum turbo code employing this encoder for its constituent codes, and we plot the results of simulations in this paper.

We structure this paper as follows. The next section details how to find a minimal-memory, non-catastrophic encoder for the FGG code. Section III details how to find an online decoder corresponding to the choice of encoder in Section II. We then detail how our technique finds a minimal-memory, non-catastrophic encoder of a more complicated quantum convolutional code from Ref. [6]. The final section discusses and plots the results of simulating the performance of the rate-1/9 FGG quantum turbo code.

II. A MINIMAL-MEMORY, NON-CATASTROPHIC ENCODER FOR THE FORNEY-GRASSL-GUHA CODE

We first describe how our technique finds a minimal-memory, non-catastrophic encoder for the FGG quantum convolutional code [2], [3]. The stabilizer generators for this quantum convolutional code are as follows:

$$\begin{array}{ccc|ccc|ccc|}
X & X & X & X & Z & Y & I & I & I \\
Z & Z & Z & Z & Y & X & I & I & I
\end{array} \cdots, \quad (1)$$

where the vertical bars indicate that three qubits are in each frame and we obtain the other generators of the code by shifting the above generators to the right by multiples of three qubits. This code features two generators for every three physical qubits, and so we should be able to encode it with a quantum convolutional encoder of the form in Figure 1(a). In particular, the encoder should act on some number m of memory qubits, two ancilla qubits, and one information qubit to produce three output physical qubits and m output memory qubits to be fed into the next round of encoding.

After inspecting Figure 1 and its caption, it should be clear that an online encoder for the FGG code should transform the below Pauli operators on the LHS to the ones on the RHS:

$$\begin{array}{ccccccccc}
I^{\otimes m} & Z & I & I & & X & X & X & g_1 \\
I^{\otimes m} & I & Z & I & & Z & Z & Z & g_2 \\
g_1 & I & I & I & \to & X & Z & Y & I^{\otimes m} \\
g_2 & I & I & I & & Z & Y & X & I^{\otimes m}
\end{array}, \quad (2)$$

where m is some unspecified number of memory qubits and g_1 and g_2 are m-qubit Pauli operators that we soon determine. The key observation to make at this point is that any choice of the Clifford encoder U should preserve commutation relations. That is, if two $(n+m)$-fold Pauli operators on the LHS commute, then the transformed versions of them on the RHS should commute as well, and similarly if the input Pauli operators in (2) anticommute. So, observe that the first two input Pauli operators commute. This implies that the transformed versions of them should commute, and g_1 and g_2 should thus anticommute to make $XXXg_1$ and $ZZZg_2$ commute. Also, consider that the first and second LHS rows in (2) commute with both the third and fourth LHS rows. The

corresponding output rows commute because the FGG code is a valid quantum convolutional code. Finally, observe that the third and fourth output rows anticommute. Thus, the third and fourth input rows should anticommute, and this is consistent with our above observation that g_1 and g_2 should anticommute.

A sufficient choice for g_1 and g_2 to satisfy the above commutation constraints is $g_1 = X$ and $g_2 = Z$, and this choice implies that the Clifford encoder acts on just one memory qubit. This choice is optimal because the encoder needs at least one memory qubit to encode the FGG code. Once we have specified the Pauli operators g_1 and g_2, we can always find a Clifford encoder that performs the transformation in (2) because the Clifford group acts transitively on Pauli operators (see Lemma 4 in Appendix B of Ref. [23] for an explicit proof). A particular encoder that performs the transformation in (2) is as follows:

$$\text{H}\,(2)\ \ \text{CNOT}\,(4,1)\ \ \text{H}\,(4)\ \ \text{CNOT}\,(4,1)\ \ \text{CNOT}\,(4,2)$$
$$\text{CNOT}\,(1,4)\ \ \text{H}\,(4)\ \ \text{CNOT}\,(3,4)\ \ \text{P}\,(1)\ \ \text{CNOT}\,(4,3)$$
$$\text{CNOT}\,(1,3)\ \ \text{CNOT}\,(2,1)\ \ \text{CNOT}\,(2,3)\ \ \text{CNOT}\,(2,4)\,, \quad (3)$$

where the ordering of gates is from left to right and top to bottom, $\text{H}(i)$ indicates a Hadamard gate acting on the ith qubit, $\text{CNOT}\,(i,j)$ indicates a CNOT gate from the ith qubit to the jth qubit, and $\text{P}(i)$ indicates a phase gate acting on the ith qubit. We found this encoder by exploiting Grassl's algorithm for determining Clifford unitaries [24].

We furthermore claim that any Clifford encoder performing the transformation in (2) is non-catastrophic. Recall from Refs. [14], [15] that a catastrophic encoder is one whose state diagram contains a cycle of zero physical weight that has non-zero logical weight (this definition is in fact *the same* as in the classical case [25]). Suppose for a contradiction that an encoder performing the transformation is catastrophic. This would imply that the encoder creates some cycle through memory states $h_1, \ldots, h_p$ of the following form:

$$\begin{array}{ccccccccc}
h_1 & s_{1,1} & s_{1,2} & l_1 & & I & I & I & h_2 \\
\vdots & \vdots & \vdots & \vdots & \to & \vdots & \vdots & \vdots & \vdots \\
h_p & s_{p,1} & s_{p,2} & l_p & & I & I & I & h_1
\end{array},$$

where $h_1, \ldots, h_p$ can be arbitrary one-qubit Pauli operators (since the memory consists of just one qubit), $s_{i,j}$ are one-qubit Pauli operators equal to either the identity or Pauli Z, and l_i are arbitrary one-qubit Pauli operators with at least one of them not equal to the identity operator. Observe that all of the output rows on the RHS above commute with the last two rows on the RHS of the transformation in (2). This observation implies that all of the rows on the LHS above should commute with the last two rows on the LHS of the transformation in (2). But this is only possible if $h_1, \ldots, h_p$ are all equal to the one-qubit identity operator because $g_1 = X$ and $g_2 = Z$. So all of the above entries are really just cycles of the form $I\quad s_1\quad s_2\quad l\ \to\ I\quad I\quad I\quad I$. This input-output relation restricts s_1, s_2, and l further—it is impossible for s_1, s_2, and l to be any Pauli operator besides the identity operator. Otherwise, the encoder would not transform the entry on the LHS to the all identity operator. Thus, the only cycle of zero-physical weight in an encoder that
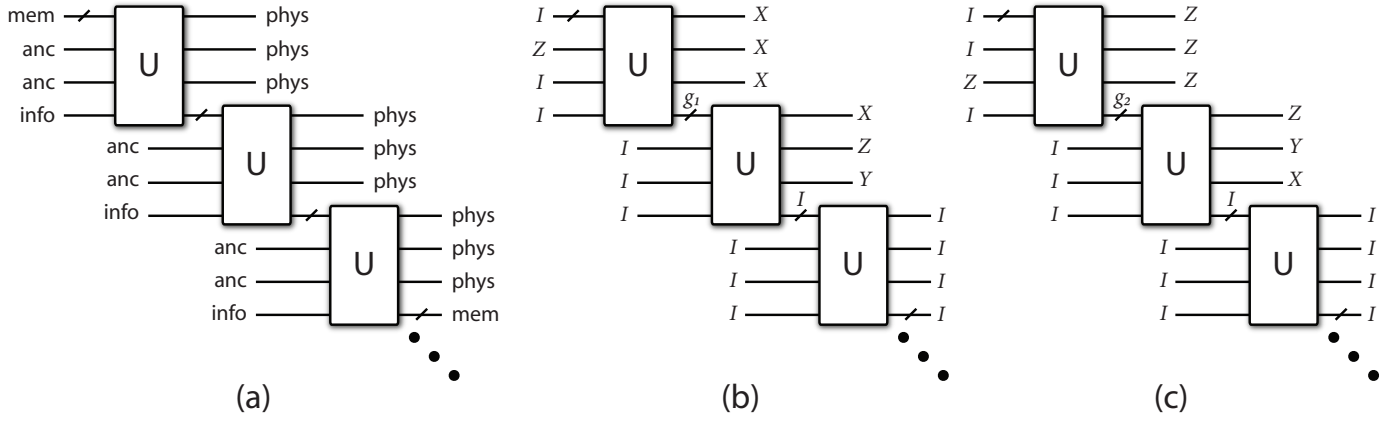
Fig. 1. The above figure provides a graphical aid for understanding our technique that encodes the Forney-Grassl-Guha (FGG) code with a minimal-memory encoder. (a) The convolutional encoder U for the FGG code should act on some unknown number m of memory qubits labeled by "mem" (the diagonal slash through a horizontal line indicates m qubits), two ancillas labeled by "anc," and one information qubit labeled by "info." It produces three output physical qubits labeled by "phys" and m output memory qubits to be fed into the next round of encoding. (b) The repeated application of the convolutional encoder U should transform the "unencoded" Pauli Z operator acting on the first ancilla qubit to the first stabilizer generator in (1). In order to so, the first application of the encoder U results in an intermediate, unspecified Pauli operator g_1 acting on the m output memory qubits. The second application of the encoder U completes the transformation of the unencoded input Pauli operator to the first generator in (1). (c) The convolutional encoder should similarly transform an "unencoded" Z Pauli operator acting on the second ancilla qubit to the second generator in (1). The shift-invariance of the encoder guarantees that shifts of the unencoded Z Pauli operators transform to appropriate shifts of the generators in (1).

implements the transformation in (2) is the self-loop at the identity memory state with zero logical weight, contradicting our original assumption that the encoder is catastrophic.

III. ONLINE DECODER FOR THE FGG CODE

We could simply use the inverse of the convolutional encoder as the decoding unitary for the FGG code. Quantum turbo codes exploit such a structure for the decoding (the decoding unitary there is actually the inverse of the first convolutional encoder, followed by a deinterleaver, followed by the inverse of the second convolutional encoder) [14], [15]. But we can actually do better when there is just one quantum convolutional code that the receiver needs to decode. The receiver can perform an online decoding unitary where he proceeds with decoding the transmitted qubits as soon as he receives them from the channel output. We illustrate how to do so for the encoding transformation in (3)—the idea is similar to our technique from the previous section.

We first need to determine how the encoder in (3) transforms the logical operators of the code, before determining its corresponding online decoder. One can check with computer programs [26] or by hand that the encoder in (3) transforms the following two unencoded logical operators

$$
\begin{array}{ccc|ccc|ccc}
I & I & X & I & I & I & I & I & I \\
I & I & Z & I & I & I & I & I & I
\end{array},
$$

to the following two encoded logical operators

$$
\begin{array}{ccc|ccc|ccc}
Z & Y & I & X & Z & Y & I & I & I \\
Y & I & Z & X & Z & Y & I & I & I
\end{array}.
$$

Thus, our goal now is to find an online decoder to decode both the above logical operators and the stabilizer generators in (1). This guarantees that the receiver decodes the information qubits properly [27], [9] and that he can perform measurements of the decoded ancilla qubits whose outcomes

he can subsequently feed into a decoding algorithm for the quantum convolutional code [14], [15]. By our same technique as before, we can deduce that the online decoder should perform the following transformation:

$$
\begin{array}{ccccccccc}
I^{\otimes m} & Z & Y & I & & I & I & I & g_1' \\
I^{\otimes m} & Y & I & Z & & I & I & I & g_2' \\
I^{\otimes m} & X & X & X & & I & I & I & g_3' \\
I^{\otimes m} & Z & Z & Z & \rightarrow & I & I & I & g_4' \\
g_1' & X & Z & Y & & I & I & X & I^{\otimes m} \\
g_2' & X & Z & Y & & I & I & Z & I^{\otimes m} \\
g_3' & X & Z & Y & & Z & I & I & I^{\otimes m} \\
g_4' & Z & Y & X & & I & Z & I & I^{\otimes m}
\end{array}. \quad (4)
$$

The input-output commutation relations of the above decoder are more complicated than those from before, but they nevertheless demand that g_1', ..., g_4' should satisfy the following commutation relations:

$$
\{g_1', g_2'\} = [g_1', g_3'] = \{g_1', g_4'\}
$$
$$
= [g_2', g_3'] = \{g_2', g_4'\} = \{g_3', g_4'\} = 0. \quad (5)
$$

A choice of g_1', ..., g_4' that suffices to implement the above transformation is $g_1' = XX$, $g_2' = ZX$, $g_3' = IX$, $g_4' = IZ$, and it is not possible for g_1', ..., g_4' to satisfy the commutation relations in (5) with fewer than two memory qubits. The technique for determining the minimal set of generators satisfying the above commutation relations is exactly the same as the technique used to determine the minimal number of ebits required by an entanglement-assisted code [21], [22] (we detail this in full generality in Ref. [19]).

An online decoder executing the transformation in (4) with the above choice of g_1', ..., g_4' is always non-catastrophic. The line of reasoning is essentially the same as in the previous section, though the cycles for the online decoder that we

should consider are instead of the following form:

$$\begin{array}{cccccccc} h_1 & I & I & I & s_{1,1} & s_{1,2} & l_1 & h_2 \\ \vdots & \vdots & \vdots & \vdots & \rightarrow & \vdots & \vdots & \vdots & \vdots \\ h_p & I & I & I & s_{p,1} & s_{p,2} & l_p & h_1 \end{array},$$

where $h_1, \ldots, h_p$ are two-qubit Pauli operators. Cycles of the above form are relevant here because we are interested in zero-physical weight cycles that have non-zero logical weight. Then by observing the input-output commutation relations in (4), the Pauli operators of the memory states in the cycle should each commute with $g'_1, \ldots, g'_4$. Again, such cycles can only be the zero-logical-weight self-loop at the identity memory state because the only operator commuting with all of $g'_1, \ldots, g'_4$ is the identity operator acting on two qubits.

IV. ENCODER FOR A GRASSL-RÖTTELER CODE

We illustrate our technique on one more example of a quantum convolutional code from Ref. [6] in order to demonstrate how to handle more complicated situations. The example in the second row of Figure 1 of Ref. [6] is a quantum convolutional code generated from the classical convolutional code in (6) (on the next page). This generator leads to the quantum convolutional code in (7). So the encoding unitary should act as follows:

$$\begin{array}{ccccccccc} I^{\otimes m} & Z & I & I & I & X & X & X & X & g_1 \\ I^{\otimes m} & I & Z & I & I & Z & Z & Z & Z & g_2 \\ g_1 & I & I & I & I & X & X & I & I & g_3 \\ g_2 & I & I & I & I & Z & Z & I & I & g_4 \\ g_3 & I & I & I & I & I & X & I & X & g_5 \\ g_4 & I & I & I & I & \rightarrow & I & Z & I & Z & g_6 \\ g_5 & I & I & I & I & I & I & X & X & g_7 \\ g_6 & I & I & I & I & I & I & Z & Z & g_8 \\ g_7 & I & I & I & I & X & X & X & X & I^{\otimes m} \\ g_8 & I & I & I & I & Z & Z & Z & Z & I^{\otimes m} \end{array}, \quad (8)$$

where $g_1, \ldots, g_8$ are Pauli operators acting on the memory qubits. In order for a Clifford encoder preserving the input-output commutation relations to exist, g_3 and g_6 should form an anticommuting pair commuting with all other memory operators, g_4 and g_5 should as well, and g_1, g_2, g_7, g_8 should commute with each other. Thus, the encoder requires six memory qubits at minimum, and a minimal set of generators that satisfies the commutation relations is

$$g_1 = Z_1, \ g_2 = Z_2, \ g_3 = X_3, \ g_4 = X_4,$$
$$g_5 = Z_4, \ g_6 = Z_3, \ g_7 = Z_5, \ g_8 = Z_6. \quad (9)$$

This number of memory qubits is in fact minimal over all different representations of the original quantum convolutional code [19]. (Again, this task of minimizing memory is the same as minimizing the amount of entanglement required for an entanglement-assisted code [21], [22].) Thus, an encoder implementing the transformation in (8) always exists by Lemma 4 in Appendix B of Ref. [23].

Determining a non-catastrophic encoder for this example is a bit more complicated than for our previous examples because the Pauli operators $g_1, \ldots, g_8$ do not form a complete basis for Pauli operators acting on the memory qubits. Nevertheless, we can set some further constraints on the encoder to ensure that the resulting choice is non-catastrophic. First, we find a set of Pauli operators that completes the basis for the Pauli operators acting on the memory. For our example, the following choice suffices: $g_9 = X_1$, $g_{10} = X_2$, $g_{11} = X_5$, $g_{12} = X_6$. Next, we determine that all elements of a cycle of zero physical weight with non-zero logical weight have the following form:

$$h_i \quad s_{i,1} \quad s_{i,2} \quad l_{i,1} \quad l_{i,2} \quad \rightarrow \quad I \quad I \quad I \quad I \quad h_{i+1} \ .$$

Thus, all memory states h_i that are part of such a cycle should commute with the last two output rows in (8). They should then commute with the last two input rows in (8) in order to be consistent with the input-output commutation relations of the encoder. This restricts them to have the form $g \otimes Z^{e_1} \otimes Z^{e_2}$ where g is some four-qubit Pauli operator and e_1 and e_2 are binary numbers. We then observe that each transition $I^{\otimes 4} \otimes g \otimes Z^{e_1} \otimes Z^{e_2}$ of a cycle should commute with the seventh and eighth rows of the output part in (8) (under the choice of $g_1, \ldots, g_8$ in (9)), and this further restricts the memory states that are part of such a cycle to have the form $Z^{e_3} \otimes Z^{e_4} \otimes g' \otimes Z^{e_1} \otimes Z^{e_2}$ so that they are consistent with the input-output commutation relations (where g' is a two-qubit Pauli operator). Continuing in this fashion, we can finally determine that states in such a cycle should have the form $Z^{e_3} \otimes Z^{e_4} \otimes I^{\otimes 2} \otimes Z^{e_1} \otimes Z^{e_2}$. We can then eliminate cycles of this form with non-zero logical weight by choosing extra input-output relations for the encoder that are consistent with its input-output commutation relations while forcing the only cycle of zero physical weight to be the self-loop at the identity memory state. Such a choice for our example is as follows:

$$\begin{array}{cccccccccc} X & I & I & I & I & I & I & I & I & I \\ I & X & I & I & I & I & I & I & I & I \\ I & I & I & I & X & I & I & I & I & I & \rightarrow \\ I & I & I & I & I & X & I & I & I & I \\ & I & I & I & I & I & I & Z & I & I & I \\ & I & I & I & I & I & I & I & Z & I & I \\ & I & Z & Z & Z & X & I & Z & I & I & I \\ & X & I & I & I & I & X & I & Z & I & Z \end{array}$$

Any encoder that implements the full transformation specified by the above relations and those in (8) is non-catastrophic. We elaborate our technique in full generality in Ref. [19].

V. SIMULATION RESULTS

Our encoder in (3) gives a dramatic reduction in the quantum memory required to encode the FGG code, and this reduction implies that the running time of a decoding algorithm for this code becomes within reason for computer simulation—it reduces from $O\left(4^{15}N\right)$ to $O\left(4N\right)$ where N is the length of the code. We can construct a quantum turbo code with the FGG encoder playing the role of both the inner and outer encoder [14], [15], and the running time of the decoding algorithm is the same order in N and the memory.

We simulated the performance of this rate-$1/9$ "FGG turbo code" on a depolarizing channel using free software [26] and according to the method outlined in Section VI of Ref. [15]. Figure 2 demonstrates that this turbo code unfortunately does

$$\begin{bmatrix} 1+D+D^4 & 1+D+D^2+D^4 & 1+D^3+D^4 & 1+D^2+D^3+D^4 \end{bmatrix}. \tag{6}$$

$$\begin{array}{cccc|cccc|cccc|cccc|cccc} X & X & X & X & X & X & I & I & I & X & I & X & I & I & X & X & X & X & X & X \\ Z & Z & Z & Z & Z & Z & I & I & I & Z & I & Z & I & I & Z & Z & Z & Z & Z & Z \end{array} \tag{7}$$
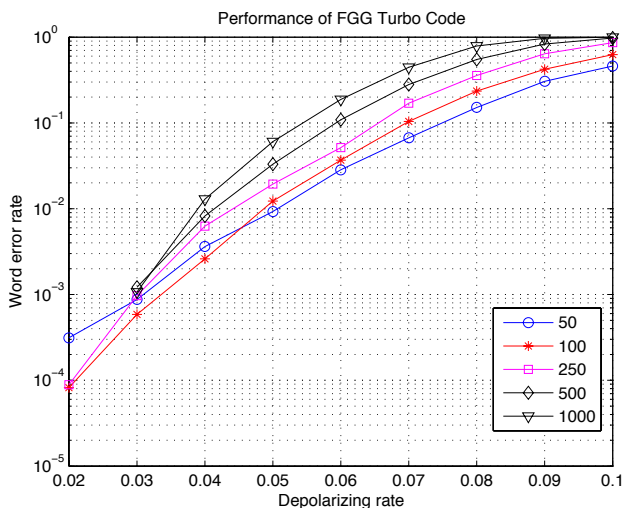
Fig. 2. The performance of the rate-1/9 FGG turbo code on a depolarizing channel. The FGG turbo code does not exhibit a pseudothreshold and thus has inferior performance when compared to the codes from Refs. [14], [15].

not exhibit a pseudothreshold [14], [15], where performance increases as the code grows larger if the noise level is below the pseudothreshold and it decreases with increasing code length if the noise level is higher than the pseudothreshold. This likely has to do with the FGG code's inferior distance spectrum when compared to the codes from Refs. [14], [15]. But the FGG turbo code only uses one memory qubit as opposed to three memory qubits, and the trade-off is a decrease in performance for a decrease in decoding time.

VI. CONCLUSION

We have outlined a technique for determining minimal-memory, non-catastrophic, polynomial-depth quantum convolutional encoders for several example codes. One benefit of our approach is that we can circumvent the complexity issues of the Grassl-Rotteler approach for encoding quantum convolutional codes [4]. MMW acknowledges support from the MDEIE (Québec) PSR-SIIRI international collaboration grant, and MH and SHK acknowledge support from the Iranian Telecommunication Research Center (ITRC). The authors thank M.-H. Hsieh and J. Gütschow for reading the manuscript.

REFERENCES

[1] H. Ollivier and J.-P. Tillich, "Description of a quantum convolutional code," *Phys. Rev. Lett.*, vol. 91, no. 17, p. 177902, Oct. 2003.
[2] G. D. Forney and S. Guha, "Simple rate-1/3 convolutional and tail-biting quantum error-correcting codes," in *IEEE Int. Symp. on Inf. Theory*, September 2005, pp. 1028–1032, arXiv:quant-ph/0501099.
[3] G. D. Forney, M. Grassl, and S. Guha, "Convolutional and tail-biting quantum error-correcting codes," *IEEE Transactions on Information Theory*, vol. 53, pp. 865–880, 2007.
[4] M. Grassl and M. Rötteler, "Noncatastrophic encoders and encoder inverses for quantum convolutional codes," in *Proceedings of the 2006 IEEE International Symposium on Information Theory*, Seattle, Washington, USA, July 2006, pp. 1109–1113, arXiv:quant-ph/0602129.
[5] ——, "Quantum convolutional codes: Encoders and structural properties," in *Proceedings of the Forty-Fourth Annual Allerton Conference*, Allerton House, UIUC, Illinois, USA, September 2006, pp. 510–519.
[6] ——, "Constructions of quantum convolutional codes," in *Proceedings of the 2007 IEEE International Symposium on Information Theory*, Nice, France, June 2007, pp. 816–820.
[7] S. A. A. *et al.*, "Quantum convolutional BCH codes," in *Proceedings of the 10th Canadian Workshop on Information Theory*, 2007, pp. 180–183, arXiv:quant-ph/0703113.
[8] S. A. Aly, A. Klappenecker, and P. K. Sarvepalli, "Quantum convolutional codes derived from Reed-Solomon and Reed-Muller codes," in *Proceedings of the 2007 International Symposium on Information Theory*, Nice, France, June 2007, pp. 821–825, arXiv:quant-ph/0701037.
[9] M. M. Wilde and T. A. Brun, "Entanglement-assisted quantum convolutional coding," *Phys. Rev. A*, vol. 81, no. 4, p. 042333, Apr. 2010.
[10] ——, "Quantum convolutional coding with shared entanglement: General structure," *Quantum Information Processing*, vol. 9, no. 5, pp. 509–540, October 2010, arXiv:0807.3803.
[11] ——, "Unified quantum convolutional coding," in *Proceedings of the IEEE International Symposium on Information Theory*, Toronto, Ontario, Canada, July 2008, pp. 359–363, arXiv:0801.0821.
[12] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Technical Program of the IEEE International Conference on Communications*, vol. 2, Geneva, Switzerland, May 1993, pp. 1064–1070.
[13] S. Benedetto and G. Montorsi, "Unveiling turbo codes: Some results on parallel concatenated coding schemes," *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 409–428, March 1996.
[14] D. Poulin, J.-P. Tillich, and H. Ollivier, "Quantum serial turbo-codes," *IEEE Trans. Inf. Theory*, vol. 55, no. 6, pp. 2776–2798, Jun. 2009.
[15] M. M. Wilde and M.-H. Hsieh, "Entanglement boosts quantum turbo codes," October 2010, arXiv:1010.1256.
[16] H. Ollivier and J.-P. Tillich, "Quantum convolutional codes: Fundamentals," 2004, arXiv:quant-ph/0401134.
[17] M. Houshmand, S. Hosseini-Khayat, and M. M. Wilde, "Minimal memory requirements for pearl necklace encoders of quantum convolutional codes," April 2010, arXiv:1004.5179.
[18] M. Houshmand and S. Hosseini-Khayat, "Minimal-memory realization of pearl-necklace encoders of general quantum convolutional codes," September 2010, arXiv:1009.2242.
[19] M. Houshmand, S. Hosseini-Khayat, and M. M. Wilde, "Minimal-memory, non-catastrophic, polynomial-depth quantum convolutional encoders," May 2011, arXiv:1105.0649.
[20] M. M. Wilde, "Quantum-shift-register circuits," *Physical Review A*, vol. 79, no. 6, p. 062325, June 2009.
[21] M. M. Wilde and T. A. Brun, "Optimal entanglement formulas for entanglement-assisted quantum coding," *Physical Review A*, vol. 77, p. 064302, 2008.
[22] M. M. Wilde, "Logical operators of quantum codes," *Physical Review A*, vol. 79, no. 6, p. 062322, June 2009.
[23] S. Bravyi, D. Fattal, and D. Gottesman, "GHZ extraction yield for multipartite stabilizer states," *Journal of Mathematical Physics*, vol. 47, no. 6, p. 062106, 2006.
[24] M. Grassl, "Convolutional and block quantum error-correcting codes," in *Proceedings of the 2006 IEEE Information Theory Workshop*, Chengdu, China, October 2006, pp. 144–148.
[25] A. J. Viterbi, "Convolutional codes and their performance in communication systems," *IEEE Transactions on Communication Technology*, vol. 19, no. 5, pp. 751–772, October 1971.
[26] M. M. Wilde, "EA-Turbo," http://code.google.com/p/ea-turbo/, September 2010, Matlab and MEX software for characterizing and simulating entanglement-assisted quantum turbo codes.
[27] D. Gottesman, "Stabilizer codes and quantum error correction," Ph.D. dissertation, California Institute of Technology, 1997.