Representation of Convolutional Stabilizer Codes as Clifford Memory Channels

Johannes Gütschow[1,*]

[1] *Institut für Theoretische Physik, Universität Hannover, Appelstraße 2, 30167 Hannover*

(ΩDated: June 30, 2010)

Here we introduce the representation of convolutional stabilizer codes by Clifford memory channels and investigate their error propagation properties. It is found that a given encoder is non-catastrophic if and only if the Channel representing it is strictly forgetful.

*johannes.guetschow(at)itp.uni-hannover.de

I. INTRODUCTION

This report is based on results from the corner report [1].

II. CONVOLUTIONAL STABILIZER CODES

It is possible to extend our notion of stabilizer codes to codes which allow for an overlap between the individual steps of the encoding operation. Their encoding operations can be described as quantum memory channels in the following way: some of the output qubits of the nth encoding step will be used as inputs in the $n+1$th step of the encoding, thus the "blocks" overlap. These qubits are the memory system of the memory channel describing the encoding. The encoding operation is the same in every step (neglecting initialization and finalization at this point), thus every step is described by the same channel. Codes with these properties are called convolutional codes. Here we will only look at convolutional stabilizer codes, because they can be described in an efficient way by Clifford memory channels. Convolutional stabilizer codes where first introduced in 1998 by H.F.Čhau [2]. Later they were studied and improved by a different researchers [3–6]. first we want to consider the question why it is worth to look at convolutional codes.

A. Reasons to use convolutional codes

In the processing of classical information convolutional codes are widely used, because they offer a better ratio of performance to complexity than block codes. The performance is measured by the rate and the error-correction properties. The complexity is measured by the complexity of the error-detection and correction algorithm. It is not yet proven if quantum convolutional codes outperform quantum block codes in the same sense, but examples, e.g. in [7], give evidence that they actually do. These examples show concrete advantages in the following code properties.

Rate: In general quantum convolutional codes need fewer physical qubits to protects the same number of logical qubits against errors, than comparable block codes.

Complexity of error-correction algorithm: The algorithms used to determine the error that occurred are easier than for block codes. The algorithms are classical.

Performance: Quantum convolutional have a better relation of performance to complexity that block codes.

(TODO: this list needs a citation) The paper [7] presents different examples which confirm these claims. On the other hand the same paper presents block codes, the so called tail-biting codes, which are obtained by the application of convolutional codes to blocks of qubits with "periodic boundary conditions". The properties of these block codes are similar to the properties of convolutional codes.

Unfortunately convolutional codes also carry disadvantages. Because information is transmitted from one block to the next, errors can spread as well. Depending on the encoding algorithm "catastrophic" errors can occur. Catastrophic errors happen during the process of encoding (or transmission or decoding) and affect only a few qubits at first. During the encoding (or decoding) the spread without bound. These errors can only be corrected with operations of unbounded support (i.e. they act on an unbounded number of qubits). If we consider transmissions of finite length, the support of the operation is only bounded by the length of the transmission. There is one exception of this behavior. The support of a catastrophic errors can also stay finite but the localization area moves in an unbounded manner, i.e. the error will always affect the qubits at the end of the transmission. In both cases the error can only be corrected at the end of the transmission. This implies that the decoding can start only after the end of the transmission. Therefore the delay between receiving the first qubit and decoding it is as long as the transmission itself. Furthermore the memory requirement on the decoder side is of the order of the length of the encoded message. Consequently these errors have to be avoided. The next section covers the question if and how this can be accomplished. The reasoning follows "Quantum Convolutional Codes: Fundamentals" [8] by Harold Ollivier and Jean-Pierre Tillich. For the encoding and decoding operations we use our new approach, the description of the encoding and decoding operations as Clifford memory channels.

B. Definition and formalism

A convolutional stabilizer coded is defined by the generators of its stabilizer group just like a block stabilizer code. The stabilizer generators of a convolutional code also exhibit some kind of block structure but with overlapping blocks. In block codes the stabilizer generators (and thus the stabilizer group) of every block are the same. The convolutional codes we consider should be translation invariant. therefore we generate the stabilizer generators of every block by translates of a single set of basic stabilizer generators. We come to the following definition:

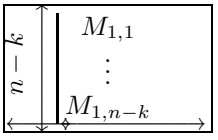
Definition II.1 ([9]). *A (n, k, m) convolutional stabilizer code is given by the stabilizer group $\mathcal{S}$*

$$\mathcal{S} = \mathrm{sp}\{M_{j,i} = I^{\otimes j\cdot n} \otimes M_{0,i},\, 1 \leq i \leq n-k,\, 0 \leq j\}, \tag{1}$$

where $M_{0,i} \in \mathcal{G}_{n+m}$ and all $M_{i,j}$ are independent of each other. Furthermore all generators have to commute. Of course strictly speaking all the generators need to have the same length (act on the same space). We can achieve this by tensoring all of the $A_{j,i}$ with identities I from the right until they have the same support. For some set of operators $\{A_i\}$ $\mathrm{sp}\{A_i\}$ denotes the set of operators that can be generated by multiplication of A_i. This is the multiplicative group generated by A_i.

A code of this form encodes k logical qubits per n physical qubits. Every "block" itself has an output of $n+m$ qubits. n of those are output qubits, while m are passed on to the next step. The matrix form of the stabilizer code is one sided infinite an has the form

$$M = \begin{pmatrix} \boxed{\begin{matrix} M_{0,1} \\ \vdots \\ M_{0,n-k} \end{matrix}} & & \\ & \boxed{\begin{matrix} M_{1,1} \\ \vdots \\ M_{1,n-k} \end{matrix}} & \\ & & \ddots \end{pmatrix}. \tag{2}$$



Every row of the matrix represents one generator $M_{j,i}$, every column represents one output qubit. The width of the rectangles represents the maximal possible support, i.e. the positions where the operators may differ from the identity. of the stabilizer generators. Every box contains the same set of operators. Due to the shift of the boxes by multiples of n the resulting stabilizer generators are copies of one set of operators $M_{0,i}$ shifted by multiples of n. Because the generators of different locks overlap by m qubits we can not split the code space into blocks but have to consider the space stabilized by all stabilizer generators at once. If we allow $m = 0$ we have a block code and we can consider each block separately. To use the code in a real world application the matrix M would have to be expanded by a bounded number of operators not fitting into this pattern. They are needed for the initialization and the termination of the transmission. We neglect them here, because in the limit of long transmission they do not play a role for the rate an error correction capabilities of the code.

We now use the invariance by n qubit translations of the generators to find a shorter description. Let us define the shift operator

$$D[A] = I^{\otimes n} \otimes A. \tag{3}$$

Again we pad the operators with tensored I. We can now describe the generators by

$$M_{j,i} = D^j[M_{0,i}],\, 1 \leq i \leq n-k,\, 0 \leq j. \tag{4}$$

Using this we only need to consider the first $n-k$ generators. All others are obtained by repeated application of D. The shift can be generalized by the use of polynomials in D. We define the action of a polynomial $P(D) = \sum_j \alpha_j D^j$, $P(D) \in \mathcal{P}(D)$ on $A \in \mathcal{G}$ by

$$P(D)[A] = \prod_j \alpha_j D^j[A]. \tag{5}$$

This definition relies critically on the fact that all copies of A shifted by D^j commute, i.e. $[D^i[A], D^j[A]] = 0\ \forall i, j \in \mathbb{N}$. If this were not the case the product would not be well defined. If, during some calculation, we end up with a result that contains negative powers of D, we apply D^{-j}, where j is the lowest exponent occurring.

We will now shift to a phase-space description of the elements of $\mathcal{G}$. To avoid matrices with tuples as coefficients we split the matrix M in two parts by replacing the vectors (ξ_+, ξ_-) by $(\xi_+|\xi_-)$.

Example II.2.

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{pmatrix} \mapsto \left(\begin{array}{ccccc|ccccc} 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \end{array}\right) \tag{6}$$

$\Diamond$

To further simplify our notation we adept the "algebraic Fourier transformation" used in the phase-space theory of CQCA [10] to our setting of translation invariance by n qubits. Using

$$\hat{\xi}_+(j) = \sum_i \xi_+(i \cdot n + j) D^i,\ 0 < j \leq n \tag{7}$$

we map the components of a phase-space vector to tuples with n coefficients which are Laurent polynomials in D over $\mathbb{F}_2$, thus elements of $\mathcal{P}$. We can also write $\hat{\xi} = \sum_i \hat{\xi}^{(i)} D^i$, where $\hat{\xi}^{(i)}$ are n-dimensional vectors with coefficients from $\mathbb{F}_2$. In the following we will work with the transformed phase-space vectors exclusively and omit the "hat".

Example II.3. *Let us illustrate this notation using the* $(5, 1, 2)$ *convolutional code introduced in [8]. The stabilizer group of this code is generated by*

$$\begin{aligned} M_{0,1} &= Z\,X\,X\,Z\,I\,I\,I \\ M_{0,2} &= I\,Z\,X\,X\,Z\,I\,I \\ M_{0,3} &= I\,I\,Z\,X\,X\,Z\,I \\ M_{0,4} &= I\,I\,I\,Z\,X\,X\,Z \\ M_{j,i} &= D^j[M_{0,i}],\ 0 \leq j \end{aligned} \tag{8}$$

and translates by $n = 5$ *qubits. Using the polynomial notation we get*

$$\widehat{M} = \left(\begin{array}{ccccc|ccccc} 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & D & 0 & 1 & 0 & 0 \\ D & 0 & 0 & 0 & 1 & 0 & D & 0 & 1 & 0 \end{array}\right). \tag{9}$$

$\Diamond$

All the basic stabilizer generators have to commute pairwise and also with all their translates by multiples of n qubits. In the polynomial notation it is easy to check this. Two Pauli operators $A \widehat{=} (\xi_+, \xi_-)$ and $B \widehat{=} (\eta_+, \eta_-)$ commute, if and only if the symplectic form vanishes for their phase-space vectors $(\xi_+\eta_- + \eta_+\xi_- = 0)$. Using the polynomial notation we get

$$AB = BA \Leftrightarrow \sum_i \xi_+^{(i)} \eta_-^{(i)} - \xi_-^{(i)} \eta_+^{(i)} = 0. \tag{10}$$

For operators shifted by $n \cdot r$ resp. $n \cdot s$ we have

$$D^r[A] D^s[P] = D^s[P] D^r[A] \Leftrightarrow \sum_l \xi_+^{(l+s)} \eta_-^{(l+r)} - \xi_-^{(l+s)} \eta_+^{(l+r)} = 0. \tag{11}$$

The right hand part of Equation (11) is the D^{s-r} coefficient of $\xi_+(D)\eta_-(1/D) - \xi_-(D)\eta_+(1/D)$. We define the generalized commutator to be

$$\forall r, s,\ D^r[A] D^s[P] = D^s[P] D^r[A] \Leftrightarrow \xi_+(D)\eta_-(1/D) - \xi_-(D)\eta_+(1/D) = 0. \tag{12}$$

The generalized commutator answers the question if two operators and all their translates by multiples of n qubits commute. Using the generalized commutator we only need to check the commutation relations of the first block of generators (the basic generators) to find the commutation relations for all generators.

C. Encoded Pauli operators

Just like in the case of block codes we want to determine the operators that act on the encoded qubits like the Pauli operators on the unencoded qubits. The operators have to be in $N(\mathcal{S}) \setminus \mathcal{S}$ and obey conditions (3.10) - (3.13) in [11].Together with the generalized commutator (12) this suffices to determine the encoded Pauli operators.

First we use the Gaussian elimination algorithm to bring M in the standard form

$$M_{\text{std}} = \left(\begin{array}{ccc|ccc} \overbrace{A(D)}^{r} & \overbrace{B(D)}^{n-k-r} & \overbrace{C(D)}^{k} & \overbrace{E(D)}^{r} & \overbrace{F(D)}^{n-k-r} & \overbrace{G(D)}^{k} \\ 0 & 0 & 0 & J(D) & K(D) & L(D) \end{array}\right) \begin{array}{l} \} r \\ \} n-k-r \end{array}. \tag{13}$$

$A(D)$ and $K(D)$ are diagonal matrices of full rank. r is the rank of the X part of M. Using the ansatz

$$\bar{X} = (U_1(D), U_2(D), U_3(D) | V_1(D), V_2(D), V_3(D)) \tag{14}$$
$$\bar{Z} = (U_1'(D), U_2'(D), U_3'(D) | V_1'(D), V_2'(D), V_3'(D)) \tag{15}$$

one can find the encoded operators. For $\bar{X}$ we get

$$U_1(D) = 0 \tag{16}$$
$$U_2(D) = L^T(1/D)K^{-1}(1/d)\Lambda(D) \tag{17}$$
$$U_3(D) = \Lambda(D) \cdot I \tag{18}$$
$$V_1(D) = (U_2(D)F^T(1/D) + \Lambda(D)G^T(1/D))A^{-1}(1/D) \tag{19}$$
$$V_2(D) = 0 \tag{20}$$
$$V_3(D) = 0. \tag{21}$$

$\bar{Z}$ fulfills

$$U_1'(D) = 0 \tag{22}$$
$$U_2'(D) = 0 \tag{23}$$
$$U_3'(D) = 0 \tag{24}$$
$$V_1'(D) = C^T(1/D)A^{-1}(1/D)/\Lambda(1/D) \tag{25}$$
$$V_2'(D) = 0 \tag{26}$$
$$V_3'(D) = I/\Lambda(1/D). \tag{27}$$

The polynomial $\Lambda(D)$ can be freely chosen. But as we want to ensure that only finite Laurent series $\bar{X}$ and that the encoded operators are as "short" as possible. We call the polynomial the *conditioning polynomial* and define it to be

Definition II.4 ([9])**.** *The conditioning polynomial $\Lambda(D)$ of a quantum convolutional code is the non-zero polynomial of minimal degree such that (16)-(21) only contain finite Laurent series.*

The conditioning polynomial always exist. Therefore we can always find a set of encoded X operators which have finite support and respect the invariance by n qubit shifts. Because of (27) this only holds for the encoded Z operators, if Λ is a monomial. Thus given encoded X operators we can not always find Z operators that respect the translations and have finite support.

Example II.5. *The $(5, 1, 2)$ code introduced in [8] has the standard form*

$$M_{std} = \left(\begin{array}{ccccc|ccccc} D & 0 & 0 & 0 & 1 & 0 & D & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1+D & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & D & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & D & 0 & 1 & 0 & 0 \end{array}\right). \tag{28}$$

The encoded Pauli operators are

$$\bar{X} = (0, 0, 0, 0, 1 | 0, 1, 1, 0, 0) \tag{29}$$
$$\bar{Z} = (0, 0, 0, 0, 0 | D, 1, 1, 1, 1). \tag{30}$$

$\Diamond$

The encoding operation is of course a quantum channel. It is convenient to take the encoding of one "block" as on use of the encoding channel. As we pass on qubits from one step of the encoding to the next we deal with a memory channel. All encoding steps are the same, so our channel can be concatenated and used for arbitrary many steps of the encoding procedure. This is stated more precisely in the following theorem:

Theorem II.6. *For every (n, k, m) convolutional stabilizer code one can find an encoding operation which is described by a the concatenation of a reversible Clifford memory channel. The channel has n input and output qubits and uses m qubits of memory. One use of the channel corresponds to on "block" in the encoding.*

Proof. The encoding of the data qubits is clearly done by a Clifford channel, because the Pauli matrices on the input side correspond to encoded operators which are again Pauli products. The encoding channel, described in the Heisenberg picture, maps the encoded operators $\bar{X}_i$ and $\bar{Z}_i$ to the unencoded ones. The only difficulty we have to overcome is the fact that the encoding needs more then one step. The image of the encoded operators on the memory input of the encoding operation is not given by the code itself, but a property of the encoding channel. But we are free to choose these images as long as the resulting channel is unitary and gives the right mapping between the encoded Pauli matrices and the input Pauli matrices when concatenated.

Our description of the encoding operation includes the $(n-k)$ ancilla qubits on the input side. Pauli products localized on the ancillas on the input side correspond to elements of the stabilizer group on the encoded side. The stabilizer group is generated by $(n-k)$ Pauli products. We map these to the $(n-k)$ commuting single cell matrices, e.g. the X matrices, on the ancillas. To each stabilizer generator we can find an operator that anticommutes with it while commuting with all other generators and their anticommuting counterparts. We map them to the remaining basis elements on the ancillas.

A reversible Clifford channel with n input and output qubits and m memory cubits is described by a $2(n+m) \times 2(n+m)$ matrix with binary coefficients. We therefore have $4(n+m)^2$ coefficients to determine. With the data qubits we fixed $2k2(n+m)$ coefficients. The ancillas fix another $2(n-k)2(n+m)$ coefficients. Due to the convolutional structure we also fixed the images of the memory qubits which are governed by another $2m2(n+m)$ coefficients. Thus the matrix of the encoding channel is completely determined. However in this process we had some freedom to choose the images of the stabilizer generators and the encoded Pauli matrices on the memory input. If we change these images,we just have to change the images of the memory outputs accordingly and we still have the same global operation. Thus we can freely choose a reversible operation on the memory which has $4m^2$ coefficients (A reversible operation is described by a symplectic matrix, so some of the coefficients are fixed by our choice of the others.). □

D. Error propagation properties

As mentioned in section II A convolutional codes can have bad error propagation properties. This is also apparent in Section II C where we showed that we can not always find encoded Z operators which respect the translation invariant structure and have finite support. An error operator which includes a Z tensor factor on one of the input qubits corresponds to an operator with unbounded support on the output side. This error could only be corrected after the end of the transmission. We call these errors "catastrophic". But we are interested in the case, where we can start the decoding after a delay independent of the length of the message to be transmitted. We call this type of decoding operation an "on-line" decoder. It exist if and only if the encoder does not exhibit catastrophic errors.

Let us now develop an accurate definition of a catastrophic error. In [8] a catastrophic error is defined as an error, which affects $O(1)$ qubits before the end of the decoding operation but can only be corrected by an operation which has a localization area that grows with q if we encode $k \cdot q$ qubits. If and only if an encoder allows such errors it is said to be catastrophic. We can think of the encoding of a convolutional code as a circuit build up by the concatenation of the same unitary shifted by multiples of n qubits as shown in Figure 1. In practice we need some operations for initialization and termination of the encoding, which we omit here. In [8] it is claimed that an encoder is free of catastrophic errors if and only if it can be converted into a circuit of finite depth as shown in Figure 2.

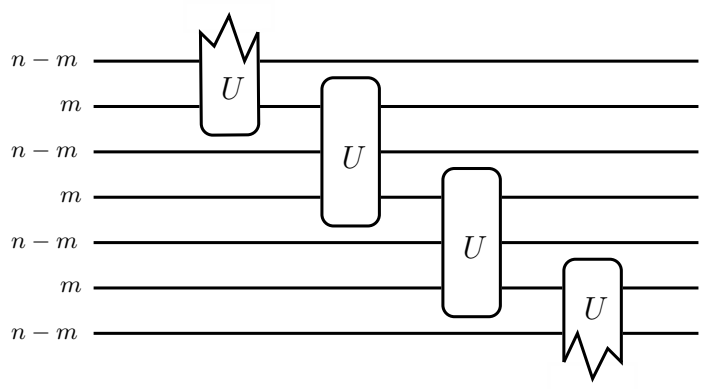


FIG. 1. Encoding circuit of a convolutional code. Every wire represents multiple qubits. The number is noted on the left.

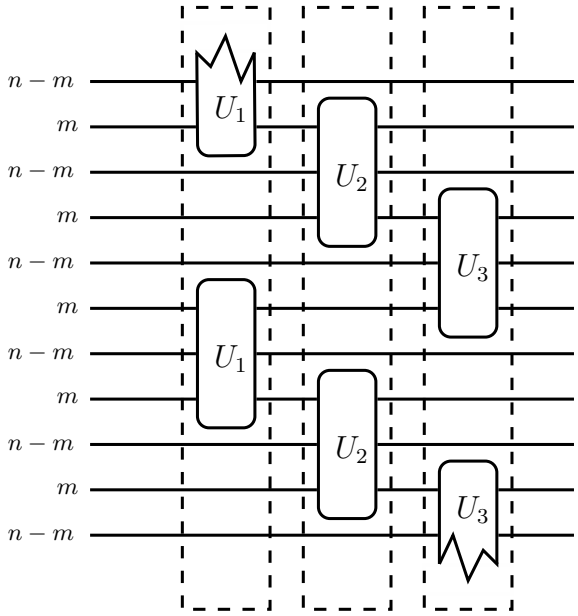


FIG. 2. Finite depth encoding circuit of a convolutional code. The stroked boxes each denote one layer of unitary operations, which can be applied in the same time, as they commute.

Conjecture II.7 ([8]). *An encoder of a quantum convolutional code is non-catastrophic if and only if the encoding operation C8q) can be decomposed in the following way for large q:*

$$C(q) = \tilde{T}_{erm}(q) \left(\prod_{i=0}^{\lfloor q/l_t \rfloor} D^{il_t}[U_t] \right) \cdots \left(\prod_{i=0}^{\lfloor q/l_1 \rfloor} D^{il_1}[U_1] \right) \tilde{I}_{nit}(q) \tag{31}$$

$\tilde{T}_{erm}(q)$ and $\tilde{I}_{nit}(q)$ are modified initialization and termination operations which can depend on q but whose localization area is bounded independent of q. $\{U_j\}_j$ is a finite set of unitary operations with bounded localization and independent of q. Furthermore $D^i[U_j]$ and $D^{i'}[U_j]$ commute. The l_j are also independent of q.

The proof of this conjecture presented in [8] contains a small error. A counter example to the necessity of the decomposition for the encoder to be non-catastrophic can be found. The part of the proof that shows that the decomposition is sufficient to exclude catastrophic errors however is correct.

The proof of the necessity of the decomposition starts from the original encoding circuit. An error is introduced that creates a local reordering of the encoding steps. This error is corrected by an operation with bounded support. At this point is is wrongly assumed that the localization area is independent of q if the encoder is non catastrophic. Following the definition in [8] catastrophic errors can be corrected by operations with operators which have a localization are whose size is independent of q. However the position of the operator can still depend on q. This only happens in pathological examples, but still the proof does not hold as stated. In II.8 we present such an example.

Example II.8 (Counterexample to the claim II.7 in [8])**.** *We start with an arbitrary $[n, k, d]$ block stabilizer code with stabilizer group $\mathcal{S}$ and encode it as a $(n, k, 1)$ convolutional code. We pad the generators of $\mathcal{S}$ by a I on the memory and get the basic generators of length $n + m = n + 1$ for the convolutional code. The remaining generators are obtained by n qubit shifts of the basic generators as usual. Figure 3 shows the encoding circuit.*

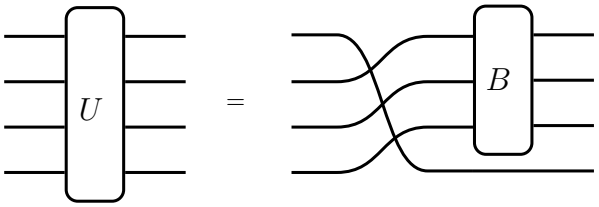


FIG. 3. Encoding circuit of a block code encoded as a convolutional code. The memory qubit is just passed on to the next step of the encoding. The operation B is the encoder of the block code.

Concatenating several steps of the encoding operation it is easy to observe that one qubit is not included in the process of encoding (see Figure 4).

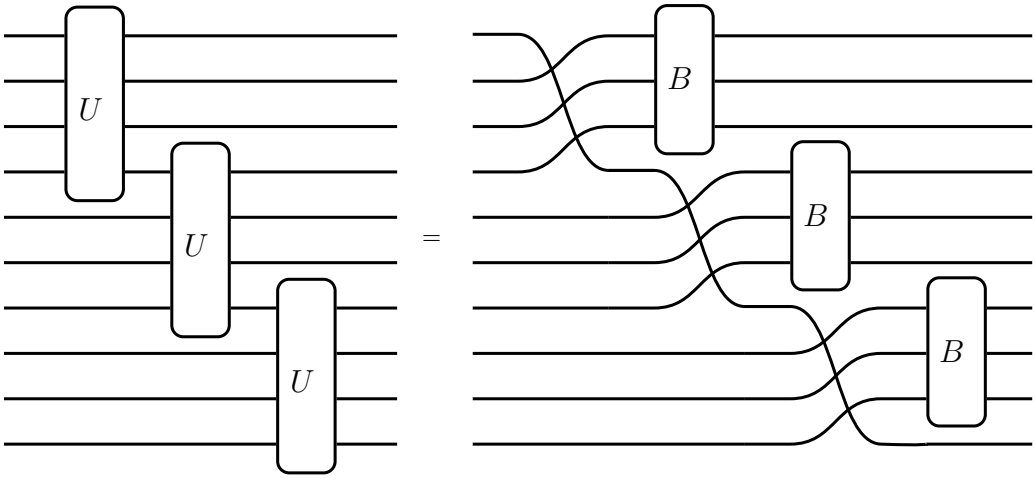


FIG. 4. Concatenation of the example encoder. One qubit does not take part in the encoding, but is just passed on from one step to the next unchanged.

An error affecting this qubit would have the same effect on the last output of the encoding procedure even after infinitely many steps of the encoding. The qubit and also the error affecting it are just transported to the end of the output stream. This transport is only possible, if information can be passed on from the first to the last step of the encoding. This is not possible in a finite depth circuit like the one proposed in II.7. By the above definition of a catastrophic error this error is not catastrophic, because it can be corrected with an operation of bounded length. This disproves Conjecture II.7.

Of course the presented encoder is an example without practical use, just constructed to refute Conjecture II.7. The convolutional encoder encodes a block code and thus gives no advantage. Furthermore catastrophic errors can only occur on one special qubit that is not used in the code. Nevertheless it is necessary to include this case in the definition of catastrophic errors to fix the proof of Conjecture II.7.

Definition II.9. *Given an (n, k, m) convolutional code used to encode $n \cdot q$ qubits. A catastrophic error is an error that affects $O(1)$ qubits before the end of encoding, but influences some outputs for arbitrarily large q. Thus the error can only be corrected after the end of the transmission. An encoder which exhibits such errors is called catastrophic. Otherwise it is called non-catastrophic.*

This definition include the old definition as well as errors like the one in the example, which could easily be extended to more than one memory qubit. Another way to understand the new definition is the idea that in the absence of catastrophic error propagation finitely localized inputs can only affect outputs in a finite area independent of the number of iterations. Simply from considering the causality it is apparent, that an encoder can not be implemented by finite depth circuit if this condition is not fulfilled. Using this new definition we ruled out the counter example and the proof of the Conjecture II.7 holds.

Using the representation of convolutional encoders by Clifford memory channels we will now study which channels correspond to catastrophic encoders and which to non-catastrophic encoders.

Theorem II.10. *A convolutional encoder is non-catastrophic if and only if the Memory channel representing it is strictly forgetful.*

Proof. By Theorem A.1 a channel is strictly forgetful if and only if every finitely localized observable on the output system is mapped to an observable on the input system which is finitely localized independent of the number of channel uses n for large n. Obviously this is equivalent to the definition of a non-catastrophic encoder. □

Remark II.11. *The proof uses neither the stabilizer structure of the code, not the Clifford property of the channel. Thus it holds for general convolutional encoders and the channels that implement them. This has implications for every operation that can be described by the, in general infinite, concatenation of a memory channel. If and only if this channel is strictly forgetful then by Theorems II.10 and II.7 it can be implemented by a finite depth circuit using only operations of finite support.*

E. Existence of non catastrophic encoders

As we have seen in the last section using the method presented in [8] we can not find non-catastrophic encoders for every code. However Markus Grassl and Martin Rötteler proved in [12] that one can always find a subcode of a given code which allows a non-catastrophic encoding.

1. *Example of a $(3, 1, 1)$ code*

We will now demonstrate the new representation of convolutional stabilizer codes with a simple example. The example code has no error correction capabilities. To actually correct errors more qubits are needed which would make the phase-space matrices occurring in our example large and rather hinder then facilitate understanding of the formalism.

The basic stabilizer generators are given by

$$\begin{pmatrix} X & X & X & I & I & I & I & I & \cdots \\ I & Z & Z & X & I & I & I & I & \cdots \\ I & I & I & X & X & X & I & I & \cdots \\ I & I & I & I & Z & Z & I & I & \cdots \\ & & \vdots & & & \vdots & & & \ddots \end{pmatrix}. \tag{32}$$

In the polynomial notation we have

$$\left(\begin{array}{ccc|ccc} 1 & 1 & 1 & 0 & 0 & 0 \\ D & 0 & 0 & 0 & 1 & 1 \end{array}\right). \tag{33}$$

Using gaussian elimination we get

$$\left(\begin{array}{ccc|ccc} D & 0 & 0 & 0 & 1 & 1 \\ 0 & D & D & 0 & 1 & 1 \end{array}\right). \tag{34}$$

To determine $\bar{X}$ and $\bar{Z}$ we use the method introduced in Section II C:

$$A(D) = \begin{pmatrix} D & 0 \\ 0 & D \end{pmatrix}, C(D) = \begin{pmatrix} 0 \\ D \end{pmatrix}, E(D) = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}, G(D) = \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \tag{35}$$

$B(D)$, $F(D)$, $J(D)$, $K(D)$ and $L(D)$ are 0×0 matrices and thus do not occur in this example because $n - k - r = 0$. This yields

$$U_1 = 0,\ U_3 = \Lambda(D),\ V_1 = \Lambda(D)(D\,D),\ V_3 = 0. \tag{36}$$

Now U_2 and V_2 do not appear, because $n-k-r = 0$. The same holds for U_2' and V_2'. As Λ is defined to be the polynomial of minimal degree such that all U_i and V_i contain only finite Laurent series, we obtain $\Lambda = 1$ and

$$U_1' = U_3' = 0,\ V_1' = (0\,1),\ V_2' = 1. \tag{37}$$

Using this we determine the encoded Pauli matrices to be

$$\bar{X} = (0\ \ 0\ \ 1 \,|\, D\ \ D\ \ 0\ ) = I\,I\,X\,Z\,Z\,I \tag{38}$$

$$\bar{Z} = (0\ \ 0\ \ 0 \,|\, 0\ \ 1\ \ 1\ ) = I\,Z\,Z\,I. \tag{39}$$

We are now searching for a channel whose concatenation maps the encoded Pauli matrices (38) to the unencoded Pauli matrices on the input system. It is easier to find a channel the implements the opposite mapping, because we use the Pauli basis for the phase-space matrices and can thus directly write down their images, the partly encoded operators. At the end we just have to calculate the inverse of the symplectic mapping we find to get the encoding channel. Figure 5 shows a diagram of the channel used for one step of the encoding. From the encoded Pauli matrices we can deduce some

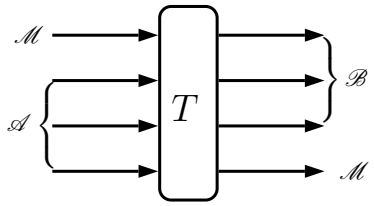


FIG. 5. Diagram of the reversible channel encoding the $(3, 1, 1)$ code

properties of the channel. These are differ depending on our choice which input qubit will be the data qubit and which one will be the memory. But the resulting channels only differ by a permutation of columns of the phase-space matrix. In the following we assume that the data qubit is the third qubit on the input side. The first input qubit is the memory output, the last output qubit the memory output. This choice is the only sensible one, if we want to concatenate the channel. See also Figure 5. With this choice the operator $I\,I\,X\,I\cdots$ on the input side is mapped to $I\,I\,X\,Z\,Z\,I$ on the output. The same has to hold for the operators shifted by multiples of 3 qubits. The first three output cubits remain unchanged after the first use of the inverse encoding channel T^{-1}. They have to match the first three qubits of the encoded operator after one channel use. The same holds for Z. So far we have

$$\begin{array}{llll} I & I & X & I \to I\ \ I\ \ X\ \ ? \\ I & I & Z & I \to I\ \ Z\ \ Z\ \ ?. \end{array} \tag{40}$$

We chose the last tensor factor of the partially encoded Z operator to be I. With this choice the Z operator is fully encoded after the first step. The encoded X has a larger support, thus information has to be passed on to the next step of the encoding. We can choose X, Y or Z to do so. We choose Z and the concatenation immediately gives us the condition that the image of $Z\,I\,I\,I$ has to be $Z\,Z\,I\,?$. The fourth qubits can be chosen to be the identity, as the encoding of X is already finished and the image of $I\,I\,X\,I$ equals $\bar{X}$ after two encoding steps. We know fixed the following mappings:

$$\begin{array}{llll} I & I & X & I \to I\ \ I\ \ X\ \ Z \\ I & I & Z & I \to I\ \ Z\ \ Z\ \ I \\ Z & I & I & I \to Z\ \ Z\ \ I\ \ I \end{array} \tag{41}$$

The images of all input basis elements have to obey

$$\{\bar{X}_i, \bar{Z}_i\} = 0 \tag{42}$$
$$[\bar{X}_i, \bar{Z}_j] = 0 \quad i \neq j \tag{43}$$
$$[\bar{X}_i, \bar{X}_j] = 0 \tag{44}$$
$$[\bar{Z}_i, \bar{Z}_j] = 0. \tag{45}$$

$\bar{X}_3 = \bar{X}$ and $\bar{Z}_3 = \bar{Z}$ are the encoded Pauli matrices of the data qubit. The Equations (42)-(45) have multiple solutions. It suffices to find one. As said above we have some freedom in building our encoder. (which is best done using a computer algebra system unless you have a lot of time to kill). The X operators on the ancillas are mapped to the stabilizer generators. We get the following complete description of the channel.

$$\begin{array}{cccccccc}
X & I & I & I & \to & X & I & I & I \\
Z & I & I & I & \to & Z & Z & I & I \\
I & X & I & I & \to & I & Z & Z & X \\
I & Z & I & I & \to & I & I & I & Z \\
I & I & X & I & \to & I & I & X & Z \\
I & I & Z & I & \to & I & Z & Z & I \\
I & I & I & X & \to & X & X & X & I \\
I & I & I & Z & \to & I & Z & I & I.
\end{array} \tag{46}$$

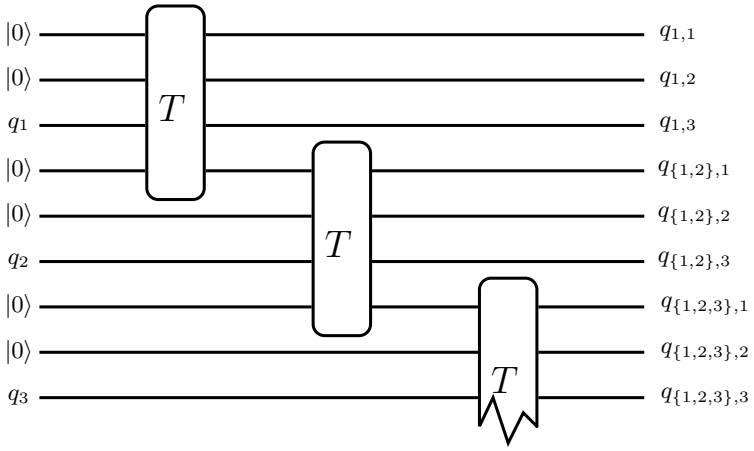


FIG. 6. Implementation of the $(3, 1, 1)$ code by a concatenation of reversible channels.

If we concatenate the channel as shown in Figure 6 the resulting encoded qubits are

$$\bar{X}_i = I^{\otimes 3i}\, I\, I\, X\, Z\, Z\, I\, \cdots \tag{47}$$
$$\bar{Z}_i = I^{\otimes 3i}\, I\, Z\, Z\, I\, \cdots. \tag{48}$$

We determined the encoder inverse, thus we still have to invert our result to obtain the actual encoding channel. By the inversion and multiplication of Equations (46) we get

$$\begin{array}{cccccccc}
X & I & I & I & \to & X & I & I & I \\
Z & I & I & I & \to & Z & I & I & Z \\
I & X & I & I & \to & X & Z & X & X \\
I & Z & I & I & \to & I & I & I & Z \\
I & I & X & I & \to & I & Z & X & I \\
I & I & Z & I & \to & I & I & Z & Z \\
I & I & I & X & \to & I & X & Z & I \\
I & I & I & Z & \to & I & Z & I & I.
\end{array} \tag{49}$$

Now on the left hand side we have observables of the output side which are mapped to observables on the input side which are on the right hand side. The phase-space matrix of the channel is

$$t = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}. \tag{50}$$

The 2×2-matrix in the upper right corner

$$t_{|\mathfrak{M}} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \tag{51}$$

describes the restriction of the channel to the memory system. The matrix is nilpotent, so the channel is strictly forgetful

But not all of the channels which implement an encoding of our example stabilizer code are strictly forgetful. Our first channel is only strictly forgetful because we determined the encoded Pauli operators by an algorithm that gives us a non-catastrophic encoder. If we change the encoded Pauli operators we can also find a non-forgetful channel, which encodes the same stabilizer code. We choose the inverse encoder to be

$$\begin{array}{cccccccc} X & I & I & I & \to & X & I & I & I \\ Z & I & I & I & \to & Z & Z & X & Z \\ I & X & I & I & \to & I & Z & Z & X \\ I & Z & I & I & \to & I & I & X & I \\ I & I & X & I & \to & X & I & I & X \\ I & I & Z & I & \to & I & I & X & Z \\ I & I & I & X & \to & X & X & X & I \\ I & I & I & Z & \to & I & Z & I & I. \end{array} \tag{52}$$

The phase-space matrix of the encoder is

$$s = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{pmatrix} \tag{53}$$

and its restriction to the memory is

$$s_{|\mathfrak{M}} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}. \tag{54}$$

$s_{|\mathfrak{M}}$ is a projector and thus not nilpotent. Therefore the channel is not forgetful. Accordingly the some of the encoded Pauli matrices have infinite an localization length:

$$\bar{X}^{(i)} = I^{\otimes 3i}\, X\, I\, I\, X\, I \cdots \tag{55}$$

$$\bar{Z}^{(i)} = I^{\otimes 3i}\, I\, I\, X\, Z\, Z\, X\, Z\, Z\, X\, Z\, Z\, X \cdots. \tag{56}$$

If during the encoding an error occurs, that corresponds to a Z operation on one of the input qubits the error will have an effect even after infinitely many time steps. But even without an error we can only start the decoding after the transmission is over, because the encoded Z operators are not localized. Not even an approximate decoding or correction would be possible, because the effect of the error does not decrease with channel uses. Waiting for the end of the transmission would need a huge amount of quantum memory, which again has to be protected against errors. Thus using such a code would be very costly in time and memory.

Remark II.12. *If we are only concerned with the forgetfulness of a channel it is sufficient to consider the inverse. If and only if a reversible Clifford channel is strictly forgetful its inverse is strictly forgetful. This is proven in Theorem A.2 in the appendix.*

Appendix A: Some General Results for Strictly Forgetful Channels

Here we present two results we used to derive the results on error propagation.

Theorem A.1. *Let $T : \mathfrak{B} \otimes \mathfrak{M} \to \mathfrak{M} \otimes \mathfrak{A}$ be a channel and $T_n : \mathfrak{B}^{\otimes n} \otimes \mathfrak{M} \to \mathfrak{M} \otimes \mathfrak{A}^{\otimes n}$ its n-fold concatenation. Then T is strictly forgetful if and only if every finitely localized observable on the output system $\mathfrak{B}^{\otimes n} \otimes \mathfrak{M}$ is mapped to an observable which is finitely localized on the input system $\mathfrak{M} \otimes \mathfrak{A}^{\otimes n}$, the localization area being independent of n for large n.*

*Proof.***:**

⇒: Let T be strictly forgetful. Let us assume there is an observable with finite localization area on the output system whose image is not independent of n for arbitrarily large n. This implies that after arbitrarily many steps the memory still transports information about the observable. Thus the image of the observable on the memory system not be the identity after finitely many steps n. But the the condition for forgetfulness of the channel can not be met and the channel is not forgetful which contradicts our assumption. Thus all all images of finitely localizes observables are finitely localized independent of n for large n.

⇐: Now let every image of every finitely localized observable be finitely localized independent of n for large n. Then for some finite n the image of all these observables on the memory has to be the identity. Thus the channel is strictly forgetful.

□

Theorem A.2. *Let $T : \mathfrak{B} \otimes \mathfrak{M} \to \mathfrak{M} \otimes \mathfrak{A}$ be a reversible strictly forgetful channel. Then its inverse $T^{-1} : \mathfrak{M} \otimes \mathfrak{A} \to \mathfrak{B} \otimes \mathfrak{M}$ is also strictly forgetful.*

Proof. Strictly forgetful channels map finitely localized observables to finitely and independent of the number of concatenations n for large n localized observables. For large n the image of a finitely localized observable commutes with every observable which is localized in the last m subsystems:

$$[\underbrace{T_n[\mathbb{1}_{\mathfrak{B}}^{\otimes n-m} \otimes B_m \otimes M]}_{=\mathbb{1}_{\mathfrak{M}} \otimes \mathbb{1}_{\mathfrak{A}}^{\otimes n-l} \otimes A_l \in \mathfrak{M} \otimes \mathfrak{A}^{\otimes n}}, M' \otimes A_m \otimes \mathbb{1}_{\mathfrak{A}}^{\otimes n-m}] = 0, \quad \text{for large } n. \tag{A1}$$

B_m is localized on the last m subsystems. Equation (A1) has to hold for all A_m, B_m, M and M'. (It might look like B_m depends on n. This is only the reason, because we count from the point of view of the input system. We could also fix B_0 at subsystem 0 and shift A_m to negative indices on the input system.) If we apply th inverse channel to the commutator, which has the same effect as applying it on the individual observables because the channel is reversible and thus a homomorphism, we get

$$[\mathbb{1}_{\mathfrak{B}}^{\otimes n-m} \otimes B_m \otimes M, T_n^{-1}[M' \otimes A_m \otimes \mathbb{1}_{\mathfrak{A}}^{\otimes n-m}]] = 0, \quad \text{for large } n. \tag{A2}$$

This is the same condition we had for T_n. The only difference is the exchanged role of input and output system. Therefore the inverse channel is also strictly forgetful. □

[1] J. Gütschow, "Clifford channels with memory," (2009), internal report for the CORNER EU-project.

[2] H. F. Chau, Physical Review A **58**, 905 (August 1998), arXiv:quant-ph/9802009 [quant-ph], `http://arxiv.org/abs/quant-ph/9802009`.

[3] M. Grassl and M. Rötteler, in *Proceedings 2007 IEEE International Symposium on Information Theory* (Nice, France, 2007) pp. 816–820, arXiv:quant-ph/0703182v1 [quant-ph], `http://arxiv.org/abs/quant-ph/0703182`.

[4] M. M. Wilde and T. A. Brun, Preprint, Submitted to IEEE Transactions on Information Theory(2007), arXiv:0712.2223v3-[quant-ph], http://arxiv.org/abs/0712.2223.
[5] M. M. Wilde and T. A. Brun, Preprint, Submitted to the 2008 IEEE International Symposium on Information Theory (ISIT 2008)(2008), arXiv:0801.0821v1-[quant-ph], http://arxiv.org/abs/0801.0821.
[6] D. Poulin, J.-P. Tillich, and H. Ollivier, Preprint(December 2007), arXiv:0712.2888-[quant-ph], http://arxiv.org/abs/0712.2888.
[7] J. G. David Forney, M. Grassl, and S. Guha, IEEE Transactions on Information Theory **53**, 865 (March 2007), arXiv:quant-ph/0511016v2 -[quant-ph], http://arxiv.org/abs/quant-ph/0511016.
[8] H. Ollivier and J.-P. Tillich, Physical Review Letters **91** (October 2003), arXiv:quant-ph/0304189v2 -[quant-ph], http://arxiv.org/abs/quant-ph/0304189.
[9] H. Ollivier and J.-P. Tillich, preprint(Jan 2004), arXiv:quant-ph/0401134v1 -[quant-ph], http://arxiv.org/abs/quant-ph/0401134.
[10] D. M. Schlingemann, H. Vogts, and R. F. Werner, Journal of Mathematical Physics **49** (2008), 0804.4447v1-[quant-ph].
[11] D. Gottesman, *Stabilizer Codes and Quantum Error Correction*, Ph.D. thesis, California Institute of Technology, Pasadena, California (May 1997), arXiv:quant-ph/9705052v1 -[quant-ph], http://arxiv.org/abs/quant-ph/9705052.
[12] M. Grassl and M. Rötteler, in "emph"bibinfo-booktitle-Proceedings -2006-IEEE-International-Symposium-on-Information-Theory (Seattle, USA, 2006) pp. 1109–1113, arXiv:quant-ph/0602129v1 -[quant-ph], http://arxiv.org/abs/quant-ph/0602129.