SIAM J. COMPUT.
Vol. 38, No. 4, pp. 1207–1282

FAULT-TOLERANT QUANTUM COMPUTATION WITH CONSTANT ERROR RATE*

DORIT AHARONOV† AND MICHAEL BEN-OR†

Abstract. This paper shows that quantum computation can be made fault-tolerant against errors and inaccuracies when η, the probability for an error in a qubit or a gate, is smaller than a constant threshold η_c. This result improves on Shor's result [*Proceedings of the* 37*th Symposium on the Foundations of Computer Science*, IEEE, Los Alamitos, CA, 1996, pp. 56–65], which shows how to perform fault-tolerant quantum computation when the error rate η decays polylogarithmically with the size of the computation, an assumption which is physically unreasonable. The cost of making the quantum circuit fault-tolerant in our construction is polylogarithmic in time and space. Our result holds for a very general local noise model, which includes probabilistic errors, decoherence, amplitude damping, depolarization, and systematic inaccuracies in the gates. Moreover, we allow exponentially decaying correlations between the errors both in space and in time. Fault-tolerant computation can be performed with any universal set of gates. The result also holds for quantum particles with $p > 2$ states, namely, p-qudits, and is also generalized to one-dimensional quantum computers with only nearest-neighbor interactions. No measurements, or classical operations, are required during the quantum computation. We estimate the threshold of our construction to be $\eta_c \simeq 10^{-6}$, in the best case. By this we show that local noise is in principle not an obstacle for scalable quantum computation. The main ingredient of our proof is the computation on states encoded by a quantum error correcting code (QECC). To this end we introduce a special class of Calderbank–Shor–Steane (CSS) codes, called polynomial codes (the quantum analogue of Reed–Solomon codes). Their nice algebraic structure allows all of the encoded gates to be transversal. We also provide another version of the proof which uses more general CSS codes, but its encoded gates are slightly less elegant. To achieve fault tolerance, we encode the quantum circuit by another circuit by using one of these QECCs. This step is repeated polyloglog many times, each step slightly improving the effective error rate, to achieve the desired reliability. The resulting circuit exhibits a hierarchical structure, and for the analysis of its robustness we borrow terminology from Khalfin and Tsirelson [*Found. Phys.*, 22 (1992), pp. 879–948] and Gács [*Advances in Computing Research: A Research Annual: Randomness and Computation*, JAI Press, Greenwich, CT, 1989]. The paper is to a large extent self-contained. In particular, we provide simpler proofs for many of the known results we use, such as the fact that it suffices to correct for bit-flips and phase-flips, the correctness of CSS codes, and the fact that two-qubit gates are universal, together with their extensions to higher-dimensional particles. We also provide full proofs of the universality of the sets of gates we use (the proof of universality was missing in Shor's paper). This paper thus provides a self-contained and complete proof of universal fault-tolerant quantum computation in the presence of local noise.

Key words. quantum computation, quantum fault tolerance, noise and decoherence, quantum error correction, concatenated codes, density matrices, polynomial codes, quantum Reed–Solomon codes, universal set of gates

AMS subject classifications. 81P68, 68Q01

DOI. 10.1137/S0097539799359385

*Received by the editors July 21, 1999; accepted for publication (in revised form) October 28, 2007; published electronically July 23, 2008. A preliminary version of this paper, under the name "Fault-Tolerant Quantum Computation with Constant Error," was published in *Proceedings of the* 29*th Annual ACM Symposium on Theory of Computing (STOC)*, El Paso, TX, 1997, pp. 176–188. An extended version quite similar to the current version is posted online at http://arxiv.org/abs/quant-ph/9906129. This research was supported by The Israel Science Foundation, grant 69/96, and the Minerva Leibniz Center at the Hebrew University in Jerusalem.

http://www.siam.org/journals/sicomp/38-4/35938.html

†School of Computer Science and Engineering, The Hebrew University, Jerusalem, Israel (doria@cs.huji.ac.il, benor@cs.huji.ac.il). The second author is the incumbent of the Jean and Helena Alfassa Chair in Computer Science.

1. Introduction. The area of quantum algorithms has witnessed remarkable discoveries over the past decade, the most remarkable of which is Shor's groundbreaking discovery of a polynomial quantum algorithm for factoring [82]. These results indicate a possibility that quantum computers exhibit exponential speedups over classical computers. It is yet unclear whether and how large scale quantum computers will be physically realizable (see [68] for possible implementation schemes). However, as in any physical system, quantum computers will *in principle* be subjected to noise, such as decoherence [103, 93, 94], and inaccuracies. Without error corrections, the effect of noise will accumulate and ruin the entire computation [97, 30, 71, 65, 66, 20]. Thus, the question of whether quantum computation can be protected against noise cannot be separated from the subject of quantum computational complexity.

One cannot hope to achieve fault tolerance of any computational model, be it quantum or classical, unless the noise model is restricted in some way. In this paper we adopt the assumption that the noise is *local*. Roughly, it means that the noise is independent between the different components of the system. This seems to be a fairly natural assumption to work with from a physical point of view and is often assumed in the classical case. We characterize the amount of noise by a parameter η which we call the *error rate*. The most basic variant of local noise with error rate η is called *independent probabilistic noise* and is defined as the following process: Each gate undergoes an arbitrary error with independent probability η, and in addition, for every time step, the qubits that do not participate in any gate undergo errors, too, each with independent probability η. Local noise can be defined much more generally, to include decoherence, systematic errors, and other kinds of physical processes. We will deal with these later in this paper, but for a first reading it is best to keep in mind the simplest local noise model of independent probabilistic noise.

This paper settles the question of whether quantum computation preserves its computational power in the presence of local noise. We give a positive answer to this question, for a very wide variety of cases under the title of local noise. To do this, we show how to simulate an unreliable quantum circuit by one that is robust to local noise of error rate η, as long as η is below a certain (constant) threshold. The overhead is only polylogarithmic in both space and time. By this we show that, in principle, local noise is not an obstacle against the physical realization of quantum computers. We do not attempt here to optimize the threshold value but rather to achieve a systematic and simple mathematical structure and as general a proof as possible. In the rest of the introduction, we state and explain the result in more detail, describe its context and limitations, and provide an outline of the proof.

1.1. Background on quantum error correction and fault tolerance. The analogue question of protecting classical computation from errors was already studied by von Neumann in 1956, when he showed that classical computation can be made robust to noise with constant error probability per gate [67]. This was done by using computation on redundant information, encoded by a repetition code.

The similar question for quantum computation, however, is far more complicated. Even the simpler question of transmitting quantum information reliably over a noisy channel poses a lot of conceptual obstacles in comparison with its classical analogue, and, in fact, scientists were skeptical about this possibility [97, 62]. There were several valid reasons for these concerns. First, the fact that quantum states cannot be cloned [99] implies that straightforward redundancy techniques cannot help. Second, Hilbert space is continuous, and this implies that it may be difficult to distinguish between bona fide quantum information and a state with a small error. Third, and

possibly the most difficult problem, is the fact that quantum noise can cause occasional *measurements* of the quantum state by the environment which lead to a collapse of the quantum state—a process which seems to cause irreversible damage to the quantum information. Finally, it seems that, in order to correct the error, one must measure the state to figure out what error had occurred. It seems impossible to perform such a measurement without collapsing the state and losing the quantum information.

These pessimistic beliefs were disputed and replaced with cautious optimism with the discoveries of the first examples of quantum error correcting codes (QECCs), encoding one qubit by nine [83] or seven [88] qubits and allowing for the correction of one faulty qubit. The main ingredients that allowed this discovery are as follows: (1) Despite the seemingly infinitely versatile set of possible quantum errors, all quantum errors on one qubit can actually be written as combinations of two errors (called *bit-flip* and *phase-flip*). It is thus sufficient to correct for these two errors. This discretization of the errors allowed turning to classical techniques of error correction. (2) In order to discover the error that had occurred, it suffices to perform a partial measurement on the state, which does not necessarily collapse the part of the state carrying the important information.

Immediately after this discovery, Calderbank and Shor [28], and independently Steane [89], presented a general construction of a large class of good quantum error correcting codes,[1] now called Calderbank–Shor–Steane (CSS) codes, which are based on known classical error correcting codes. These results were followed by the development of a whole theory for quantum error correction, including the important definition of stabilizer codes, and with many examples of QECCs (see, e.g., [68, 87] for more information and references). The theory of QECCs shows that, in principle, quantum information can be transmitted over a noisy channel in a reliable and efficient way.

The existence of QECCs is not in itself sufficient to ensure the possibility of quantum computation in the presence of noise. The reason is as follows. To compute in the presence of noise, one might try to encode the quantum state by using a QECC. In order to prevent the accumulation of errors, one must perform error corrections frequently, in between the computational steps. However, when dealing with noisy computation we are faced with several new problems that do not exist when dealing with the problem of transmitting information over noisy channels. First, in the information transmission setting, one assumes that the encoding and decoding processes are error-free. In the case of noisy computation, on the other hand, the error correction procedure itself is done in the presence of noise. It can thus introduce additional errors into the state and quite possibly cause more harm than help. A second problem is that the process of computation involves interactions between different qubits. If a gate is applied on two qubits, where one of them is faulty, it is likely that at the end of the operation the two qubits are faulty, or, in other words, the gate has caused the *propagation* of the error from a faulty qubit to an error-free qubit. One must therefore be able to compute on encoded states by using procedures and error corrections which do not allow the errors to propagate too much. Such procedures which limit the propagation of errors are called *fault-tolerant.*

In [84], Shor showed how to design fault-tolerant procedures for a universal set of quantum gates.[2] These procedures are applied to states encoded by a code from

[1] Good codes are codes which have a nonzero asymptotic transmission rate, in the presence of a constant error rate per bit.

[2] A set of gates is universal if an arbitrary unitary transformation can be constructed by using only gates from the set.

a certain class of CSS codes. In order to use these fault-tolerant constructions so as to improve the reliability of the quantum circuit, one would first encode the qubits in the original circuit by using the CSS quantum error correcting code and then apply fault-tolerant computation and fault-tolerant error corrections on these encoded states alternately. Shor showed that the resulting quantum circuit performs the desired computation reliably when the error rate, or the fault probability at each time step, per qubit or gate, decays polylogarithmically with the size of the quantum circuit. This result is a major improvement over the performances of quantum circuits without error corrections, in which the error probability is required to decay as one over the size of the circuit in order for the computation to succeed [21].

Nevertheless, the assumption that Shor used, namely, that the error probability decays polylogarithmically with the size of the computer, is a physically unrealistic assumption. We would like to assume that the resources required for applying one elementary operation are fixed and independent of the input size and, in particular, that the probability for a fault in each computer element, as well as the inaccuracy in each gate, is a constant number independent on the number of computer components.

1.2. Results. In this paper we improve on Shor's result and show how to perform fault-tolerant quantum computation in a more realistic model of computation, in which the error rate η is a fixed parameter, independent of the size of the computation.

THEOREM 1 (the threshold result, roughly). *There exists a threshold $\eta_0 > 0$ such that the following holds. Let $\epsilon > 0$. If Q is a quantum circuit operating on n input qubits for t time steps using s two- and one-qubit gates, there exists a quantum circuit Q' with depth, size, and width overheads which are polylogarithmic in n, s, t, and $1/\epsilon$ such that, in the presence of local noise of error rate $\eta < \eta_0$, Q' computes a function which is within ϵ total variation distance from the function computed by Q.*

The locality assumption is at the core of our threshold result. Our methods cannot handle errors which hit large sets of qubits chosen by an adversary. Within this realm of local noise, our fault tolerance result holds for a very general noise model. Our first threshold theorem (Theorem 12) is devoted to proving the threshold result for the independent probabilistic noise model. We use it as a basis to prove several other versions of the threshold theorem. Section 8 extends the result to work with what we call general local noise (Theorem 13). This version allows us to include in the noise model also decoherence, amplitude and phase damping, depolarization, systematic inaccuracies in the gates, leakage errors (of some sort—not disappearance of particles), and more. In fact, our result seems to apply to almost any conceivable quantum process as long as it is local. Later on, we generalize Theorem 12 in another direction (section 10) and show that our construction is reliable even when the probabilistic noise model allows correlations in space and time, as long as these correlations decay exponentially in some well defined sense.

We further generalize the threshold result by considering not only generalizations of the noise model but also different constraints to which the quantum system may be subjected. In particular, Theorem 14 shows that fault tolerance against general local noise can be achieved with any universal set of gates (not necessarily with the set of gates that we show how to encode fault tolerantly). This may be important in the likely scenario in which the set of gates implementable in the laboratory is different than the one we use in our constructions. Furthermore, in various implementation schemes only nearest-neighbor gates are available. Theorem 15 shows that fault tolerance can be achieved even when the particles of the quantum computer are

set on a one-, two-, or three-dimensional grid, and only nearest-neighbor interactions are allowed.

The various versions of the threshold theorem lead to different values of the threshold.

1.3. Assumptions on the architecture of the quantum system. It might be helpful to make explicit various issues related to the requirements from the physical realization of the quantum computer in order for our fault tolerance scheme to work.

- *Redundancy in the input.* We always assume that the input is given to the quantum circuit in many copies. This redundancy requirement is unavoidable if we want to achieve robustness to noise, because otherwise we would have negligible probability that we even start the computation correctly. This is also assumed in the classical scenario [67].
- *Redundancy in the output.* The output of the circuit is also given in a redundant way. In order to interpret the output string, one needs to calculate a certain majority function of the many output bits, which are all equal in the ideal case but not in the presence of errors.
- *Noise on wires vs. noise on gates.* In some quantum systems, the noise and decoherence on the wires are negligible. In other words, errors occur only while the qubits participate in a gate. In this paper we do not make such an assumption, and we assume that the wires are noisy, too. If one is dealing with a quantum system with noisy wires, then one must bear in mind that, in order to achieve fault tolerance, the quantum system must satisfy two important requirements.
  1. *Parallelism.* It is required that gates on different qubits can be applied in parallel, so that many gates can be applied at the same time step. It can be shown that, without parallelism, fault tolerance is impossible when wires are noisy, since error corrections cannot be applied fast enough to prevent accumulation of errors [5].
  2. *A supply of fresh qubits.* Another requirement is that fresh or clean qubits namely, qubits in the state $|0\rangle$, are available at any time and not just in the beginning. In other words, it must be possible to *restart* a qubit, namely, initialize it to the state $|0\rangle$, in the middle of the computation. Once again, without this assumption, reliable quantum computation is impossible because there is no way to discard entropy which accumulates rapidly in the circuit due to noise [7].
- *Intermediate measurements.* One might imagine cases in which measurements can be applied during the quantum computation, and so parts of the intermediate computation (e.g., computations inside the error correction procedure) can be performed by a classical computer. Moreover, in some cases it is also reasonable to assume that classical computations can be performed infinitely fast. This would be the case if the time it takes to perform a classical operation is negligible in comparison to the time it takes to perform a quantum gate. Of course, such cases are significantly simpler to handle than the general case. The reason is that, to a very large accuracy, classical computation can be assumed to be error-free, and so intermediate computations, such as parts of the error correction, can then be assumed to be error-free. In this paper we do not require this assumption. Measurements in the middle of the computation are not necessary in order to achieve fault tolerance. Though allowing intermediate measurements simplifies the situation considerably, we

make an effort to show that fault tolerance can be achieved without using classical operations or measurements. We do this because of two reasons. One is purely theoretical: One would like to know that measurements and classical operations are not essential and that the quantum model is complete in the sense that it can be made fault-tolerant within itself. Even more importantly, the assumption on the infinitely fast classical computation breaks down at some point and does not scale with the size of the input. One should certainly try to avoid such an assumption when attempting to prove a threshold theorem that holds asymptotically. The other reason we make an effort to avoid intermediate measurements is practical: In some suggestions for physical realizations of quantum computers (such as the NMR-based quantum computation—see [68]), it is very difficult to perform intermediate measurements during the quantum computation, and therefore everything should be done within the quantum computation model, without the help of classical computers.

- *Different sets of gates.* Our main fault-tolerant construction (Theorem 12) uses very specific universal sets of gates (see subsections 4.2 and 5.2 for the definitions). Nevertheless, one can actually convert the fault-tolerant circuit into one that uses any universal set of gates. This is proved and explained in section 9. Again, this is important for practical purposes, because the set of gates that is implementable in the laboratory might be entirely different than the ones that are easy to implement fault-tolerantly.
- *Nearest-neighbor interactions.* In some implementation schemes, interaction is limited to nearest-neighbor particles. We show that this is not a restriction in terms of fault tolerance: Resilience to noise can be achieved even under very restricted geometrical conditions, namely, that qubits are arranged in a one-dimensional lattice and gates are allowed only between nearest neighbors.

We summarize the division of our threshold theorems with respect to their assumptions on the architecture of the system. All of our theorems allow noise on the wires and thus assume parallelism and a constant supply of fresh qubits. Also, all of our theorems work without the assumption of intermediate measurements. Our first two versions of the threshold result, Theorems 12 and 13, and also section 10, are concerned with quantum systems subject to the following two assumptions on their architecture: First, a gate can be applied on any set of particles, regardless of their physical location. In other words, qubits that participate in a gate need not be nearest neighbors. Second, the final realization of the quantum circuit may consist of gates taken out of two particular sets, which we call $\mathcal{G}_1$ or $\mathcal{G}_2$. The other versions of our theorem release these assumptions. Theorem 15 shows that nearest-neighbor interactions suffice, and Theorem 14 allows the use of arbitrary universal sets of gates, including sets that contain only two-qubit gates.

1.4. Proof overview. We outline here the proof of our first threshold result, Theorem 12, which holds for the simplest noise model of independent probabilistic errors. The extensions to other cases build upon this case and do not require much more innovation.

1.4.1. Ingredients for the construction. There are several ingredients that are required in the proof.

1. A quantum error correcting code. To this end we define a new type of a quantum error correcting code (QECC) called *polynomial quantum error correcting code*. This is the quantum analogue of the Reed–Solomon code [64],

and it is a special case of CSS codes over large fields. We draw intuition here from Ben-Or, Goldwasser, and Wigderson [24], who used Reed–Solomon codes to achieve classical fault-tolerant distributed computation. An alternative version of the proof uses the general class of CSS code that Shor used in [84].

2. To compute on states encoded by QECCs, we need fault-tolerant maintenance procedures. First, we require a fault-tolerant way to perform error correction, which would allow us to correct errors in encoded states without introducing too many new errors. Second, we need a zero-state preparation procedure, namely, a procedure which prepares a state that encodes $|0\rangle$ (and uses fresh qubits for this purpose). These encoded zero states will be used as ancilla states for error correction as well as for computation. Both of these procedures can be performed in many different ways (e.g., [4]). Here we choose to work with the nice idea used by Steane for error correction [89]. Finally, we need a fault-tolerant decoding procedure which would allow us to read the logical value of the encoded state. This is not difficult to design.
3. To perform the actual computation, we need a universal set of gates which can be performed fault-tolerantly on encoded states. Each gate g needs to be replaced by an *encoded gate*, which takes the encoding of a state $|\alpha\rangle$ to the encoding of $g|\alpha\rangle$. Most importantly, the encoded gates need to be applied fault-tolerantly, i.e., without letting the errors propagate too much. Here is where the reason we chose to work with polynomial codes becomes apparent. Due to their algebraic structure, these codes exhibit the following very nice property: The entire universal set of quantum gates can be performed in a manner which is called *transversal*, or *bitwise*. This term refers to the application of the encoded gate g simply by applying g on each of the qubits in the code word individually (see Figure 1.1).
   This means that the set of encoded gates is extremely simple. Unfortunately, some complication cannot be avoided. The transversal operations in the case of polynomial codes indeed perform the desired computation, but in some cases they cause the degree of the polynomials involved in the code to increase (to see why this might happen, consider the case in which two polynomials are multiplied). To avoid accumulation of this effect, we perform a fairly simple procedure called *degree reduction*. This procedure, too, can be applied transversally, except it requires the help of ancilla zero states. Overall, we get a set of fault-tolerant procedures which possesses a nice algebraic structure. We also provide fault-tolerant procedures for the alternative codes we use, namely, the more general CSS codes that are used in Shor's scheme. Our constructions are based on those used by Shor [84], adapted to our re-

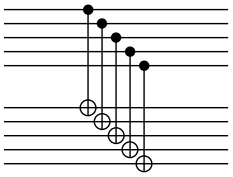


FIG. 1.1. *Encoded* CNOT *gate on two logical qubits encoded by five qubits each. Five* CNOT *gates are applied, where the ith gate is applied from the ith qubit in the first block to the ith qubit in the second block.*

quirement of no measurements during the computation. The fault-tolerant application of the Toffoli gate in this case does not possess as nice an algebraic structure as the procedures we design for polynomial codes.

4. We next need to show that the set of gates which we use is indeed universal, so that arbitrary quantum computations can be performed using our scheme. A known result due to Solovay [85] and Kitaev [50, 52] implies that it suffices for this purpose to show that the set of gates in question generates a dense subset in the group of unitary matrices. We use algebraic and field theoretical tools, and employ a few known group theoretical results, to prove that both sets of gates we use indeed satisfy this density requirement. In the case of polynomial codes, the proof is fairly involved. We note that a proof of the universality of the set of gates used by Shor was missing in [84].

1.4.2. The fault-tolerant circuit. We now want to combine all of the above ingredients to construct a fault-tolerant circuit from a given circuit M_0 which is not protected against errors. We want to make the circuit reliable against a constant error rate η, independent of the size of the computer n.

The first step of the idea is already present in Shor's original fault-tolerant result (except for using different codes and fault-tolerant procedures): Compute on states encoded by a QECC. We use a QECC that encodes one qubit into, say, m qubits. The circuit M_1 which computes on encoded states is defined as a *simulation* of M_0. A qubit in M_0 transforms to a block of m qubits in M_1, and each gate in M_0 transforms to a fault-tolerant procedure in M_1 applied on the corresponding blocks. Note that procedures might take more than one time step to perform. Thus, a time step in M_0 is mapped to a time interval in M_1, which is called a *working period*. In order to prevent accumulation of errors, at the beginning of each working period an error correction is applied on each block. The working period now consists of two stages: a correction stage and a computation stage. To be precise, the initialization and output stages in M_1 require some special attention, in order to encode the redundant input, and also decode the output states into a redundant classical output. We do not elaborate on this part here.

The idea is therefore to apply alternately computation stages and correction stages, hoping that during the computation stage the damage that accumulated is still small enough so that the corrections are able to handle it. We will soon argue that if the error rate is below a certain threshold, the reliability of M_1 is better than that of M_0. Let us assume this for the moment. In any case, we need to reduce the effective error rate of the final circuit to something which is inverse polynomially small, which would guarantee that with high probability there is effectively no error in the entire computation. Some thought would lead the reader to the conclusion that, regardless of how we choose the QECC in the above scheme, it seems impossible to achieve an improvement from a constant error rate to an effective error rate that is inverse polynomial.

Our solution is fairly simple to state. The crucial point here is that the new circuit M_1 might not be reliable enough for our purposes, but it suffices that it is even *slightly* more reliable than the original circuit. If this is the case, we can continue to improve the reliability by repeating the same process as we did for the original circuit, but starting from the new, more reliable circuit M_1. We do this for several levels; let us denote the number of levels by r, so the final circuit is M_r. How many levels are needed? It turns out that the improvement in the error rate is doubly exponential in the number of levels, and so it suffices that r is taken to be polyloglog in the

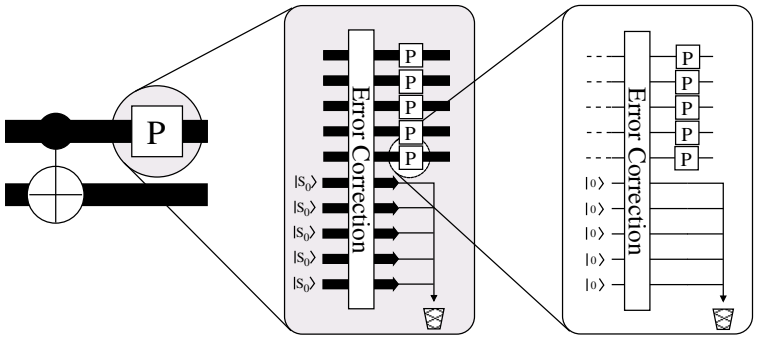


FIG. 1.2. *This figure shows the two-level simulation of a two-gate circuit. When looking at it at the coarsest resolution, we see the circuit on the left. The gates we see are those gates which* M_0 *consists of, namely, a* CNOT *followed by a one-qubit gate. By increasing the resolution a bit, we see the next level circuit* M_1*. The figure shows only part of it, namely, the second gate in* M_0 *transformed in* M_1 *to its encoded version. This includes an error correction of that block (with the help of an ancilla), followed by a transversal application of the one-qubit gate. By increasing the resolution yet one last time, we get to see the actual physical circuit, namely,* M_2*. Here we show only the encoding of one one-qubit gate in* M_1*.*

parameters of M_0.

The resulting circuit M_r exhibits a hierarchical structure resembling the one in Figure 1.2.

Each qubit in the original circuit transforms to a block of qubits in the next level, and they in their turn transform to a block of blocks in the second simulation, and so on. A gate in the original circuit transforms to a procedure in the next level, which transforms to a larger procedure containing smaller procedures in the next level, and so on. The final circuit computes in all of the levels: The largest procedures, computing on the largest (highest-level) blocks, correspond to operations on qubits in the original circuit. The smaller procedures, operating on smaller blocks, correspond to computation in lower levels. Note that each level simulates the error corrections in the previous level and adds error corrections in the current level. The final circuit, thus, includes error corrections of all of the levels, where during the computation of error corrections of larger blocks, smaller blocks of lower levels are being corrected. The lower the level, the more often error corrections of this level are applied, which is in correspondence with the fact that smaller blocks are more likely to be quickly damaged. This is the source of the advantage of using recursive simulations, rather than one step of simulation, as is done in Shor's scheme [84].

One other way to understand this is that the hierarchical structure takes advantage of a very important fact, namely, that the errors are located *randomly*. It is easy to see that the resulting QECC, namely, a concatenation of the QECC for r levels, can correct very few (less than a fraction of n) errors if their locations are chosen by an adversary, but as long as their locations are random, all is fine.

We note that the final circuit in one of our constructions (the one that uses polynomial codes) is made of p-qudits rather than qubits. This circuit can be easily converted to one that works with qubits by replacing each qudit by the appropriate number of qubits and each gate by a sequence of gates on these qubits. The analysis of robustness of this final circuit follows exactly the lines of section 9.

1.4.3. New reliability and the threshold value. We now want to argue that the reliability of the simulating circuit M_1 is larger than that of the original circuit

M_0, as long as the error rate is below a certain threshold. In other words, we want to explain why, below a certain threshold error rate, we should expect some improvement in reliability from one level of simulation to the next.

In the original circuit, the occurrence of one fault may cause the failure of the computation. In contrast, suppose that one fault occurs in each procedure of the simulating circuit. If the fault-tolerant procedures allow this fault to propagate to, say, at most one error in each block, then these faults would have no effect on the correctness of the simulation since they would be corrected during the error correcting stages. In other words, if in each encoded gate and the preceding error correction procedure the number of faults is such that they can cause only less than the correctable number of errors, then the error corrections will prevent the errors from accumulating. (In fact, this argument needs to be made much more carefully (see Lemma 9), but for the sake of the introduction here it suffices.) The *effective* error rate of M_1 is thus, roughly, the probability for more than the correctable number of faults to occur in one procedure.

Just for the sake of illustration, let us consider an example in which our QECC corrects one error but not two and a fault-tolerant procedure allows one fault to propagate to at most one error in each of the final blocks. We now estimate the effective error rate in a slightly incorrect way, but which demonstrates the idea. In this case, the effective error rate is the probability for more than one fault to occur in one procedure. In other words, we need to consider all ways in which two or more faults can occur in one procedure. This is a combinatorial calculation, which behaves approximately like

$$\eta_{\text{eff}} \approx \begin{pmatrix} A \\ 2 \end{pmatrix} \eta^2,$$

where A is the number of locations an error can possibly occur during the application of one procedure. Now observe that, for η_{eff} to be strictly smaller than η, we get the requirement $\eta < \frac{1}{\binom{A}{2}}$. Thus, if η is below a certain threshold that depends on A, we gain an improvement in the reliability of the circuit. This is the origin for the *threshold* in the threshold theorem. See section 12 for estimations of the values in our construction.

1.4.4. The proof. The analysis of the scheme turns out to be quite complicated. To do this, we borrow terminology used by Khalfin and Tsirelson [49] and Gács [42] to analyze self-correcting classical cellular automata. First, we distinguish between two notions which we have been using so far as if they are the same: the *error* in the state of the qubits, i.e., the set of qubits which have errors, and the actual *faults*, namely, the occasions where noise operators were applied. We use the term *fault path* to denote the list of locations, namely, points in time and space, where faults had occurred (in a specific run of the computation).

We then define the notion of sparse errors and sparse fault paths. Naturally, due to the hierarchical structure of this scheme, these notions are defined recursively. A block in the lowest level is said to be k-close to its correct state if it does not have more than k errors, and a higher level block is "k-close" to its correct state if it does not contain more than k blocks of the previous level which are far from their correct state. If all of the blocks of the highest level are k-close to being correct, we say that the set of errors, or the deviation, is *k-sparse*.

Which set of faults does not cause the state to be too far from correct in the above metric? The answer is recursive, too: A computation of the lowest-level procedure

is said to be undamaged if not too many faults occurred in it. Computations of higher-level procedures are not damaged if they do not contain too many lower-level procedures which are damaged. A fault path will be called *sparse* if the computations of all of the highest-level procedures are not damaged.

The proof of the threshold result is done by showing that the probability for "bad" faults, i.e., the probability for the set of faults not to be sparse, decays as a double exponential with the number of levels r. Thus, bad faults are negligible, if the error rate is below the threshold. This is the easy part. The more complicated task is to show that the "good" faults are indeed good; i.e., if the set of faults is sparse enough, then the set of errors is also kept sparse until the end of the computation. This part is done by using a double induction on the number of levels r.

1.5. Related work and the state of the art in quantum fault tolerance. A preliminary version of the threshold result (for independent probabilistic errors) was published in the proceedings version of this paper [3]. Different variants of the threshold result for probabilistic noise were independently discovered at about the same time (1996) by Knill, Laflamme, and Zurek [56, 57, 58], who used Steane's seven-qubit QECC, and by Kitaev [50], who used toric codes. All of these works are based on the same idea of applying one scheme recursively to get a hierarchical structure and achieve approximately the same estimated threshold value of 10^{-6}.

An important unique contribution of our work is that our proof does not require intermediate measurements and classical operations during the quantum computation: Our fault-tolerant circuits use only quantum gates and, thus, work entirely within the framework of the quantum model. All works on fault tolerance we are aware of (other than the current paper) work under the assumption that intermediate measurements are allowed, and, moreover, classical computation can be performed infinitely fast. This makes the problem significantly easier, because under these assumptions large parts of the computation can be assumed to be error-free.

Our proof, as well as the proof of Kitaev [50], requires using a QECC that corrects two errors. It turns out that our construction works for a QECC that corrects one error, but a different analysis is required. An idea of how to perform this analysis was suggested in [56, 57, 58], by using the notion of overlapping rectangles. This idea was recently made into a proof by Aliferis, Gottesman, and Preskill [14]. A different proof was independently given by Reichardt [78].

Since the discovery of the threshold result mentioned above, intensive scientific effort was put into generalizing, simplifying, and improving the fault-tolerant networks. This effort started with the important work by Gottesman [45, 44] showing that fault tolerance can be achieved with any stabilizer code and with the works of Preskill [73, 74] and of Steane [90, 91]; it continued with a long line of works which we will not attempt to survey here for lack of space. The main goal of this line of work is practical: that of improving the threshold and making it realistic.

We mention a few main lines of developments. The first is that of developing the usage of ancilla state distillation, also known as *ancilla factories*. In the basic fault-tolerant schemes, one initiates the computation with states encoding 0, and the preparation of these states can be thought of as a separate process. Steane [90], Gottesman and Chuang [48], and finally Bravyi and Kitaev [27] have each shown how to use more and more cleverly designed ancillary states to significantly simplify the main computational process and improve the threshold.

The creation of ancilla states of high fidelity is the bottleneck in the threshold calculations in the above-mentioned schemes. In a breakthrough work, Knill [54] has

suggested to use concatenation of quantum error *detection* (rather than correction) codes, for the distillation of ancilla states. If an error is detected, the ancilla state can be discarded, without affecting the ongoing computation. The overhead in the number of qubits is large, but numerical estimations [54] of the threshold of this scheme revealed a value of up to several percent. Reichardt [76] and independently Aliferis, Gottesman, and Preskill [15] provide a rigorous proof of a much worse threshold than the estimated one, but still the best rigorously proven today: 10^{-3}. It is left open to close the gap between the numerics and the theoretical work.

Completely different methods to achieve fault tolerance were also explored. The first idea in this context is the beautiful idea of Kitaev, of achieving fault tolerance in the different model of topological quantum computation with anyons [51]. In this model of computation, the information is protected by topological means, and no concatenation is required.

Another interesting development along nonstandard lines is that of using subsystem codes [59]. These are a generalization of quantum error correcting codes. Aliferis and Cross [13] recently showed how to use subsystem codes due to Bacon [16], to get a threshold of $1.9 \cdot 10^{-4}$.

So far, we have discussed only methods which did not take into consideration geometrical restrictions on the system. Another issue which is explored in this paper is that of fault-tolerant quantum computation with nearest-neighbor interactions between qubits in one-, two, or three-dimensional systems. We prove a threshold result under such assumptions in section 11. This was proved independently also by Gottesman in [47]. In contrast to [47], our proof of generalization to low dimensions is very simple (see section 11), since one only has to keep track of the error propagation in one level, and the general threshold theorem (section 7) takes care of the rest. Both results provide very small values of the threshold. The lower bound on the threshold value in the case of $1D$ was improved recently to 10^{-6} in [92]. In two dimensions, better results are known. Svore, Divincenzo, and Terhal [95] have proved a threshold in $2D$ which is of the order of 10^{-5}, and Raussendorf and Harrington [75] have provided a scheme in which topological codes are used, in which the simulated threshold is of the order of 10^{-3}.

All of the above works consider the case of probabilistic noise only. Yet another different line of work in the area of fault tolerance is the study of the limitations and applicability of the threshold result in the presence of more general noise models. This direction is explored quite thoroughly in the current paper, where we prove the threshold result for a general local noise model (section 8) and, moreover, (slightly) relax the locality requirement to allow exponentially decaying correlations in the noise process (section 10). Following the preliminary version of this work [4], and based on similar methods, Terhal and Burkard [96], and recently Aliferis, Gottesman, and Preskill [14], proved threshold results in the presence of non-Markovian noise. The most recent development in this context [9] shows that fault tolerance can be achieved even in the presence of correlations which decay *polynomially* in space.

We also mention here an alternative approach to fault tolerance which has attracted interest, namely, "decoherence-free subspaces" (DFSs) [41, 102, 63, 101, 55]. A DFS is a subspace of the Hilbert space, which, under certain assumptions on the noise, is completely unaffected by the noise process. The computation is then performed entirely within this subspace [17, 18]. The DFS method might indeed be very useful in the presence of special types of noise, such as collective decoherence (see references above), and can thus deal with various correlated noise processes. However, we note that the assumptions on the noise made in the DFS model are quite

restrictive and, in particular, do not allow local noise. If one wants to correct for local noise, these methods must be combined with the standard methods of fault tolerance presented in this paper.

Finally, since the publication of the preliminary version of this paper [3], polynomial codes have found several applications in other areas of quantum computation such as secure multiparty computation [36] and quantum secret sharing [33]. Nonbinary codes were defined independently also by Chuang, Leung, and Yamamoto [31] and Knill [53]. Fault tolerance with higher-dimensional particles was discussed also in [46].

1.6. Conclusions and open problems. This paper implies that quantum computation can in theory be carried out in the presence of noise, as long as the noise satisfies a fairly reasonable assumption, namely, locality, and as long as the error rate in the system can be decreased below a certain constant value. Within this locality assumption, our results holds for a very general noise model. Moreover, our result holds for quantum systems with fairly limited constraints, such as no classical computation and subject to geometrical restrictions. Thus, this paper provides a thorough and general solution to the problem of quantum computation in the presence of local noise.

The value of the threshold is of utmost importance to the practicality of the theorem. We did not attempt to optimize the threshold value in our fault tolerance scheme, and our estimated threshold of $\approx 10^{-6}$ (see section 12) is far from being practical, according to the state of the art in experimental systems. As mentioned in subsection 1.5, since the preliminary publication of this work there have been many improvements. The best current rigorous results give a threshold value of approximately 10^{-3}, whereas computer simulations of the best constructions say that the threshold value is of the order of one to several percent. Providing rigorous proofs for the latter is a challenge. Any improvement in the threshold value is significant, as it brings quantum computation closer to being practical. We note that one should be careful in interpreting these numbers, since the threshold depends on many parameters; see section 12 and subsection 1.3 for further discussion of this matter.

Another parameter which is crucial for practical purposes is the space efficiency of the fault-tolerant construction, namely, the overhead in the number of qubits. Once again, we have not attempted to optimize this parameter. This parameter has achieved significantly less attention in the literature than the threshold value. See [77, 54] for some discussion of this matter.

It is of course desirable to save time as well. Our scheme requires a polylogarithmic blowup in the depth of the circuit. In the original version of this paper [3, 4] we conjectured that it might be possible to use a quantum analogue of multilinear codes [86], to reduce the multiplicative factor of $O(\log(n))$ to a factor of $O(\log(\log(n))$. Recently, Ahn [10] indeed achieved fault tolerance with this improved depth overhead, by using clever combinatorial considerations for toric codes. An open question is whether it is possible to reduce the time cost to a constant, as in the classical case. We conjecture that the answer is negative.

Another important open problem is to further relax the assumptions on the noise model and to make them as realistic as possible. This is a natural continuation of the work presented in sections 8 and 10 of the current paper and in several papers of the past few years [96, 14, 9].

The results achieved in this work shed light on a very interesting question, regarding the transition from quantum to classical physics [103]. Traditionally, this transition is viewed as a gradual transition (but see [49]). The threshold result suggests that in some cases this transition can actually be viewed as a *phase transition*,

occurring at a critical error rate, above which the system abruptly loses its ability to entangle large sets of particles. This is supported by the result of this paper, showing that at a very small error rate quantum systems can maintain their quantum nature, together with another result of ours [5], showing that at very high error rates quantum computers can be simulated efficiently by a classical Turing machine. In [2] one of us formalized this phase transition idea and provided a physical explanation for this computational difference between high and low error rates, by showing that in fault-tolerant circuits a phase transition in a certain property of entanglement called *entanglement length* occurs at a critical error rate. This draws a natural connection between the transition from quantum to classical and the notion of entanglement. The connection between the quantum-classical transition, entanglement, and quantum phase transitions has since become an active field of research (see, e.g., [69, 70]). We believe that the physical realization of fault-tolerant quantum computers might enable far better understanding of this fundamental question.

1.7. Organization of paper and instructions as to how to read it. Section 2 starts with a rigorous definition of the model of quantum circuits which we use, namely, quantum circuits with mixed states. We define the basic noise model, of independent probabilistic errors, and its various generalizations. Section 3 provides the definitions of quantum error correcting codes in general and the particular QECC codes we use in this paper. Section 4 describes how to apply fault-tolerant procedures for polynomial codes. Section 5 describes how to apply fault-tolerant gates on states encoded by CSS QECC. Section 6 proves the universality of the sets of gates defined in the previous two sections. Section 7 proves the threshold result for independent probabilistic noise. This is the main result of the paper. Section 8 generalizes the threshold result to general local noise. In section 9 we generalize our result to circuits which use any universal set of gates and not necessarily the set of gates for which we have constructed fault-tolerant procedures. In section 10 we relax the assumption of independence and explain how to deal with exponentially decaying correlations in the noise. Section 11 proves the threshold result for quantum circuits with restricted geometry, i.e., one-dimensional quantum circuits. In section 12 we discuss the estimation of the threshold values in various cases.

Many readers would be satisfied with understanding the main ingredients of the proof of the threshold result, for the simplest noise model of independent probabilistic noise, and for one type of QECC, either general CSS codes or polynomial codes. Such readers might also be less interested in the details of the proof that the set of gates we use is universal. The relevant chapters for those readers are sections 2.8, 2.9, 3, 4 (or 5), and 7. In general, the paper is written in a very modular way, so the readers can skip to the interesting parts.

Each section starts with a short overview. A reader who is interested in a more detailed overview than the one provided in the introduction, but still not in all of the details, can read only these overviews.

2. The model of noisy quantum circuits.

2.1. Overview. This section provides the necessary definitions for this paper. For more details on the quantum computation model consult [68]; for more on quantum mechanics consult [34, 79, 72].

The standard model [38, 39, 100] of quantum circuits with unitary gates allows only unitary operations on qubits. However, noisy quantum systems are not isolated from other quantum systems, usually referred to as the environment. Their inter-

actions with the environment are unitary, but, when restricting the system to the quantum computer alone, the operation on the system is no longer unitary, and the state of the quantum circuit is no longer a vector in the Hilbert space. It is indeed possible to stay within the unitary model with pure states, by keeping track of the state of the environment. However, this environment is not part of the computer, and it is assumed that we have no control or information on its state. We find it more elegant to follow the framework of the physicists and to work within the model of quantum circuits with mixed states defined by Aharonov, Kitaev, and Nisan [6]. In this model, the state of the set of qubits is always defined: It is a probability distribution over pure states, i.e., a *mixed state*. It is described by a *density matrix* (see below). The quantum gates in this model are not necessarily unitary: Any physically allowed operator on qubits is a quantum gate. In particular, this model allows an operation which adds a blank qubit to the system, discards a qubit, or applies a measurement in some basis. The model of quantum circuits with mixed states is equivalent in computational power to the standard model of quantum circuits [6] but seems more adequate for the study of noisy quantum computation. Since noise is a dynamical process which depends on time, the circuits we consider are leveled; i.e., gates are applied at discrete time steps.

To include noise in our model, we add noise operators which act in between the time steps of the circuit. The simplest model for noise is the independent probabilistic noise. After every time step, we consider the sets of qubits that interacted in the previous time step. Each such set undergoes a fault (i.e., an arbitrary quantum operation) with independent probability η, and η is referred to as the *error rate*. A qubit that did not participate in a gate undergoes the same process on its own.

The probabilistic noise process can be generalized to a more realistic model of noise, which is the following process. After each time step, each set of qubits participating in the same gate goes through a physical operator which is at most η-far from the identity (in some operator metric discussed below). This model includes, apart from probabilistic errors, also decoherence, amplitude and phase damping, systematic inaccuracies in the gates, and more (see [68] for an explanation of these terms). Two important assumptions were made in this definition: independence between different faults in space, i.e., locality, and independence in time, which is called the Markovian assumption. We remain within this framework until the end of the paper, and only in section 10 do we relax it to allow also exponentially decaying correlations in both time and space.

We note that all definitions here use qubits, namely, two-state particles, but can trivially be extended to any finite-dimensional particles, namely, to *qudits*.

2.2. Pure states. A quantum physical system in a *pure state* is described by a unit vector in a Hilbert space, i.e., a vector space with an inner product. In the *Dirac* notation a pure state is denoted by $|\alpha\rangle$. The physical system which corresponds to a quantum circuit consists of n quantum two-state particles, called qubits. The Hilbert space of a qubit is a two-dimensional complex space and is denoted by $\mathcal{H}_2$. We choose an orthonormal basis for this space and denote its vectors by $|0\rangle$ and $|1\rangle$. The Hilbert space of n qubits is the n-fold tensor product of $\mathcal{H}_2$, namely, $\mathcal{H}_2 \otimes \cdots \otimes \mathcal{H}_2$, its dimension is 2^n, and it is isomorphic to $\mathcal{C}^{2^n}$. As a basis for this space we choose all possible tensor products of the basis vectors of the individual qubits $|i_1\rangle \otimes |i_2\rangle \otimes \cdots \otimes |i_n\rangle$, where i_j is either 0 or 1. We often use the notation $|i_1, i_2, \ldots, i_n\rangle$ or $|i\rangle$ for brevity. A pure state $|\alpha\rangle \in \mathcal{H}_2^n$ is a *superposition* of the basis states: $|\alpha\rangle = \sum_{i=1}^{2^n} c_i|i\rangle$, with $\sum_{i=1}^{2^n} |c_i|^2 = 1$. The transposed-complex conjugate of $|\alpha\rangle$ is denoted by $\langle\alpha|$. The

inner product between $|\alpha\rangle$ and $|\beta\rangle$ is denoted by $\langle\alpha|\beta\rangle$.

2.3. Mixed states and density matrices. In general, a quantum system is not in a pure state but in a *mixed state*, namely, a probability distribution, or a *mixture* of pure states, denoted by $\{\alpha\} = \{p_k, |\alpha_k\rangle\}$. This means that the system is with probability p_k in the pure state $|\alpha_k\rangle$. This description is not unique, as different mixtures might represent the same physical system. An equivalent unique description uses *density matrices*. A density matrix ρ on $\mathcal{H}_2^n$ is an Hermitian (i.e., $\rho = \rho^\dagger$) semipositive-definite matrix, of dimension $2^n \times 2^n$, with trace $\mathrm{Tr}(\rho) = 1$. A pure state $|\alpha\rangle = \sum_i c_i|i\rangle$ is represented by the density matrix $\rho_{|\alpha\rangle} = |\alpha\rangle\langle\alpha|$ (by definition, $\rho(i,j) = \langle i|\rho|j\rangle$). A mixture $\{\alpha\} = \{p_s, |\alpha_s\rangle\}$ is associated with the density matrix $\rho_{\{\alpha\}} = \sum_s p_s|\alpha_s\rangle\langle\alpha_s|$. This association is not one-to-one, but it is *onto* the density matrices, because any density matrix describes the mixture of its eigenvectors, with the probabilities being the corresponding eigenvalues.

Given a density matrix of n qubits, one can assign a density matrix to a subsystem A of $m < n$ qubits. We say that the rest of the system, represented by the Hilbert space $\mathcal{G} = \mathbf{C}^{2^{n-m}}$, is *traced out* and denote the new matrix by $\rho|_A$. We have $\rho|_A(i,j) = \sum_{k=1}^{2^{n-m}} \rho(ik, jk)$, and k runs over a basis for $\mathcal{G}$. In words, this means averaging over $\mathcal{G}$. The new density matrix $\rho|_A$ is called the *reduced* density matrix to A. If the state of the qubits which are traced out is in tensor product with the state of the other qubits, then discarding the qubits means simply erasing their state. However, if the state of the discarded qubits is not in tensor product with the rest, the reduced density matrix is always a mixed state.

2.4. Measurements. A quantum system can be *measured*, or observed. The measurement is a probabilistic process which, given a density matrix, yields a pair: a classical output and the associated density matrix. In this paper we will use only a measurement of one qubit in the computational basis. Let P_0 (P_1) be a projection on the subspace spanned by all states in which the measured qubit is 0 (1). For a given density matrix ρ, the classical output is $m \in \{0,1\}$ with probability $\Pr(m) = \mathrm{Tr}(P_m\rho)$. Given the outcome m, the resulting state of the quantum system after the measurement is equal to $\Pr(m)^{-1}P_m\rho P_m$ (this has the same meaning as a conditional probability distribution). Any measurement thus defines a probability distribution over the possible outputs.

2.5. Quantum operators and quantum gates. To define the possible transformations on density matrices, we consider linear operators on operators (sometimes called *superoperators*). A superoperator is called *positive* if it sends positive-semidefinite Hermitian matrices to positive-semidefinite Hermitian matrices. Superoperators can be extended to operate on larger spaces by taking a tensor product with the identity operator. A superoperator is a *completely positive map* if all of its extensions are positive. For a superoperator to be physically permissible, it must take density matrices to density matrices. Thus, it must be a completely positive map which is trace-preserving. It turns out that any superoperator which satisfies these conditions is indeed physically permissible, as it takes density matrices to density matrices [60, 61]. Thus quantum superoperators are defined as follows.

DEFINITION 1. *A* permissible superoperator T *from k to ℓ qubits is a trace-preserving, completely positive, linear map from density matrices on k qubits to density matrices on ℓ qubits. Its action on a density matrix ρ is denoted as follows: $\rho \longmapsto T \circ \rho$.*

Linear operations on mixed states preserve the probabilistic interpretation of the

mixture: $T \circ \rho = T \circ (\sum_s p_s |\alpha_s\rangle\langle\alpha_s|) = \sum_s p_s T \circ (|\alpha_s\rangle\langle\alpha_s|)$.

A very important example of a quantum superoperator is the superoperator corresponding to the standard unitary transformation $|\alpha\rangle \mapsto U|\alpha\rangle$, which sends a quantum state $\rho = |\alpha\rangle\langle\alpha|$ to the state $U\rho U^\dagger$.

Another important superoperator is to discard a set of qubits. This operation corresponds to tracing out the discarded qubits. In this paper, a discarding qubit gate will always be applied to qubits which ideally (if no error occurred) are in tensor product with the rest of the system.

One more useful quantum superoperator is the one which adds a blank qubit to the system $V_0 : |\xi\rangle \mapsto |\xi\rangle \otimes |0\rangle : \mathbf{C}^{2^n} \to \mathbf{C}^{2^{n+1}}$. This is described by the superoperator $T : \rho \mapsto \rho \otimes |0\rangle\langle 0|$.

A lemma by Choi [29] and Hellwig and Kraus [60, 61] asserts that any permissible superoperator from k to l qubits can be described as a combination of the above three operations: Add $k + l$ blank qubits, apply a unitary transformation on the $2k + l$ qubits, and then trace out $2k$ qubits. One can think of this as the definition of a permissible superoperator. We define a quantum gate to be the most general quantum operator.

DEFINITION 2. *A quantum gate from k to ℓ qubits is a permissible superoperator from k to ℓ qubits.*

In this paper we will use only three types of quantum gates: discard or add qubits and, of course, unitary gates. However, quantum noise will be allowed to be an arbitrary permissible superoperator (with the required restrictions as explained below).

We mention here that it is possible to describe a measurement as a superoperator. For this, we replace the classical result of a measurement $m \in \{0, 1\}$ by the density matrix $|m\rangle\langle m|$ in an appropriate Hilbert space $\mathcal{M}$. We then present the measurement as a superoperator T:

$$T\rho \;=\; \sum_m (P_m \rho P_m) \otimes \Big(|m\rangle\langle m|\Big). \tag{2.1}$$

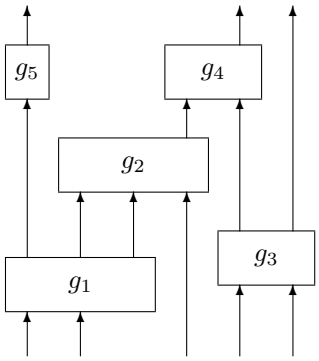
2.6. The trace metric on density operators. A useful metric on density matrices is the *trace* metric. It is induced from the *trace norm* for general Hermitian operators. The trace norm on Hermitian matrices is half the sum of the absolute values of the eigenvalues. $\|\rho\| = \frac{1}{2}\sum_i |\lambda_i|$. We thus have that $0 \le \|\rho_1 - \rho_2\| \le 1$. It was shown in [6] that the trace distance between two matrices equals the measurable distance between them in the following sense. Consider the two probability distributions D_1, D_2 which one gets when applying the same measurement to the two density matrices. Each such measurement thus induces some statistical difference between the outcomes, quantified by the total variation distance between the two distributions.[3] The trace distance is exactly the maximum over all possible measurements of half the total variation distance.

2.7. Quantum circuits with mixed states. We now define a quantum circuit.

DEFINITION 3. *Let $\mathcal{G}$ be a family of quantum gates.* A quantum circuit that uses gates from $\mathcal{G}$ *is a directed acyclic graph. Each node v in the graph is labeled by a gate $g_v \in \mathcal{G}$ from k_v to ℓ_v qubits. The in-degree and out-degree of v are equal k_v and ℓ_v, respectively.*

[3] The total variation distance between two probability distributions D_1, D_2 is defined to be $\sum_j |D_1(j) - D_2(j)|$.

Here is a schematic example of such a circuit.



The circuit operates on a density matrix as follows.

DEFINITION 4 (final density matrix). *Let Q be a quantum circuit. Choose a topological sort for Q: $g_t, \ldots, g_1$, where g_j are the gates used in the circuit. The final density matrix for an initial density matrix ρ is $Q \circ \rho = g_t \circ \cdots \circ g_2 \circ g_1 \circ \rho$.*

It is easy to see that $Q \circ \rho$ is well defined and does not depend on the topological sort of the circuit (see [6] for a detailed proof). At the end of the computation, the r output qubits are measured, and the classical outcome, which is a string of r bits, is the output of the circuit. For an input string i, the probability for an output string j is $\langle j|(Q \circ |i\rangle\langle i|)|j\rangle$.

2.8. Quantum circuits with independent probabilistic noise. To introduce noise to the quantum circuits, we consider leveled quantum circuits, where each level corresponds to one time step. Let us denote by T the number of time steps in the quantum circuit. The gates are then applied at integer times from 1 to T. We recall that, due to the input gate, qubits can be input to the circuit at different time steps. Faults occur in between time steps. We define a *location* (namely, a possible place where a fault can occur) as follows.

DEFINITION 5. *A set $(b_1, b_2, \ldots, b_\ell, t)$ is a* location *in the quantum circuit Q if the qubits $b_1, \ldots, b_\ell$ participate in the same gate in Q at time step t, and no other qubit participates in that gate. For this matter, the absence of a gate is viewed as the identity gate on one qubit, so if a qubit b does not participate in any gate at time t, then (b, t) is a location in Q as well.*

We will need the following terminology for our analysis of fault tolerance.

DEFINITION 6 (fault path). *The list of locations where faults had occurred (in a specific run of the computation) is called a fault path.*

DEFINITION 7 (error rate). *In the independent probabilistic noise model, each location is faulty with independent probability η, and this probability is called the error rate of the circuit.*

Each fault path $\mathcal{F}$ is thus assigned a probability $\Pr(\mathcal{F})$, which is a function of the number of locations in $\mathcal{F}$, the total number of locations in the circuit, and η.

The fault path does not determine the exact faults that occur in the circuit. In our model, once the fault path is chosen, an adversary chooses the exact noise superoperators applied on the faulty locations. That is, given a fault path, the adversary chooses T permissible superoperators; each one is applied on the faulty locations of time t in the fault path. We denote this choice of superoperators by $\mathcal{E}(\mathcal{F}) = (\mathcal{E}_T, \ldots, \mathcal{E}_2, \mathcal{E}_1)$, where $\mathcal{E}_t$ is a permissible superoperator applied by the noise process on the faulty locations of time step t.

We can now define the model precisely.

DEFINITION 8 (the independent probabilistic noise model). *Let Q be a leveled quantum circuit of T time steps. Let $\mathcal{E}(\mathcal{F})$ be a function from fault paths on Q to T-tuples of superoperators (as above). The noisy quantum circuit is denoted by $Q^{\mathcal{E}}$ and is defined as follows. We first define the final density matrix for a given choice of a fault path to be the result of applying the noise superoperators in between the time steps: $Q^{\mathcal{E}}(\mathcal{F}) \circ \rho = \mathcal{E}_t \circ g_t \circ \cdots \circ \mathcal{E}_2 \circ g_2 \circ \mathcal{E}_1 \circ g_1 \circ \rho$. The final density matrix of the circuit $Q^{\mathcal{E}} \circ \rho$ is defined as a weighted average over the final density matrices for each fault path:*

$$Q^{\mathcal{E}} \circ \rho = \sum_{\mathcal{F}} Pr(\mathcal{F}) Q^{\mathcal{E}}(\mathcal{F}) \circ \rho. \tag{2.2}$$

We say that a noisy quantum computer computes a function f with error ϵ in the presence of error rate η if $Q^{\mathcal{E}}$ defined above computes the function f with an error probability at most ϵ for any adversarial choice of faults $\mathcal{E}$.

This model includes, for example, probabilistic measurements of individual qubits, as well as the depolarization model in which each qubit is randomly replaced by a completely random qubit, i.e., a qubit in the identity density matrix.

2.9. Quantum circuits with general local noise. Most of the paper uses only the independent probabilistic noise model defined above. However, in section 8 we extend the threshold result to hold for a much more general noise model. In the general noise model, we replace the probabilistic faults which occur with probability η in each location with a different noise operator which is applied at each and every location in the quantum circuit, after every time step. The only restriction on the noise operator applied at a specific location is that it is within η distance to the identity. Thus, in between time steps, a *noise operator* of the following form operates on *all* of the qubits (here we give an example for the noise operator after time step t):

$$\mathcal{E}(t) = \mathcal{E}_{A_{1,t}}(t) \otimes \mathcal{E}_{A_{2,t}}(t) \otimes \cdots \otimes \mathcal{E}_{A_{l_t,t}}(t). \tag{2.3}$$

$A_{i,t}$ runs over all possible locations at time t, and for each one of them,

$$\|\mathcal{E}_{A_{i,t}}(t) - I\| \leq \eta. \tag{2.4}$$

The norm we use here is the *diamond norm* on superoperators [50, 6]. The exact definition of the diamond norm is not needed in this paper. We merely use the following properties (see [6]):

1. $\|T\rho\| \leq \|T\|\|\rho\|$.
2. $\|TR\| \leq \|T\|\|R\|$.
3. $\|T \otimes R\| = \|T\|\|R\|$.
4. The norm of any permissible superoperator T is equal to 1.

This defines the general local noise model with error rate η.

An interesting example of such a noise process is a restricted version of the independent probabilistic noise model. Suppose for each one of the superoperators in the product (2.3) we write

$$\mathcal{E}_{A_{i,t}}(t) = \left(1 - \frac{\eta}{2}\right) I_{A_{i,t}} + \frac{\eta}{2} \mathcal{E}'_{A_{i,t}}(t), \tag{2.5}$$

where $\mathcal{E}'_{A_{i,t}}(t)$ is some permissible superoperator. This operator is within η distance from the identity, and so this satisfies the conditions of local noise (2.3), (2.4) with error rate η. It is easy to see that this gives a restricted version of the independent noise model with error rate $\eta/2$, because the operator $\mathcal{E}_{A_{i,t}}$ can be interpreted as if the operator $\mathcal{E}'_{A_{i,t}}(t)$ is applied with probability $\eta/2$, and otherwise nothing is done to the qubit. Note that we do not get the independent probabilistic model in its full generality because (2.3) implies that the noise operator on the faulty locations of each time step is a tensor product of operators, one on each location.

This gives a very general error model, under the locality restriction. The model includes many physical sources for noise and errors, such as decoherence, systematic inaccuracies in the gates, amplitude and phase damping, and more (for explanations of these terms, see [43, 68]). As for leakage errors, this is a term that can mean one of two things. The first is that the state of the particle, due to noise, leaves the subspace spanned by computational states. This can be easily taken care of by defining the computational space to include all possible states of the particle, and so this is included implicitly in our discussion (allowing, as we do, higher-dimensional particles). The other meaning of leakage errors is the disappearance of a particle. We do not deal with this source of errors in this paper.

The general local noise model can be further relaxed to allow exponentially decaying correlations in time and space, but we delay the exact definition to section 10 where we deal with such correlations.

3. Quantum error correcting codes.

3.1. Overview of QECC, CSS codes, and polynomial codes. We will use here only QECCs which encode one qubit into, say, m qubits, and we call the set of m qubits a *block*. By linearity, if $|0\rangle$ is encoded by $|S_0\rangle$ and $|1\rangle$ by $|S_1\rangle$, then $a|0\rangle + b|1\rangle$ is encoded by $a|S_0\rangle + b|S_0\rangle$. It follows that such a quantum code is a two-dimensional subspace in the Hilbert space of m qubits. We note that we slightly abuse language here, confusing between the code space and the map encoding the states into this subspace. The exact meaning should be clear from the context. The state of the encoded qubit is sometimes called the *logical* state. The code is said to correct q errors if the logical state is recoverable given that not more than q errors occurred in the block. Roughly speaking, we say that no more than q errors occurred if the density matrix of the remaining $m - q$ qubits did not change.

There is a big difference between classical ECCs and QECCs, in that, for QECCs, not only basis states but also quantum superpositions should be recoverable after an error occurred. The codes we use here are CSS codes [28], and their construction is based on two ideas that enable the protection of superpositions.

1. Despite the fact that quantum operations are extremely versatile and can be continuous, the most general fault or quantum operation on a qubit can be described as (loosely speaking) a linear combination of four simple operations: the identity, i.e., no error at all, a bit-flip ($|0\rangle \leftrightarrow |1\rangle$), a phase-flip ($|0\rangle \longmapsto |0\rangle, |1\rangle \longmapsto -|1\rangle$), or both a bit-flip and a phase-flip. Thus correcting for these three errors suffice. This important fact was first proved by Bennett et al. [22], by using a beautiful group symmetry argument and a notion called *twirl*. We give here a simple proof based on an idea by Steane [89].
2. In order to correct bit-flips, one can use the analogue of classical error correcting codes. To correct phase-flips, one observes that phase-flips are actually bit-flips in the Fourier basis. One can therefore correct bit-flips in the Fourier transform basis, and this will translate to correcting phase-flips in the correct

basis. To correct general errors, one would first correct bit-flips and then correct bit-flips in the Fourier transformed basis, which translates to correcting phase-flips in the original basis.

CSS codes [28, 89] were first presented for qubits, that is, over the field F_2. We give here a proof for the fact that CSS codes are quantum codes, which is simpler than the original proof [28], and generalize it to the field F_p for any prime[4] $p > 2$.

Next, we define a new class of quantum codes, which are called *polynomial codes*. The idea underlying their construction is based on a theorem by Schumacher and Nielsen [80] which asserts that a quantum state can be corrected only if no information about the state has leaked to the environment through the noise process. More precisely, if the reduced density matrix on any t qubits does not depend on the logical qubit, then there exists a unitary operation which recovers the original state even if the environment interacted with t qubits, i.e., t errors occurred. This is reminiscent of the situation in classical secret-sharing schemes [81] and hints at the following possibility: We should divide the "secret," i.e., the logical qudit, among many parties (i.e., physical qudits) such that no t parties share any information about the secret. We adopt Shamir's secret-sharing scheme [81] which suggested to use random polynomials, evaluated at different points in a field of p elements, as a way to divide a secret among m parties. A random polynomial of degree d is chosen, and then each party gets the evaluation of the polynomial at a different point in the field F_p. The secret is the value of the polynomial at 0. To adopt this scheme to the quantum setting, we simply replace the random polynomial by a superposition of all polynomials of the appropriate degree to get a QECC. The result is the quantum analogue of Reed–Solomon codes [64]. It turns out that polynomial codes are a special case of CSS codes over a large field F_p.

The remainder of the section assumes basic knowledge in classical error correcting codes, which can be found in [64].

3.2. Quantum codes. A quantum code is a subspace of some dimension, say, k, in the Hilbert space of $m > k$ qubits. A word in the code is any vector in the subspace or, more generally, any density matrix ρ supported on the subspace. We say that a density matrix ρ' on m qubits is a word with q errors in the qubits $b_1, \ldots, b_q$, if ρ' can be written as $\mathcal{E}(b_1, \ldots, b_q) \circ \rho$, where ρ a word in the code, and $\mathcal{E}(b_1, \ldots, b_q)$ is some permissible superoperator on the q qubits. A quantum circuit $\mathcal{R}$ is said to correct an error $\mathcal{E}(b_1, \ldots, b_q)$ on ρ if

$$\mathcal{R} \circ \mathcal{E}(b_1, \ldots, b_q) \circ \rho = \rho. \tag{3.1}$$

We can now define a quantum error correcting code which corrects q errors.

DEFINITION 9 (QECC). *An $[[m, k, q]]$ quantum error correcting code is a subspace of dimension 2^k in the Hilbert space of m qubits such that the following holds. Let $\{|\alpha_i\rangle\}_{i=1}^k$ be a basis for C. There exists a quantum circuit $\mathcal{R}$, called an error correction procedure, such that for any i, j and $\mathcal{E}(b_1, \ldots, b_q)$*

$$\mathcal{R} \circ \mathcal{E}(b_1, \ldots b_q) \circ |\alpha_i\rangle\langle\alpha_j| = |\alpha_i\rangle\langle\alpha_j|.$$

Note that, from linearity, the last requirement implies the correction of any word in the code and not just the basis states. Note that $\mathcal{R}$ may (and, in fact, has to) use extra qubits during its application, but at the end all ancillary qubits are discarded, and the original state should be recovered.

[4] In fact, the results in this paper can be extended to work over any field, but we do not provide the details here.

3.3. From general errors to bit-flips and phase-flips. We now prove that it suffices to correct bit-flips and phase-flips in order to correct for general errors. To do this, we use the linearity of quantum operators. We recall the definition of the Pauli operators (slightly modified for simplicity here):

$$\mathcal{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},\ \sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},\ \sigma_y = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix},\ \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{3.2}$$

σ_x is called a bit-flip, since it takes $|0\rangle$ to $|1\rangle$ and $|1\rangle$ to $|0\rangle$. σ_z is called a phase-flip, as it multiplies $|1\rangle$ by a minus sign, while leaving $|0\rangle$ as is. σ_y is a combination of the two, since $\sigma_y = \sigma_z\sigma_x$. Let us define a *Pauli error* on a qubit as the application of one of the nontrivial Pauli matrices on that qubit. We note that the Pauli operators form a basis for the matrices on one qubit.

We proceed to many qubits. Consider all possible strings e of length m in the alphabet $\mathcal{I}, X, Y, Z$. The set of 4^m matrices:

$$\sigma_e = \sigma_{e_m} \otimes \cdots \sigma_{e_1}, \quad e \in \{\mathcal{I}, X, Y, Z\}^m, \tag{3.3}$$

form a basis for the linear operators on the Hilbert space of m qubits. We call it the Pauli basis. It is easy to check that this is an orthonormal basis with respect to the inner product $(V, U) = \frac{1}{2^m}\mathrm{tr}(VU^\dagger)$ for $2^m \times 2^m$ matrices.

DEFINITION 10. *We say that σ_e, for $e \in \{\mathcal{I}, X, Y, Z\}^m$, is a Pauli error of length q, if the string e has at most q coordinates that are not the identity.*

Before we show that it suffices to correct Pauli errors, we need two definitions.

DEFINITION 11. *We say that $\mathcal{R}$ corrects Pauli errors of length q in C if for any two basis states in the code, $|\alpha\rangle$ and $|\alpha'\rangle$, and any Pauli error of length q, σ_e, we have*

$$\mathcal{R} \circ \sigma_e|\alpha\rangle\langle\alpha'|\sigma_e^\dagger = |\alpha\rangle\langle\alpha'|. \tag{3.4}$$

DEFINITION 12. *We say that $\mathcal{R}$ detects Pauli errors of length q in C if for any two basis states of the code, $|\alpha\rangle$ and $|\alpha'\rangle$, and any two different Pauli errors of length q, $\sigma_e \neq \sigma_{e'}$,*

$$\mathcal{R} \circ \sigma_e|\alpha\rangle\langle\alpha'|\sigma_{e'}^\dagger = 0. \tag{3.5}$$

The reason for the fact that such $\mathcal{R}$ is said to detect errors is the following. Suppose $\mathcal{R}$ first writes down e on an ancilla $\mathcal{R} \circ \sigma_e|\alpha\rangle = |\alpha\rangle \otimes |e\rangle$. Then for two different errors the two ancilla states are orthogonal, and thus when discarding the ancilla qubits we get zero. If $\mathcal{R}$ does not write e on an ancillary register, but instead e can be discovered from the extra information written on the ancilla qubits, then discarding the ancilla register would result similarly in 0 for two different errors. In this paper we use error correcting procedures which not only correct but also detect the errors. We can now prove the following.

THEOREM 2. *Let C be a quantum code, and let $\mathcal{R}$ correct and detect any Pauli error of length q in C. Then R corrects any general error on any q qubits in C.*

Proof. We recall (section 2) that any permissible noise operator $\mathcal{E}$ on q qubits can be written as a combination of three operators: adding $2q$ blank qubits, applying a unitary transformation on the $3q$ qubits, and then discarding $2q$ qubits. We write the error operator as $\mathcal{E} = \mathcal{T} \circ \mathcal{L}$, where $\mathcal{T}$ is the discarding qubits operator. In the most general case, $\mathcal{L}$ is the following operation on q qubits. For all $0 \leq i \leq 2^q - 1$,

$$\mathcal{L} : |i\rangle \longmapsto |0^{2q}\rangle|i\rangle \longmapsto \sum_{j=0}^{2^q-1} |A_{i,j}\rangle \otimes |j\rangle, \tag{3.6}$$

where all of the coefficients are put into the vectors $|A_i^j\rangle$ which are states of $2q$ qubits (these vectors are not necessarily unit vectors). Observe that $\mathcal{L}$ can be presented as a multiplication of the state of the right register of q qubits by

$$\sum_{i,j} |A_{i,j}\rangle \otimes |j\rangle\langle i|. \tag{3.7}$$

The above object can be viewed as a $2^q \times 2^q$ matrix, whose entries are not real numbers but operators. In particular, each entry is a tensor product with a 2^{2q}-dimensional vector $|A_{i,j}\rangle$.

We would like to present this object in terms of the Pauli basis. This is easy to do by recalling that the Pauli basis is orthonormal. Define A to be the matrix whose entries are the vectors $|A_{i,j}\rangle$. We get that the object in (3.7) is equal to

$$\sum_{e \in \{\mathcal{I},X,Y,Z\}^q} |A_e\rangle \otimes \sigma_e,$$

where $|A_e\rangle = (A, \sigma_e) = \frac{1}{2^q}\mathrm{tr}(A\sigma_e^\dagger)$ ($A\sigma_e^\dagger$ is a matrix of vectors, the trace of which is a vector).

We thus have that

$$\mathcal{L} = \sum_{e,e'} |A_e\rangle\langle A_{e'}| \otimes \sigma_e \cdot \sigma_{e'}^\dagger. \tag{3.8}$$

We can now see that $\mathcal{R}$ indeed corrects for the error $\mathcal{E} = \mathcal{T} \circ \mathcal{L}$, by the following:

$$\begin{aligned} \mathcal{R} \circ \mathcal{E} \circ |\alpha\rangle\langle\alpha'| &= \mathcal{R} \circ \mathcal{T} \circ \sum_{e,e'} |A_e\rangle\langle A_{e'}| \otimes \sigma_e |\alpha\rangle\langle\alpha'| \sigma_{e'}^\dagger \\ &= \left(\mathcal{T} \circ \sum_e |A_e\rangle\langle A_e|\right) \otimes |\alpha\rangle\langle\alpha'| = |\alpha\rangle\langle\alpha'|, \end{aligned} \tag{3.9}$$

where in the second equality we have used the fact that $\mathcal{T}$ commutes with $\mathcal{R}$ because they operate on different qubits and that $\mathcal{R}$ corrects and detects errors. In the last equality we used the fact that $\mathcal{T} \circ \sum_e |A_e\rangle\langle A_e| = c$ does not depend on $|\alpha\rangle, |\alpha'\rangle$. By choosing $|\alpha\rangle = |\alpha'\rangle$, we have that $\mathrm{Tr}(\mathcal{R} \circ \mathcal{E} \circ |\alpha\rangle\langle\alpha|) = \mathrm{Tr}(c|\alpha\rangle\langle\alpha|) = c$, but on the other hand $\mathcal{R} \circ \mathcal{E}$ is trace-preserving, so $c = 1$. ∎

3.4. Calderbank–Shor–Steane codes. We give here the definition of CSS codes, which is a slight modification (and simplification) of the definition from [28] but gives the same codes. A linear code of length m and dimension k is a subspace of dimension k in F_2^m, where F_2^m is the m-dimensional vector space over the field F_2 of two elements. Let C_1, C_2 be two linear codes of length m such that $\{0\} \subset C_2 \subset C_1 \subset F_2^m$, and let us define a quantum code by taking the superpositions of all words in a coset of C_2 in C_1 to be one basis word in the code. We have

$$\forall a \in C_1/C_2 : |S_a\rangle = \frac{1}{\sqrt{2^{\dim(C_2)}}} \sum_{w \in C_2} |w + a\rangle. \tag{3.10}$$

Note that $|S_a\rangle$ is well defined and does not depend on the representative of the coset since if $(a_1 - a_2) \in C_2$, then $|S_{a_1}\rangle = |S_{a_2}\rangle$. Also, for different cosets the vectors are orthogonal. Thus, this defines a basis for a subspace of dimension $2^{\dim(C_1)-\dim(C_2)}$.

This is our quantum code. Note that the support of $|S_a\rangle$ are words in the code C_1. We see that bit-flips can be corrected by using classical error correction techniques for the code C_1 (that is, the classical error correction is performed unitarily, adding ancillary qubits).

Before we discuss how to correct phase-flips, let us define a very important quantum gate on one qubit, called the *Hadamard* gate, or the Fourier transform over F_2:

$$H = H^{-1} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}. \tag{3.11}$$

Observe that

$$H\sigma_z H^{-1} = \sigma_x. \tag{3.12}$$

This means that a phase-flip transforms to a bit-flip in the Fourier transform basis. By applying the Hadamard gate on each qubit in $|S_a\rangle$, we get the state:

$$\begin{aligned} |C_a\rangle = H \otimes H \otimes \cdots \otimes H|S_a\rangle &= \frac{1}{\sqrt{2^{m+\dim(C_2)}}} \sum_{b=0}^{2^m-1} \sum_{w \in C_2} (-1)^{(w+a)\cdot b}|b\rangle \\ &= \frac{1}{\sqrt{2^{m-\dim(C_2)}}} \sum_{b \in C_2^\perp} (-1)^{a\cdot b}|b\rangle, \end{aligned} \tag{3.13}$$

which is a superposition of words in the perpendicular subspace $C_2^\perp$. Thus, to correct phase-flips, one transforms to the Fourier basis and corrects bit-flips in the code $C_2^\perp$. These observations led to the following theorem due to Calderbank and Shor [28]. We give here a simple proof of this theorem, based on Theorem 2.

THEOREM 3 (CSS codes). *Let C_1 and $C_2^\perp$ be linear codes over F_2, of length m, such that $0 \subset C_2 \subset C_1 \subset F_2^m$, and such that $C_2^\perp, C_1$ correct q errors. Then the subspace spanned by $|S_a\rangle$ for all $a \in C_1/C_2$ is a $[[m, 2^{\dim(c_1)-\dim(C_2)}, q]]$ QECC. The error correction procedure $\mathcal{R}$ is constructed by correcting bit-flips with respect to C_1 in the S-basis, rotating to the C-basis by applying Hadamard on each coordinate, correcting with respect to $C_2^\perp$, and rotating back to the S-basis.*

Proof. We define the procedure $\mathcal{R}_{C_1}$ to be a unitary embedding of m qubits into the space of $2m$ qubits, by

$$\mathcal{R}_{C_1}|i\rangle = |w(i)\rangle \otimes |e(i)\rangle \tag{3.14}$$

for each $i \in F_2^m$, where $w(i) \in C_1$ is a string of minimal distance to i, and $e(i) \in \{0,1\}^m$ satisfies $w(i) + e(i) = i$. Since this is a one-to-one transformation, it is a unitary embedding, and therefore it is a permissible superoperator. Let $e_b \in \{0,1\}^m$ have at most q 1′s in it. Let $\mathcal{E}_b$ be an error operator which is the corresponding tensor product of bit-flips (σ_x) and identities. In the coordinates where e_b is 0, $\mathcal{E}_b$ has the identity, and in the coordinates where it is 1, $\mathcal{E}_b$ has a bit-flip. Then for any $|\alpha\rangle, |\alpha'\rangle$ supported on C_1, we have

$$R_{C_1} \circ \mathcal{E}_b \circ |\alpha\rangle\langle\alpha'| = |\alpha\rangle\langle\alpha'| \otimes |e_b\rangle\langle e_b|. \tag{3.15}$$

$\mathcal{R}_{C_2^\perp}$ is defined similarly:

$$\mathcal{R}_{C_2^\perp}|j\rangle = |w(j)\rangle \otimes |e(j)\rangle, \tag{3.16}$$

where $w(j) \in C_2^\perp$ is a string of minimal distance to j, and $e(j) \in \{0,1\}^m$ satisfies $w(j) + e(j) = j$. Let $e_f \in \{0,1\}^m$ have at most q 1′s. Let $\mathcal{E}_f$ be the error operator which is the corresponding tensor product of phase-flips (σ_z) and identities. Then for any $|\beta\rangle, |\beta'\rangle$ supported on $C_2^\perp$, we have

$$R_{C_2^\perp} \circ \mathcal{E}_f \circ |\beta\rangle\langle\beta'| = |\beta\rangle\langle\beta'| \otimes |e_f\rangle\langle e_f|. \tag{3.17}$$

Denote by $\mathcal{H}$ the operator that applies the Hadamard gate H on every qubit: $\mathcal{H} = H \otimes H \otimes \cdots \otimes H$. We claim that the operator

$$\mathcal{R} = \mathcal{T}_f \circ \mathcal{T}_b \circ \mathcal{H} \circ \mathcal{R}_{C_2^\perp} \circ \mathcal{H} \circ \mathcal{R}_{C_1} \tag{3.18}$$

is the desired error correcting procedures. $\mathcal{T}_f, \mathcal{T}_b$ are the operators discarding the qubits added for $\mathcal{R}_{C_2^\perp}, \mathcal{R}_{C_1}$, respectively.

By Theorem 2, it is enough to show that this procedure corrects and detects q Pauli errors. To show that it corrects q Pauli errors, write the error vector e in two parts e_b and e_f as follows: e_b is 1 in the coordinates where σ_x or σ_y occurred and 0 elsewhere. e_f is 1 in the coordinates where σ_z or σ_y occurred and 0 elsewhere. We can therefore write the error operator $\mathcal{E}$ as a product of two operators $\mathcal{E} = \mathcal{E}_f \circ \mathcal{E}_b$, by using the fact that $\sigma_y = \sigma_z \sigma_x$. We now have

$$\mathcal{R} \circ \mathcal{E} \circ |\alpha\rangle\langle\alpha'| = \mathcal{T}_f \circ \mathcal{T}_b \circ \mathcal{H} \circ \mathcal{R}_{C_2^\perp} \circ \mathcal{H} \circ \mathcal{R}_{C_1} \circ \mathcal{E}_f \circ \mathcal{E}_b \circ |\alpha\rangle\langle\alpha'|. \tag{3.19}$$

We would like to start developing the expression above by applying $\mathcal{R}_{C_1}$, but the phase errors stand in the way. For this, note that for any string $|i\rangle$ we have

$$\mathcal{R}_{C_1} \mathcal{E}_f |i\rangle = \mathcal{E}_f R_{C_1} |w(i)\rangle \otimes \mathcal{E}_f |e(i)\rangle.$$

This means that

$$R_{C_1} \mathcal{E}_f = (\mathcal{E}_f \otimes \mathcal{E}_f) R_{C_1}. \tag{3.20}$$

We apply (3.20) as well as (3.15), to the right-hand side of (3.19). We get

$$\mathcal{R} \circ \mathcal{E} \circ |\alpha\rangle\langle\alpha'| = \mathcal{T}_f \circ \mathcal{H} \circ \mathcal{R}_{C_2^\perp} \circ \mathcal{H} \circ \mathcal{E}_f \circ |\alpha\rangle\langle\alpha'| \otimes \mathcal{T}_b \circ \mathcal{E}_f \circ |e_b\rangle\langle e_b|, \tag{3.21}$$

where we have used the fact that $\mathcal{T}_b$ commutes with all operators except R_{C_1}. Since $\mathcal{E}_f$ is trace-preserving, we can now apply $\mathcal{T}_b$ and get a scalar which is exactly 1. We proceed with

$$\begin{aligned} &\mathcal{T}_f \circ \mathcal{H} \circ \mathcal{R}_{C_2^\perp} \circ \mathcal{H} \circ \mathcal{E}_f \circ |\alpha\rangle\langle\alpha'| \\ &= \mathcal{T}_f \circ \mathcal{H} \circ \mathcal{R}_{C_2^\perp} \circ (\mathcal{H} \circ \mathcal{E}_f \circ \mathcal{H}) \circ (\mathcal{H} \circ |\alpha\rangle\langle\alpha'|) \\ &= \mathcal{T}_f \circ \mathcal{H} \circ (\mathcal{H} \circ |\alpha\rangle\langle\alpha'|) \otimes |e_f\rangle\langle e_f| = |\alpha\rangle\langle\alpha'|. \end{aligned} \tag{3.22}$$

We thus see that $\mathcal{R}$ indeed corrects q Pauli errors. It is left to show that $\mathcal{R}$ also detects q Pauli errors. To show that

$$\mathcal{R} \circ \sigma_e |\alpha\rangle\langle\alpha'| \sigma_{e'}^\dagger = 0,$$

we observe that if $e \neq e'$, then either $e_b \neq e_b'$ or $e_f \neq e_f'$. We can repeat the previous argument. If $e_b \neq e_b'$, we get zero due to $\mathcal{T}_b \circ |e_b\rangle\langle e_b'| = 0$. Otherwise, we have $e_f \neq e_f'$, and we get zero because $\mathcal{T}_f \circ |e_f\rangle\langle e_f'| = 0$. ∎

3.5. CSS codes over F_p. The theory of quantum error corrections can be generalized to quantum computers which are composed of quantum particles of $p > 2$ states, called qudits. If we want to stress the dimensionality of the particles, we call them p-qudits. To generalize the notion of bit-flips and phase-flips to p-qudits, define the following two matrices:

- B : $B|a\rangle = |(a+1) \text{mod p}\rangle$,
- P : $P|a\rangle = w^a|a\rangle$,

where $w = e^{\frac{2\pi i}{p}}$. The analogue of Pauli matrices are combinations of powers of these matrices, i.e., the p^2 matrices:

$$B^c P^{c'} \quad \forall c, c' \in F_p. \tag{3.23}$$

Just as we did in the case of Pauli matrices, we can consider the p^{2m} m-fold tensor products of such matrices and show that they form a basis for matrices on m p-qudits. This basis is orthonormal, with respect to the inner product $(U, V) = \frac{1}{p^m}\text{tr}(UB^\dagger)$.

Like in the case of qubits, errors of type B transform to errors of type P and vice versa, via the analogue of the Hadamard, namely, the appropriate Fourier transform. In fact, there are several possible analogues of the Hadamard matrix for the case of F_p. Let $w = e^{2\pi i/p}$ and $w_l = w^l$. Then we define the lth Fourier transform over F_p to be

$$W(w_l) : |a\rangle \longmapsto \frac{1}{\sqrt{p}} \sum_{b \in F} w^{lab}|b\rangle. \tag{3.24}$$

It can be easily checked that

$$\forall c \in F_p, \quad W(w_l)P^c W(w_l)^{-1} = B^{c\ell^{-1}}, \quad W(w_l)B^c W(w_l)^{-1} = P^{c\ell}. \tag{3.25}$$

We can now define CSS codes over F_p in a very similar way as it is done for F_2. We first fix a choice of inner product over F_p^m with which we work: We choose the coefficients c_i in the bilinear form $\vec{a} \cdot \vec{b} = \sum_{i=1}^m c_i a_i b_i$. This fixes what we mean by the orthogonal code $C_2^\perp$. The statements and proofs of Theorems 2 and 3 are generalized to F_p by using the above definition of the generalized bit-flip B and the generalized phase-flip P, where F_2^m is replaced by F_p^m and the Hadamard gate on the ith qubit is replaced by the Fourier transform $W(w_{c_i})$ over the ith qudit.

3.6. Polynomial quantum codes. We define polynomial codes over the field F_p, for p a prime. We set d to be an upper bound on the degree of the polynomials used in the code and m to be the length of the code. Let $m < p$ be the number of elements in the field F_p with which we will be working. Set $\alpha_1, \ldots, \alpha_m$ to be m distinct nonzero elements of the field F_p. We consider the set of polynomials over F_p of degree at most d:

$$V_d = \{f(\cdot) \in F_p[x], \deg(f) \leq d\}, \tag{3.26}$$

where $F_p[x]$ is the field of polynomials with coefficients in F_p. Define the following classical codes:

$$\begin{aligned} C_1 &= \{(f(\alpha_1), \ldots, f(\alpha_m))|f(\cdot) \in V_d\} \subset F_p^m, \\ C_2 &= \{(f(\alpha_1), \ldots, f(\alpha_m))|f(\cdot) \in V_d, f(0) = 0\} \subset C_1. \end{aligned} \tag{3.27}$$

We can now define the quantum code:

$$(3.28)\quad \forall a \in F_p,\quad |S_a\rangle = \frac{1}{\sqrt{p^d}} \sum_{f(\cdot)\in V_d, f(0)=a} |f(\alpha_1), \ldots, f(\alpha_m)\rangle = \frac{1}{\sqrt{p^d}} \sum_{w\in C_2} |w + \vec{a}\rangle,$$

where $\vec{a}$ is the vector of length m with a at each coordinate. Since C_2 has p different cosets in C_1, the dimension of the code is p, and the code encodes exactly one p-qudit. We prove the following theorem.

THEOREM 4. *A polynomial code of degree d with length m over F_p is a* $[[m, p, \min\{\lfloor \frac{m-d-1}{2} \rfloor, \lfloor \frac{d}{2} \rfloor\}]]$ QECC.

Proof. Two different words in C_1 agree on at most d coordinates, and thus C_1 is a linear code of distance $m-d$. It can thus correct and detect $\lfloor (m-d-1)/2 \rfloor$ errors. $C_2^\perp$ (under any choice of the inner product in which the coefficients are nonzero) is a linear code of minimal distance $\geq d+1$. This is true since the projection on any d coordinates of the code C_2 contains all possible strings of length d, and therefore the only vector of length d orthogonal to all of the vectors is the 0 vector. Thus, $C_2^\perp$ corrects and detects $\lfloor d/2 \rfloor$ errors. Theorem 4 follows from Theorem 3 for F_p. □

We call the polynomial code which uses polynomials of degree $m-d-1$ (the codegree of d) the *dual code* of the code which uses polynomials of degree d. By the above lemma, the two codes correct the same number of errors.

4. Fault-tolerant gates for polynomial codes.

4.1. Overview. In this section we define $\mathcal{G}_1$ (subsection 4.2), the universal set of gates with which we work when computing with polynomial codes. We note that our set of gates involves only one- and two-qudit gates. We show how to apply the gates in $\mathcal{G}_1$ on encoded states, in a fault-tolerant manner. Our fault-tolerant procedures will all have what we call *spread* 1. This notion will be defined shortly (subsubsection 4.3.4), but roughly it means that one fault in a procedure can cause at most one error in the final state of the procedure. We augment the fault-tolerant gates by fault-tolerant procedures for error correction, zero-state preparation and decoding (a zero-state preparation procedure prepares a state $|S_0\rangle$, and a decoding procedure takes $|S_a\rangle$ to m copies of a). These procedures also have spread 1. This section thus proves the following theorem.

THEOREM 5. *For any gate in $\mathcal{G}_1$, there exists a fault-tolerant procedure with spread* 1 *that computes the gate on states encoded by a quantum polynomial code. There exist also fault-tolerant error correction, decoding, and zero-state preparation procedures for this code, which all have spread* 1. *Moreover, all of these procedures use only gates from $\mathcal{G}_1$ and in particular do not use measurements.*

The requirement that the procedures use only gates from $\mathcal{G}_1$ is imposed so that the scheme can be applied recursively, as we will do in section 7.

In an ideal situation, all gates can be applied transversally, as in Figure 1.1, in which case it is clear that one fault can propagate to at most one error in each block. Unfortunately, we do not know of any universal set of gates and any code in which all gates can be applied transversally in the simple way depicted by Figure 1.1. It is here that the advantage of polynomial codes comes into play. Due to the algebraic properties of polynomials, all of the gates in $\mathcal{G}_1$ can be applied essentially transversally: The gates are applied either transversally or by first preparing some states of the form $|S_0\rangle$ and then applying some operations transversally on the computational blocks and the ancilla blocks together.

To achieve this nice property, we observe that in the case of polynomial codes the transversal application of the gates always achieves the correct result, except for one problem: Instead of getting the final state as a superposition of polynomials with degree d, for some gates we end up with the correct logical dit, except that it is encoded with polynomials of a different degree. This can be illustrated if one considers gates which involve multiplication of polynomials, such as the generalized Toffoli gate, where the degree d becomes $2d$. Likewise, the Fourier transform gate takes d to the codegree $m-d-1$. To get back into the code that uses polynomials of degree d, we make use of a procedure called *degree reduction*. In the classical case, degree reduction was used by Ben-Or, Goldwasser, and Wigderson [24] to achieve fault-tolerant classical distributed computation. These techniques can be adapted to the quantum case, as was done in the original version of this paper [3, 4]. Here, however, we present a much simpler construction that reduces the degree by teleportation between states of different degrees. This technique, which was used in [8, 36], is based on ideas in [48].

In order for the above ideas to work, it must be that, even after the degree has changed, the state is still inside a QECC so that errors can be corrected. For this reason we work with codes of length $m = 3d + 1$ such that the quantum polynomial code of twice the degree (which is equal to the codegree) corrects the same number of errors (see Theorem 4).

It is therefore the case that, for polynomial codes, all fault-tolerant procedures can be applied in a remarkably simple manner, namely, transversally with the help of ancilla zero states.

We start with the definition of the set of gates $\mathcal{G}_1$ and continue to general definitions related to encoded gates and to fault-tolerant procedures, such as transversal and semitransversal operations and the spread of errors inside an encoded gate. We then proceed to the description of fault-tolerant encoded gates and finally to fault-tolerant error correction, zero-state preparation, and decoding procedures. We note that much work is put into making these final three procedures measurement-free. At the end of this section we remark about how to simplify procedures when measurements and classical computations can be used.

4.2. The set of gates for polynomial codes $\mathcal{G}_1$. We fix $m = 3d + 1$ in the polynomial codes we use. We work with the following set of gates, which we denote by $\mathcal{G}_1$:

1. $\mathsf{NOT}_p(c)$: $\forall\ c \in F_p$, $|a\rangle \longmapsto |a + c\rangle$ (also denoted B^c).
2. CNOT_p: $|a, b\rangle \longmapsto |a, a + b\rangle$.
3. CNOT_p^{-1}: $|a, b\rangle \longmapsto |a, a - b\rangle$.
4. SWAP: $|a\rangle|b\rangle \longmapsto |b\rangle|a\rangle$.
5. Multiplication by a constant: $0 \neq c \in F_p$: $|a\rangle \longmapsto |ac\rangle$.
6. P^c (generalized phase): $\forall c \in F_p\ |a\rangle \longmapsto w^{ca}|a\rangle$, for $w = e^{2\pi i/p}$.
7. Generalized controlled phase of order c: $\forall c \in F_p\ |a\rangle|b\rangle \longmapsto (w)^{abc}|a\rangle|b\rangle$.
8. $W(c)$ (generalized Fourier transform (of order c)): $|a\rangle \longmapsto \frac{1}{\sqrt{p}} \sum_{b \in F_p} w^{abc}|b\rangle \forall 0 < l < p$.
9. Generalized Toffoli: $|a\rangle|b\rangle|c\rangle \longmapsto |a\rangle|b\rangle|c + ab\rangle$.
10. Adding a qudit in the state $|0\rangle$.
11. Discarding a qudit.

All of the additions and multiplications are in F_p (i.e., modulo p). We will often denote $\mathsf{NOT}_p(1)$ simply by NOT_p. We note that, if the characteristic of the field we work with is not 2, we can replace the generalized Toffoli gate in the above construction

with the following gate:

- Squaring gate: $|a\rangle|b\rangle \longmapsto |a\rangle|b + a^2/2\rangle$.

We get a set of gates which consists of two-qudit gates without any three-qudit gate involved. In either case, the set is proved to be universal in section 6. This set is by no means a minimal set for universality, but the fault-tolerant procedures become simpler and shorter if we have a larger repertoire of fault-tolerant gates that we can use.

4.3. Fault-tolerant procedures—general definitions.

4.3.1. Encoded gates. Say we have a unitary gate g which is applied on the state $|\alpha\rangle$ in the original circuit. We now want to apply the corresponding gate to the state encoding $|\alpha\rangle$. We denote the encoding of $|\alpha\rangle$ by $\phi(|\alpha\rangle)$.

DEFINITION 13. *A sequence of gates Q is said to encode a gate g for the code C, and is denoted by $\Phi(g)$, if, for any superposition $|\alpha\rangle$,*

$$\Phi(g)\phi(|\alpha\rangle) = \phi(g|\alpha\rangle).$$

4.3.2. Transversal and semitransversal gates. The way to apply a gate which is the simplest and which allows the least propagation of errors is the transversal application.

DEFINITION 14. *Consider a gate g on, say, k qudits (k is between 1 and 3 in this paper's case). Consider k blocks of m qudits each. Let us label the qudits in each block from 1 to m (from left to right). We say that the gate g is applied* transversally *on $1, 2$, or 3 encoded blocks if, in order to apply $\Phi(g)$ on k encoded blocks, it suffices to apply the gate g m times, each time on all qudits labeled by the same label.*

We will sometimes need to modify the above transversal construction by just a little bit. Instead of applying the same gate on the set of ith qudits, independent of i, we allow ourselves to apply a gate which depends on the index i. The structure of the circuit remains the same, as in Figure 1.1. We call this way of encoding a gate *semitransversal*. In both cases, it is clear that one fault of a gate during the procedure can affect at most one qubit in each block at the end of the procedure. We now make these notions slightly more precise.

4.3.3. Errors versus faults. To analyze the propagation of errors in our fault-tolerant procedures, we need to make an important distinction between *errors*, which are the actual deviations of the quantum state from being correct, and *faults*, which are the events that occur that cause the qubits to have errors. Let us start by defining what we mean by errors. This requires some definition since the state of one qudit is not well defined in the quantum model, and so we cannot consider one qudit and say that it is "correct" or not.

DEFINITION 15 (deviation). *Consider a density matrix ρ' of a set of qudits B. We say that ρ' is deviated from the correct matrix ρ on the set of qudits $A \subseteq B$ if $\rho'|_{B-A} = \rho|_{B-A}$.*

This definition coincides with the more operative notion of errors which we used in the discussion about error correction in section 3.

CLAIM 1. *Let ρ be a (correct) density matrix of a pure state on a set of qudits B. The matrix ρ' is deviated from ρ on a set of qudits A if and only if there exists a permissible quantum operator on the set A which takes ρ to ρ'.*

Proof. For one direction, suppose that the deviation in ρ' is confined to the set of qudits A. Consider a *purification* of ρ', namely, a state $|\psi'\rangle$ of the set of qudits B plus extra qudits C, such that $|\psi'\rangle\langle\psi'||_B = \rho'$. Such a state exists by, e.g., [68, page 110].

Note that reducing the matrices $|\psi\rangle\langle\psi| \otimes |0\rangle_C\langle 0|_C$, $|\psi'\rangle\langle\psi'|$ to the set of qudits $B - A$ results in the same matrix. By standard results (see, e.g., [68, page 111, Exercise 2.81]), there is a unitary matrix U acting on $C \cup A$ which takes $|\psi\rangle \otimes |0\rangle_C$ to $|\psi'\rangle$. Thus, to get from ρ to ρ', we add the qudits in C in the state $|0\rangle_C$, apply U on $A \cup C$, and discard C. We have designed a permissible operator on A that takes ρ to ρ'. The other direction of the claim is trivial. □

The above claim relates the notions of deviation and error operators. Thus, if ρ is the density matrix of the correct state (which is always a pure state in our construction), we can loosely say that the set of qudits A are the faulty qudits or that the *errors* occurred on the qudits in A. We note that the set A is not uniquely defined. In the rest of the paper, however, in every place where we assume something about the deviation set A, it suffices to choose an arbitrary set of qudits that satisfies the relevant assumption and continue from there.

We will use the above correspondence between error operators and deviations many times in the paper. The way we do this is that we prove, say, that the deviation from the correct state at some stage in our construction is confined to some set of qudits. This implies, by Claim 1, that one can view the density matrix as if an error operator was applied on the deviated qudits, and so all of the results about quantum error correction apply. From now on, therefore, we can talk only about deviation.

The notion of a fault is completely different from the notion of error or deviation. To understand this difference, recall subsection 2.8, where the notions of "locations" and "fault paths" were defined. A *fault* in the circuit is thus the noise operator which is applied at a certain location that appears in the fault path.

We will analyze the effect of faults occurring at certain locations, on the resulting errors in the final states.

4.3.4. Spread. In order to analyze how faults in the circuit affect the errors in the final state, we define the notion of a "spread" of a circuit or a procedure. In most cases, a very simple consideration is required: It is clear that a fault in a location $(q_1, \ldots, q_l, t)$ can affect a qubit q' at time $t' > t$ only if there is a path in the circuit from $(q_1, \ldots, q_l, t)$ to (q', t'). In other words, if we know the correct propagation of some density matrix in the circuit, an additional fault at a certain location can cause a deviation from that correct propagation only in locations affected by the location of the fault via such a path in the circuit. For transversal procedures, it is thus easy to see that one fault can affect at most one qudit in each block. We say that the spread of the procedure is 1.

Unfortunately, for the error correction, zero-state preparation, and decoding procedures, a more careful analysis is required, because the circuit itself is far from being transversal. In such a case one needs to actually take into account the computation performed by the circuit, in order to bound the propagation of errors. It is sufficient for our purposes that the error propagation is limited only when the total number of errors is small. For example, we cannot hope to control the number of errors in the output if the number of errors in the input to an error correction procedure is large.

Definition 16. *We consider procedures that compute on states encoded by a QECC which can correct q errors. We say that the procedure has* spread 1 *if the following holds. Consider a fault path with k faulty locations in this procedure. Suppose that the input state to the procedure is deviated on a set of qudits which has at most f qudits in each of the blocks on which the procedure works. We require that, as long as $f + k \leq q - 1$, adding one additional faulty location to the fault path increases the final*

deviation in each one of the relevant blocks by at most 1 *(while leaving the number of errors in the other blocks unchanged).*

Except for sections 9 and 11, all of our procedures have spread equal to 1. However, there are cases in which more complicated situations arise, where the propagation of errors is confined to a small number of qudits but larger than 1. For example, such is the case when we introduce geometrical constraints to the system, which cause more propagation of errors. The above definition can be generalized to spread equal to ℓ.

DEFINITION 17. *We consider procedures that compute on states encoded by a QECC which can correct q errors. We say that the procedure has* spread ℓ *if the following holds. Consider a fault path with k faulty locations in this procedure. Suppose that the input state to the procedure is deviated on a set of qudits which has at most f qudits in each of the blocks on which the procedure works. We require that, as long as* $(f+k)\ell \leq q-\ell$*, adding one additional faulty location to the fault path increases the final deviation in each one of the relevant blocks by at most* ℓ *(while leaving the number of errors in the other blocks unchanged).*

The proofs of section 7, showing that our general hierarchical scheme (without exact specification of the code being used) is fault-tolerant, are done for the more general case of spread ℓ. This does not impose any additional difficulty in the proof. For a first reading, it is perhaps simpler to keep the definition of spread 1 in mind.

4.3.5. Issues related to ancillas. In some of the procedures, we will use ancilla qudits as extra working space. At the end of the procedure these qudits will be discarded, in order to get exactly the state we need. As was explained in subsection 2.5, we will always discard qudits which are (in an ideal noiseless situation) in tensor product with the rest of the system. In this case the operation of discarding qudits means simply erasing their state, and the resulting state is a pure state. This is necessary if we want to apply unitary operations on the encoded states. We will describe a procedure by specifying what it does to basic states of the code. It is easy to see that if for any input basis state the ancilla qubits at the end of the procedure are in a tensor product with the rest of the qubits, and their state does not depend on the input basis state, i.e.,

$$|S_a\rangle \longmapsto |S_{g(a)}\rangle \otimes |A\rangle, \tag{4.1}$$

where A is independent of a, then for any input superposition for the procedure the ancilla qubits are in tensor product with the rest of the qudits, and thus they can be discarded simply by erasing them.

4.4. Transversal and semitransversal gates for polynomial codes. We begin with the simplest cases.

LEMMA 1. *The first seven gates:* $\mathsf{NOT}_p(c), \mathsf{CNOT}_p, \mathsf{CNOT}_p^{-1}, \mathsf{SWAP}$*, multiplication by a constant, generalized phase, and generalized controlled phase, can all be applied in a transversal or semitransversal manner.*

Proof. It is easy to check that the first five gates, namely, $\mathsf{NOT}_p(c)$, CNOT_p, CNOT_p^{-1}, SWAP, and multiplication by a constant different than zero, can be applied transversally by applying the gate on corresponding qudits from the different blocks. We give here just one example, for the $\mathsf{NOT}_p(c)$ gate:

$$|S_a\rangle = \sum_{w\in C_2} |a+w_1\rangle\otimes\cdots\otimes|a+w_m\rangle \longmapsto \sum_{w\in C_2} |a+c+w_1\rangle\otimes\cdots\otimes|a+c+w_m\rangle = |S_{a+c}\rangle. \tag{4.2}$$

The other four gates are similar. The generalized phase is just slightly more complicated. It can be applied semitransversally. Define c_l as the interpolation coefficients such that

$$\forall\ f \in F[x],\ \deg(f) \le m-1, f(0) = \sum_{i=1}^{m} c_i f(\alpha_i). \tag{4.3}$$

We apply on the lth qudit the gate $|a\rangle \longmapsto w^{c_l a}|a\rangle$. This achieves the desired operation because

$$\begin{aligned} |S_a\rangle &= \frac{1}{\sqrt{p^d}} \sum_{f\in V, f(0)=a} |f(\alpha_1), \ldots, f(\alpha_m)\rangle \\ &\longmapsto \frac{1}{\sqrt{p^d}} \sum_{f\in V, f(0)=a} \Pi_{i=1}^{m} w^{c_l f(\alpha_l)} |f(\alpha_1), \ldots, f(\alpha_m)\rangle \\ &= \frac{1}{\sqrt{p^d}} \sum_{f\in V, f(0)=a} w^a |f(\alpha_1), \ldots, f(\alpha_m)\rangle = w^a |S_a\rangle. \end{aligned} \tag{4.4}$$

Finally, the generalized controlled phase of order c can also be applied semitransversally, by using the same idea. On the lth coordinate we apply the generalized controlled phase of order $c \cdot c_l$. We get

(4.5)

$$\begin{aligned} |S_a\rangle|S_b\rangle &= \frac{1}{p^d} \sum_{f\in V, f(0)=a} |f(\alpha_1), \ldots, f(\alpha_m)\rangle \sum_{g\in V, g(0)=b} |g(\alpha_1), \ldots, g(\alpha_m)\rangle \\ &\longmapsto \frac{1}{p^d} \sum_{f,g\in V, f(0)=a, g(0)=b} \Pi_{i=1}^{m} w^{cc_l f(\alpha_l) g(\alpha_l)} |f(\alpha_1), \ldots, f(\alpha_m)\rangle |g(\alpha_1), \ldots, g(\alpha_m)\rangle. \end{aligned}$$

Since f, g are both of degree at most d, the product of the two polynomials is of degree at most $2d$, and the interpolation applies. If we denote $f \cdot g = h$, we get $\Pi_{i=1}^{m} w^{cc_l f(\alpha_l) g(\alpha_l)} = w^{c\sum_l c_l h(\alpha_l)} = w^{ch(0)} = w^{cf(0)g(0)} = w^{abc}$, as we wanted. ∎

4.5. Rotation to the dual code. The remaining two gates, namely, the generalized Fourier transform and the generalized Toffoli (or the squaring gate), are not as simple, since the transversal application changes the degree of the polynomials involved. To this end we add a superscript d or $2d$ denoting the degree of the polynomials used in the code, as in $|S_a^d\rangle$ or $|S_b^{2d}\rangle$. We start by showing the effect of a semitransversal Fourier transform on a state encoded by using degree d' polynomials. Denote by $w_l = w^{c_l}, l = 1, \ldots, m$, for c_l the interpolation coefficients, as in the proof of Lemma 1. Recall that in our notation

$$W(c_l) : |a\rangle \longmapsto \frac{1}{\sqrt{p}} \sum_{b\in F_p} w_l^{ab} |b\rangle. \tag{4.6}$$

We apply $W(c_l)$ to the lth qudit for all $1 \le l \le m$.

$$|S_a^{d'}\rangle \longmapsto W(c_1) \otimes W(c_2) \otimes \cdots \otimes W(c_m) |S_a^{d'}\rangle. \tag{4.7}$$

Claim 2 (rotation to the dual code). *The transformation of* (4.7) *performs the generalized Fourier transform, except it moves the word to the dual code, namely, the*

polynomial code with the codegree $m-d'-1$*:*

$$|S_a^{d'}\rangle \longmapsto \frac{1}{\sqrt{p}} \sum_{b\in F_p} w^{ab} |S_b^{m-d'-1}\rangle.$$

Proof. Let us denote the final state of the transformation by $|\alpha\rangle$:

$$|S_a^{d'}\rangle = \frac{1}{\sqrt{p^d}} \sum_{f\in V, f(0)=a} |f(\alpha_1),\dots,f(\alpha_m)\rangle \tag{4.8}$$

$$\longmapsto |\alpha\rangle = \frac{1}{\sqrt{p^{d+m}}} \sum_{b_1,b_2,\dots,b_m\in F} \sum_{f\in V, f(0)=a} w^{\sum_{l=1}^m c_l f(\alpha_l) b_l} |b_1,\dots,b_m\rangle.$$

For each string $b_1,\dots,b_m \in F_p$, associate the unique polynomial $b(x)$ which satisfies $b(\alpha_l)=b_l$ and has degree $\deg(b)\le m-1$. The exponent of w in (4.8) can be written in a much simpler form when $b(x)$ is of degree $\deg(b)\le m-d'-1$. For such $b(x)$, the polynomial $h(x)=b(x)f(x)$ is of degree $\deg(h)\le m-1$ so

$$\sum_{l=1}^m c_l f(\alpha_l) b(\alpha_l) = \sum_{l=1}^m c_l h(\alpha_l) = h(0) = f(0)b(0). \tag{4.9}$$

Hence, the sum over all b with $\deg(b)\le m-d-1$ in (4.8) gives

$$\frac{1}{\sqrt{p^{d+m}}} \sum_{b_1,b_2,\dots,b_m\in F,\deg\ b(x)\le m-d-1} \sum_{f\in V, f(0)=a} w^{b(0)f(0)} |b_1,\dots,b_m\rangle \tag{4.10}$$

$$= \frac{1}{\sqrt{p^{m-d}}} \sum_{b_1,b_2,\dots,b_m\in F,\deg\ b(x)\le m-d-1} w^{b(0)a} |b_1,\dots,b_m\rangle$$

$$= \frac{1}{\sqrt{p}} \sum_{b\in F_p} w^{ab} \frac{1}{\sqrt{p^{m-d-1}}} \sum_{b_1,b_2,\dots,b_m\in F,\deg\ b(x)\le m-d-1, b(0)=b} |b_1,\dots,b_m\rangle$$

$$= \frac{1}{\sqrt{p}} \sum_{b\in F_p} w^{ab} |S_b^{m-d'-1}\rangle.$$

Now we claim that the sum over the rest of the b's must vanish. The reason is that the norm of the above vector is 1. Now $|\alpha\rangle$ can be written as a sum of two vectors: the contribution from b's with $\deg(b)\le m-d-1$ and that from the rest of the b's. The two are orthogonal, since different $|b\rangle$'s are orthogonal. Hence, the squared norm of $|\alpha\rangle$, which is 1 (because the operation is unitary and we started with a norm one vector), is the sum of the squared norms of the contribution of $\deg(b)\le m-d-1$, which is also 1, and the norm of the orthogonal vector. Thus, the norm of the sum over b's with $\deg(b) > m-d'-1$ must vanish. ∎

4.6. Degree reduction and degree increase.

DEFINITION 18. *A* degree reduction *is a procedure which takes a state* $|S_a^{d'}\rangle$ *and returns the state* $|S_a^d\rangle$*, for* $d' > d$. Degree increase *is defined similarly, except we require that* $d' < d$.

To construct these procedures, we need to have at our disposal a *zero-state preparation* procedure, namely, a fault-tolerant procedure which generates the state $|S_0^d\rangle$. In fact, we will need a zero-state preparation procedure also for the polynomial code of degree $2d = m-d-1$, so that we have the states $|S_0^{m-d-1}\rangle$ available, too. We

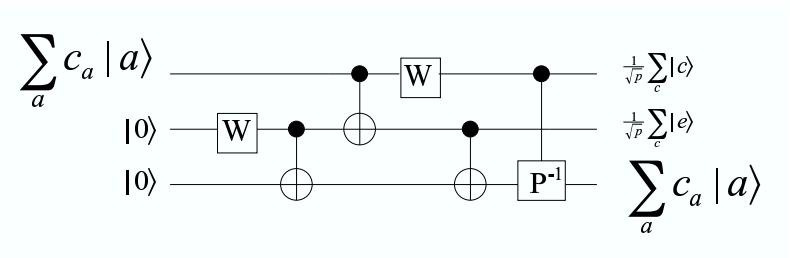


FIG. 4.1. *Teleportation of a general state of the topmost p-qudit to the bottommost one.*

will see how to perform these procedures in subsection 4.13, and for now we merely assume that we have them at our disposal.

The degree increase is derived by using teleportation. The idea is to apply teleportation from a state encoded with polynomials of degree d to a state encoded with polynomials of degree $2d$. We first recall how teleportation works for p-qudits (see Figure 4.1) and then consider its encoded version.

The first qudit is in a general state $\sum_{a\in F_p} c_a|a\rangle$, and the other two qudits are in the state $|0\rangle$. The first step is transforming the last two qudits to the analogue of an EPR pair for F_p: We apply the generalized Fourier transform W on the second qudit and then a CNOT_p^{-1} from the second qudit to the third one. This gives the state

$$\frac{1}{\sqrt{p}} \sum_{a,b\in F_p} c_a|a\rangle|b\rangle|-b\rangle.$$

We then apply CNOT_p from the first qudit to the second, which gives

$$\frac{1}{\sqrt{p}} \sum_{a,b\in F_p} c_a|a\rangle|b+a\rangle|-b\rangle.$$

Next, we apply a generalized Fourier transform on the first qudit to get

$$\frac{1}{p} \sum_{a,b,c\in F_p} c_a w^{ac}|c\rangle|b+a\rangle|-b\rangle.$$

We then apply CNOT_p from the second qudit to the third:

$$\frac{1}{p} \sum_{a,b,c\in F_p} c_a w^{ac}|c\rangle|b+a\rangle|a\rangle.$$

Finally, we apply a generalized controlled phase of order -1 from the first qudit to the third, to get

$$\frac{1}{p} \sum_{a,b,c\in F_p} c_a|c\rangle|b+a\rangle|a\rangle = \frac{1}{p} \sum_{a,e,c\in F_p} c_a|c\rangle|e\rangle|a\rangle = \frac{1}{\sqrt{p}} \sum_{c\in F_p} \otimes \frac{1}{\sqrt{p}} \sum_{e\in F_p} \otimes|c\rangle|e\rangle \otimes \sum_a c_a|a\rangle,$$

which is the desired teleportation.

The degree increase is basically an encoding of the above circuit, by using the correct choice of degrees for each block. We work on three blocks of qudits. The first

block is a general state of one qudit encoded by using polynomial codes of degree d. The second block is initially in the state $|S_0^d\rangle$, and the last block is in the state $|S_0^{2d}\rangle$. All of the gates in the teleportation are now applied transversally, except for the generalized Fourier transform which is applied semitransversally, as in Claim 2. We thus see that, after applying the first two gates in the teleportation circuit transversally, we get that the last two blocks are in the state

$$\frac{1}{\sqrt{p}} \sum_b |S_b^{2d}\rangle |S_{-b}^{2d}\rangle.$$

The next gate, which is the encoded CNOT_p, is now applied from a word in the code of degree d to that of degree $2d$. It is easy to check that this does not matter to the correctness of this gate. After the next step, namely, the semitransversal generalized Fourier transform, Claim 2 implies that all blocks are encoded by using degree $2d$ polynomials, and so the correctness follows from the correctness of the teleportation circuit.

The construction we have shown for degree increase does not work for degree reduction, since, for example, the first CNOT_p does not work correctly if the target state is of a smaller degree than that of the control state. To bypass this problem, we can instead perform a degree increase in the dual code. This is done as follows. We start with a state of degree $2d$. We first apply the semitransversal generalized Fourier transform (4.7). This takes us to the dual code (of degree d), by Claim 2. We now apply a degree increase, to get the same state encoded by using polynomials of degree $2d$. Finally, we apply the reverse of the transformation in (4.7), which achieves the desired result, again, by Claim 2.

4.7. Fault-tolerant Fourier transform. To achieve the generalized Fourier transform of order 1, we simply apply the transformation of (4.7). By Claim 2, this yields the desired state but in the wrong degree. Applying a degree reduction solves the problem. To achieve a generalized Fourier transform $W(c)$ of order $c \neq 1$, we use w^c instead of w everywhere in subsection 4.5.

4.8. Fault-tolerant generalized Toffoli and squaring. To apply the generalized Toffoli gate on $|S_a\rangle|S_b\rangle|S_c\rangle$, we first increase the degree of the third register to $2d$. We now apply the general Toffoli gate transversally on the m coordinates, which gives $|S_a\rangle|S_b\rangle|S_{ab+c}^{2d}\rangle$, as is easy to check. We now apply a degree reduction on the third register, and this achieves the desired result. The squaring gate is applied in exactly the same manner.

4.9. Remaining gates. The fault-tolerant version of the gate that adds a qubit in the state $|0\rangle$ is simply the zero-state preparation procedure. We will see how to perform this procedure in subsection 4.13. The fault-tolerant gate that discards a qudit can obviously be done transversally by discarding all of the qudits in the block. This completes our description of fault-tolerant procedures for the set $\mathcal{G}_1$.

4.10. Fault-tolerant error correction. Our construction of the error correction procedure is based on the simplest known quantum error correcting technique, namely, Steane's error correction [90]. Once again, we assume here that we have at our disposal a fault-tolerant zero-state preparation procedure, which we will show later.

The error correction procedure is composed of two stages: The first stage detects and corrects dit-flips (faults of type B), by using classical error correction techniques for the code C_1. The second stage applies on each coordinate the generalized Fourier

transform (4.7). We then correct dit-flips by using classical error correction techniques for the code C_2. Finally, we rotate back by applying the inverse of the generalized Fourier transform on each coordinate. By subsections 3.5 and 3.6 this achieves the desired error correction. It is therefore sufficient to describe how to correct dit-flips fault-tolerantly.

Let us first describe one part of the error correction procedure. Generate an ancilla block in the state $|S_0^{2d}\rangle$, by using the fault-tolerant zero-state preparation procedure. Apply a generalized Fourier transform semitransversally, as in (4.7), on the ancilla block. Its state is now $\frac{1}{\sqrt{p}}\sum_{a\in F_p}|S_a^d\rangle$. Then apply CNOT_p transversally from the block we would like to correct to the ancilla blocks. Now observe that there is a classical circuit that, given any string from the ancilla block, can output the value of e_i, the ith dit of the error vector on the original block, as long as the number of faults is at most q, the number of errors that C_1 corrects. To see this, note that, for any $w \in C_2, w' \in C_1$, and e a word in F_p^m, the transversal CNOT_p takes

$$|w+e\rangle|w'\rangle \mapsto |w+e\rangle|w'+e\rangle,$$

and we can find e by finding the closest word in C_1 to the string $w'+e$.

To correct dit-flips, we repeat the above procedure m times: Generate m ancilla states, rotate them to the dual code, apply CNOT_p transversally from the computational block to each one of the ancilla blocks, and then apply on the ith ancilla block a circuit that outputs $p-r$, where B^r is the dit-flip that occurred on the ith coordinate. Finally, apply a CNOT_p from this output dit into the ith coordinate of the computation block.

By ignoring the zero-state preparation procedures in the above construction for now, it is easy to see that one fault in the above procedure can affect only one qudit in the final block.

4.11. Error correction which projects any word into the code. We would like to insert here one important modification, which is not necessary for the fault-tolerant error correction but will become crucial when applying the error corrections in the recursive scheme. This is the requirement that the error correction takes any word to some word in the code, regardless of the number of faults in the original state. This requirement applies, of course, only if during the error correction there are no faults.

Roughly, this is done by checking whether too many errors occurred and, if so, replacing the entire block by another block which is initialized in the state $|S_0\rangle$. However, we should be careful to keep the procedure fault-tolerant. We do this in the following way: Before starting the correction procedure, we generate another ancilla state $|S_0\rangle$, by using the state preparation procedure. When computing the value of e_i in the error correction procedure, we also compute whether the total number of faults is at most q and write the answer on another qudit. Let us call this an *indication bit*. The CNOT which checks if the ith dit is wrong and if so applies NOT on the i*th* qubit is replaced by a generalized Toffoli gate which takes as an input also the ith indication bit and also checks if the number of faults is at most q. We then swap the ith qudit with the ith qudit of the state $|S_0\rangle$, conditioned that the ith indication bit indicates that the number of faults is larger than q. Such a conditional swap can be achieved by a small circuit which uses only the classical gates from $\mathcal{G}_1$.

To see that this procedure indeed takes any word to some word in the code, observe that this is true if no fault occurs during the procedure itself. Since the procedure is performed fault-tolerantly, the final state will differ from a word in the code only in the qubits affected by an error.

4.12. More issues regarding ancillas. In subsection 4.3.5 we required that, in all of our procedures, the state of the ancilla qudits that are discarded at the end of a procedure is in tensor product with and independent of the state of the computer (in a noiseless situation). This requirement was imposed so that the ancilla states are in tensor product with the state of the computer even if this state is a superposition of the basis states and not only in a basis state like in our analysis. This requirement can be released in two cases: the zero-state preparation and the decoding procedures.

The zero-state preparation procedure is supposed to get as an output one basis state. Thus, we can release the above requirement about ancilla qudits and demand only that the ancilla state is in tensor product with the computational qubits at the end of the procedure. This requirement can also be released in the decoding procedure, since once we decode a state in our scheme, we do not use it any more. Hence, we should check only that it gives the correct answer when measured.

4.13. Fault-tolerant zero-state preparation. We need to show how to generate $|S_0\rangle$ with spread 1. By Definition 16, we need to check the propagation of errors only under the assumption that the total number of errors in the input string, plus the number of faults during the procedure, is at most q, the number of errors that the codes can correct. In this case, we can assume that the number of faults is at most q, since the zero-state preparation has no input. The construction is based on one basic design, which is essentially concatenated with itself, up to some modifications.

4.13.1. Zero-state preparation resilient to one fault. Assume for a moment that $q = 1$, so that we only have to make sure that the spread of the state preparation procedure is 1 if there is one fault in the procedure. In this situation, our state preparation procedure is the following construction. Let $\mathcal{Q}$ be some quantum circuit that generates $|S_0\rangle$ from $|0^m\rangle$, without any fault-tolerant requirement. We start with $5m$ blank qubits and apply $\mathcal{Q}$ on the first, second, third, fourth, and fifth m-tuples of qubits. We now apply a circuit that is very similar to the error correction circuit: It attempts to detect errors with respect to the code containing one word, namely, $|S_0\rangle$. To detect dit-flips in the first block, we apply CNOT_p transversally from the first block to the second one and then perform some computation on the qudits of the second block to assess the number of dit-flips. This time, we do not attempt to infer the exact error vector, or to correct the state, but just to decide whether we accept the state or not. To do this, we copy each dit in the block to m different dits. The circuit we apply on the second block gets as an input the ith copy of these dits, namely, a string in F_p^m, and outputs whether its distance from the code C_2 is more than 1 or not. The output bit is called the *dit-flips indication bit*. We perform m independent calculations to get m such indication bits.

We use the third block to detect phase-flips. We would like to follow a similar construction as for the detection of dit-flips. The difference is that we first rotate both the first block and the third block to the dual code, as in (4.7). After we apply CNOT_p from the first to the third block, we detect errors by using the third block, except that the error detection is done with respect to the code C_1 for polynomials with the codegree. Unfortunately, this scheme requires some modification. The problem is that if one fault occurred in the generation of the third block, such that after its rotation this block contains many phase-flips, these phase-flips will propagate through the CNOT_p gates to the first block but will not be detected. To prevent this, before we apply the above construction, we use the fourth block to detect for dit-flips of the third block. This is done just as the above dit-flip detection with the first and second blocks. This initial check creates m additional indication bits. We condition each of

the CNOT_p gates from the first to the third block on the relevant additional indication bit—hence instead of applying a transversal CNOT_p, we in fact apply generalized Toffoli gates transversally. After the application of the Toffoli gates, we rotate the first block back to the dual of the dual code, namely, to the original code. We now compute from the third block m phase-flip indication bits, independently, exactly as we have computed the dit-flip indication bits before.

In fact, we need to insert one last modification in this construction. We would like to be able to bound the total number of dit-flips and phase-flips, and not just each of them separately, so that we can bound the deviation. Hence, instead of the two separate indication bits, we actually calculate one indication bit, as follows. The circuit gets as an input two strings in F_p^m. It checks whether there exists one coordinate such that it suffices to change this coordinate in both strings, in some way (including leaving it as is), to put the first string in C_2 and the second string in C_1 (with the codegree). If there is no such coordinate, this bit is turned to 1.

Finally, we apply transversally the following three-qudit transformation, on the ith dit of the first block, the ith indication bit, and the ith dit of the fifth block. The transformation swaps the ith dits of the first and fifth blocks, conditioned that the indication bit is 1 (namely, more than one error was detected). This three-qudit transformation can be constructed from classical gates in $\mathcal{G}_1$.

CLAIM 3. *If exactly one fault in the above construction occurred, then the final state has at most one error.*

Proof. By subsections 3.5 and 3.3 we can treat the error as if it is a linear combination of dit-flips and phase-flips. Let us first consider the case of a fault occurring in one of the non-fault-tolerant circuits preparing the states $|S_0\rangle$. If the fault occurs in the fifth circuit, nothing happens, since no swap occurs. If the fault occurs in the first circuit, it could be that this fault propagated to more than one qudit in $|S_0\rangle$. If there is more than one coordinate in which either a dit-flip or a phase-flip occurred, the remainder of the circuit, which is fault-free, will detect it. Hence, all indication bits would indicate that the first and fifth blocks should be swapped. If the fault propagated to at most one qudit, this would not cause a swap, but this is OK since there will only be one error in the final state.

A similar argument applies if the fault occurs in the second block. If there is more than one faulty qudit in the $|S_0\rangle$ state of this block, this will result in swapping the first block with the correct state (the fifth block). If there is one error, this will not cause a swap but can only cause one error in the final state.

What about the third block? If the $|S_0\rangle$ state of the third block has more than one dit-flip, these will be detected in the dit-flip preliminary detection of the third block, and the conditioning on the additional indication dits will prevent them from propagating. Hence, we can assume that there is at most one dit-flip in this state. One such dit-flip can propagate from the third block to the first block through the CNOT_p gates, after the rotation, but can cause at most one error. Now let us add phase-flips. After the rotation of the third block, these transform to dit-flips. If there are at least 2 of them, a swap will occur and no error will be caused in the final block. If there is at most one such dit-flip, it has no affect at all since it does not propagate through the CNOT_p gates. Overall, we have seen that one fault of any kind in the third block can cause at most one error in the final block.

Similar arguments apply to show that one fault in the fourth block can cause at most one error. If the fourth $|S_0\rangle$ state has more than one dit-flip, nothing will happen due to the additional indication bits. But even if it has at most one dit-flip, this does not propagate through the CNOT_p gates from the third to the fourth block,

and so the dit-flips have no effect. As for the phase-flips, these can propagate to the third block, and the argument proceeds as in the case of the third block.

The next case is a fault that occurs during the CNOT_p and rotation gates. Since these are applied transversally, one fault can affect only exactly one qudit in each block. Let us see how these errors propagate through the remaining fault-free circuit. As for the propagation of a dit-flip, since one dit-flip cannot cause the indication bits to turn into 1, we have that all indications bits will be zero. As for a phase-flip, this does not propagate at all through the remaining gates. Hence, as no swap will occur, the final state will have at most one error.

A more subtle case is the case in which the fault occurred during the copying of the bits of the second (or third, or fourth) block. Suppose a fault occurred while copying the ith bit. It could be that the ith indication dit is flipped and/or had a phase flip, and, possibly, the ith dit in the second block is flipped, too, and/or suffers from a phase-flip. Let us first consider just the dit-flips and concentrate on a dit-flip in the copied dit itself. The problem is that in this case the remaining copied dits will all be wrong! Fortunately, this does not pose a problem. This affects exactly one dit in the input string of each circuit that computes an indication bit. Because of the fact that the indication bits turn to 1 only if they see more than one error, all indication dits (except, possibly, for the one in which the fault actually occurred) will still be 0. So in this case no swap occurs, except for possibly in one qudit, and the state, which was error-free to begin with, will have at most one error. An extra phase-flip in the same location of the dit-flip, if one occurred, does not change this analysis. A phase-flip in the copied qudit itself does not propagate via the CNOT_p gates to the target dit. Hence, by the end of the copying stage, the error will still be confined to the qudits where the fault had occurred, namely, one qudit in the second or third or fourth block, and one indication bit. So once again, at most one qudit will be swapped. Note that if just a phase-flip occurred in the indication bit, this does not affect the final state at all because it does not propagate through the SWAP gates.

Finally, a fault in the circuit calculating one of the indication bits, or a fault in one of the three-qudit circuits, can cause only one error since the remainder of the circuit is transversal. □

4.13.2. Zero-state preparation resilient to q faults. Let us now consider the case of $q > 1$. The above construction does not work any more, even for $q = 2$, since two faults can ruin completely both the first and the last block, which are constructed in a non-fault-tolerant way. Instead, we apply the above construction in a way that is sort of concatenated. For $q = 2$, we concatenate it once with itself, in the following way: We first apply the above construction five times, to get five $|S_0\rangle$ states—we call these states the first-level states. We then apply the above construction once more, except for two changes. The first change is that, instead of applying the original non-fault-tolerant circuit to construct the five $|S_0\rangle$ states, we use the five first-level states that we have generated. The second change is that we set the threshold of the indication bits in the second-level construction to be 2 instead of 1; namely, we only flip the indication bit to 1 if more than *two* errors were detected.

To increase q to any value, we simply apply the above concatenation q times, each time by using as input states the final states from the previous level of concatenation and increasing the threshold for the indication bits by 1. Overall, the scheme that allows for q faults, denoted by Q_q, uses 5^q $|S_0\rangle$ states.

CLAIM 4. *If $k \leq q$ faults occur during the application of Q_q, the final output block will have at most k errors.*

Proof. We prove this by induction. The case of $q = 1$ was essentially given in Claim 3, where it was shown that if $k = 1$, then the final block will have at most one error. It is obvious that if $k = 0$, the state will have no error. Now assume for q, and prove for $q + 1$. We divide this proof into two cases.

In the first case, in each one of the five q-level blocks that are used to generate the final block, there are at most q faults. In this case we can apply the induction. Let $x_1, \ldots, x_5$ be the number of faults occurring in the five circuits generating the q-level blocks, respectively. By induction, the number of errors in the final states is at most $x_1, \ldots, x_5$, respectively. If x is the number of faults occurring in the final part of the circuit, we have that $x_1 + \cdots + x_5 + x = k$. The next step in the circuit involves CNOT_p gates, which do not increase the number of errors in each block beyond k. The next step is the copying of each dit m times and computation of the indication bits. Recall that, at this step, the threshold for the indication bit to flip is for the number of errors it sees to be larger than $q + 1$. However, the calculation of each indication bit gets as an input two strings, such that the union of their errors is confined to at most $k \leq q + 1$ qudits, and so, unless the calculation itself involves a fault, the indication bit will remain 0. Hence, in this stage, too, one fault can propagate only to one error. The remainder of the circuit is transversal. This means that the total number of faults in the final $q + 1$-level block is at most k.

In the other case, $q + 1$ faults occurred during the generation of one of the five q-level blocks which are input to the final $(q + 1)$ level. In this case, the remainder of the circuit is fault-free. We consider five subcases: The faulty block is the first, second, third, fourth, or fifth $|S_0\rangle$s. Suppose first that it is the first block. Then either the final state of the faulty block has more than $q+1$ errors, and then the entire block will be swapped with the fifth block and the final state will be error-free, or the final state of the first block has $q + 1$ errors or less, in which case it will not be swapped, but still the number of errors is at most $k = q + 1$. A similar argument works if the faulty block is the second block. Suppose that there are x dit-flips, and suppose that y phase-flips have propagated to the first block. If $x + y > q + 1$, there will be a swap. Otherwise, this will cause at most $k = q + 1$ errors. Suppose that the faulty block is either the third or the fourth block. Observe that phase-flips before the rotation become dit-flips, and these do not propagate from the third block to the first block. Either there are more than $q+1$ of these phase-flips, and they cause a SWAP, or they do not affect the final state. Hence, we need consider only the effect of the at most $q + 1$ dit-flips, which after the rotation become phase-flips, and can propagate to at most $q + 1$ errors in the final state. Finally, if the faulty block is the fifth block, the final state will not be swapped, so there will not be errors at all. ∎

This implies that the zero-state preparation procedure has spread 1.

4.14. A fault-tolerant decoding procedure. A decoding procedure applies

$$|S_a\rangle \longmapsto |A_a\rangle|\vec{a}\rangle, \tag{4.11}$$

where the state $|A_a\rangle$ is an ancillary state which depends on a. (We can discard this state at the end of the procedure; we will see that, whenever the decoding procedure is applied, the state encodes a well-defined logical bit a.) To apply this transformation, we compute a independently m times from the state $|S_a\rangle$. To do this, we add m^2 blank qubits and copy each qudit from $|S_a\rangle$ m times to m different blank qudits, by using m CNOT_p gates. We get m "copies" of $|S_a\rangle$. These are, of course, not real copies of $|S_a\rangle$, since they are entangled. However, each word in the classical code is copied m times. On each copy of the word we apply the quantum analogue of the

classical computation that computes what is the logical bit that the word encodes. The answer, which is a if no error occurred, is written on another blank qubit. For this computation we use the classical gates in $\mathcal{G}_1$. We might need some extra blank qudits as working space. The computation of a is done in the shortest way possible, regardless of whether it is fault-tolerant. An error in this computation can affect only the one copy of a which it computes. A fault in the first stage of copying the qubits m times can affect only one qubit in each of the copies, and, if the number of errors in S_a plus the number of faults in the first stage is smaller than the number of errors correctable by the code, these faults have no effect. One fault in the second stage of the procedure, during the computation of one of the a's, can affect only the correctness of that a.

Remark 1. We remark regarding the significantly simpler version of the error correction, decoding, and zero-state preparation procedures, in case measurements and classical computation are allowed.

In the error correction procedure, no repetition is required, and one ancilla state is needed for either dit-flips or phase-flips. This saved a factor of m in the construction.

The simplification is most notable in the case of the zero-state preparation procedure. In this case, we can omit the entire concatenated construction. We use the circuit as in the case of $q = 1$ and simplify it further as follows. Observe that, after the CNOT_p gates from the computational block to the ancilla blocks it suffices to measure every qudit in the ancilla blocks and perform the calculation of the error vector classically, without copying the dits first. Hence, the construction is transversal except for the classical computation which does not introduce errors, and the spread is 1.

Finally, in the decoding procedure we can omit the copying of the dits.

5. Fault-tolerant gates for CSS codes over F_2. In this section we give an alternative construction to the one using polynomial codes. Here we use a restricted class of CSS codes, which were used by Shor in [84]. Shor showed how to apply a universal set of gates (which we denote by $\mathcal{G}_2$) on states encoded by such codes. It turns out that almost all of the gates in $\mathcal{G}_2$ can be applied transversally. The only complicated procedure is the Toffoli gate. We repeat the constructions of Shor for the simple gates for completeness. As for the Toffoli gate, we essentially use Shor's construction, except that we adopt it to our framework in which no measurements are allowed, which requires some extra work.

5.1. Some restrictions on the CSS codes. In the following, we will put some restrictions on the CSS codes which we will use. This is done in order to be able to apply several gates transversally, as will be seen shortly. We start with C_1, a punctured doubly even self-dual code, and set $C_2 = C_1^{\perp}$. A punctured self-dual code is a code which is obtained from a self-dual code C' (namely, a code for which $C' = C'^{\perp}$) by deleting one coordinate. We also require that C' is a doubly even code; i.e., the weight of each word in the code is divisible by 4. To see that in this case $C_2 = C_1^{\perp} \subset C_1$, as in the definitions of CSS codes, observe that if $v \perp C_1$, then $v0 \perp C'$, so $v0 \in C'^{\perp} = C'$, so $v \in C_1$. We will denote that $C_1 = C$ and $C_2 = C^{\perp}$.

We now claim that there are only two cosets of $C^{\perp}$ in C. If the length of C is m, then $\dim(C'^{\perp}) = \dim(C') = (m+1)/2$. Hence $\dim(C) = (m+1)/2$ as well, since $|C| = |C'|$, because no two words in C' are mapped to the same word in C by the punctuation. Hence, $\dim(C^{\perp}) = m - (m+1)/2 = (m-1)/2$, and so $\dim(C) - \dim(C^{\perp}) = 1$. Observe that C includes the all-one vector: $\vec{1} \in C$. This is true since $1^{m+1} \in C'^{\perp}$, because C' is even, and since C' is self-dual, $1^{m+1} \in C'$. Hence $1^m \in C$. Observe also that the length m must be odd due to the above considerations.

This implies that $\vec{1} \notin C^{\perp}$. The two code words in our quantum code can thus be written as

$$|S_{\vec{0}}\rangle = \sum_{w \in C^{\perp}} |w\rangle, \qquad |S_{\vec{1}}\rangle = \sum_{w \in C^{\perp}} |w + \vec{1}\rangle. \tag{5.1}$$

$|S_{\vec{0}}\rangle$ and $|S_{\vec{1}}\rangle$ can be thought of as encoding $|0\rangle$ and $|1\rangle$, respectively. We will make use of the fact that $\vec{a} \cdot \vec{b} \bmod 2 = ab$, for $a, b \in 0, 1$, and that $\vec{a} + \vec{b} = \overrightarrow{a+b}$. This fact allows us to shift easily between operations on vectors and operations on the bits they represent. We will therefore usually omit the vectors in the notations of $|S_{\vec{0}}\rangle$ and $|S_{\vec{1}}\rangle$, unless there is ambiguity.

5.2. The set of gates for CSS codes $\mathcal{G}_2$. We work with the following set of gates, which we denote by $\mathcal{G}_2$:

1. NOT: $|a\rangle \longmapsto |1 - a\rangle$.
2. CNOT: $|a, b\rangle \longmapsto |a, a + b\rangle$.
3. Phase: $|a\rangle \longmapsto i^a |a\rangle$.
4. SWAP: $|a\rangle|b\rangle \longmapsto |b\rangle|a\rangle$.
5. Controlled phase: $|a\rangle|b\rangle \longmapsto (-1)^{ab}|a\rangle|b\rangle$.
6. Hadamard: $|a\rangle \longmapsto \frac{1}{\sqrt{2}} \sum_b (-1)^{ab} |b\rangle$.
7. Toffoli gate: $|a, b, c\rangle \longmapsto |a, b, c + ab\rangle$.
8. Adding a qubit in the state $|0\rangle$.
9. Discarding a qubit.

All of the additions and multiplications above are in F_2 (i.e., modulo 2). Section 6 shows that this set of gates is universal. We remark here once again that, like the set $\mathcal{G}_1$, the set $\mathcal{G}_2$ is by no means a minimal universal set of gates, but the fault-tolerant procedures become simpler and shorter if we have a larger repertoire of fault-tolerant gates that we can use. The following theorem shows how to perform gates from $\mathcal{G}_2$ on encoded states fault-tolerantly.

THEOREM 6. *Fix a CSS code that satisfies the restrictions of subsection* 5.1. *For any gate in $\mathcal{G}_2$, there exists a fault-tolerant procedure with spread* 1 *that computes the gate on states encoded by the code. There exist also fault-tolerant error correction, decoding, and zero-state preparation procedures for this code, which all have spread* 1. *Moreover, all of these procedures use only gates from $\mathcal{G}_2$ and in particular do not use measurements.*

The constructions of the error correcting, decoding, and zero-state preparation procedures follows exactly the construction of section 4. It remains to show the constructions of the computational gates.

5.3. Transversal gates. In the following we omit overall normalization factors, since all vectors are known to be unit vectors. We also set $a, b \in C/C^{\perp}$. We start with the NOT gate:

$$\begin{aligned} |S_a\rangle &= \sum_{w \in C^{\perp}} |a_1 + w_1\rangle \otimes \cdots \otimes |a_m + w_m\rangle \\ &\longmapsto \sum_{w \in C^{\perp}} |a_1 + 1 + w_1\rangle \otimes \cdots \otimes |a_m + 1 + w_m\rangle = |S_{a+\vec{1}}\rangle. \end{aligned} \tag{5.2}$$

For CNOT,

(5.3)
$$\begin{aligned}|S_a\rangle|S_b\rangle &= \sum_{w\in C^\perp} |a_1+w_1\rangle\otimes\cdots\otimes|a_m+w_m\rangle \sum_{w'\in C^\perp} |b_1+w'_1\rangle\otimes\cdots\otimes|b_m+w'_m\rangle\\ &\longmapsto \sum_{w\in C^\perp} |a_1+w_1\rangle\otimes\cdots\otimes|a_m+w_m\rangle \sum_{w'\in C^\perp} |a_1+b_1+w_1+w'_1\rangle\otimes\cdots\\ &\quad\otimes|a_m+b_m+w_m+w'_m\rangle\\ &= |S_a\rangle|S_{a+b}\rangle,\end{aligned}$$

where the last equality follows from the fact that $C^\perp$ is a linear subspace, and therefore summing over $w+w'$ for a fixed w in the code is equivalent to summing over w'. For the phase gate, apply the gate $|a\rangle \longmapsto i^a|a\rangle$ three times on each coordinate. This gives

(5.4)
$$|S_a\rangle = \sum_{w\in C^\perp} |a_1+w_1\rangle\otimes\cdots\otimes|a_m+w_m\rangle \longmapsto \sum_{w\in C^\perp} i^{3(\sum_k a_k+w_k)}|a_1+w_1\rangle\otimes\cdots\otimes|a_m+w_m\rangle.$$

This is the desired result, because of the following fact. Since C is obtained from a doubly even self-dual code C' by deleting one coordinate, it is easy to see that $C^\perp \subset C$ are exactly those words in which the deleted coordinate was 0. Thus, all words in $C^\perp$ have weight which is divisible by 4, and all words in C but not in $C^\perp$ have weight which is 3 mod 4.

For the encoded controlled phase gate,

$$|S_{\vec a}\rangle|S_{\vec b}\rangle = \sum_{w,w'\in C^\perp} |\vec a+w\rangle|\vec b+w'\rangle \longmapsto \sum_{w,w'\in C^\perp} (-1)^{(\vec a+w)\cdot(\vec b+w')}|\vec a+w\rangle|\vec b+w'\rangle. \tag{5.5}$$

Now $(\vec a+w)\cdot(\vec b+w') = \vec a\cdot\vec b \bmod 2$. This is true since $\vec a\in C$ and $w'\in C^\perp$ so $\vec a\cdot w' = 0 \bmod 2$, and likewise $w\cdot\vec b = w\cdot w' = 0 \bmod 2$. Moreover, $\vec a\cdot\vec b \bmod 2$ is equal to ab, and so the final state is indeed the desired state. Finally, for the Hadamard gate,

$$\begin{aligned}|S_a\rangle &= \sum_{w\in C^\perp} |a_1+w_1\rangle\otimes\cdots\otimes|a_m+w_m\rangle \longmapsto \sum_{x\in F_2^m}\sum_{w\in C^\perp} (-1)^{(a+w)\cdot x}|x\rangle\\ &= \sum_{x\in C}(-1)^{a\cdot x}|x\rangle = \sum_{b\in C/C^\perp}\sum_{w\in C^\perp}(-1)^{a\cdot(b+w)}|b+w\rangle = \sum_b (-1)^{a\cdot b}|S_b\rangle.\end{aligned} \tag{5.6}$$

It remains to show how to apply the Toffoli gate on encoded states.

5.4. The fault-tolerant Toffoli gate. To apply the Toffoli gate, we roughly follow Shor's scheme, where we construct an ancillary state denoted by $|A_0\rangle$ and use it to obtain the Toffoli gate. The main difference from Shor's scheme is in the construction of the ancillary state, which is not completely straightforward if one wants to avoid using measurements.

Our constructions of encoded gates for polynomial codes, which we discussed in section 4, is much simpler than the construction we get for the Toffoli gate. First, for the Toffoli gate we need not only zero-state preparations but also the preparation of a three-block ancilla state. Second, the subsequent operations given the ancilla state are not transversal as in the case of the polynomial codes. The work of [48]

shows how to encode a universal set of gates by procedures which consist of ancilla state preparation followed by transversal operations (namely, teleportation). This, however, is done by assuming measurements and classical computation.

5.4.1. Construction of the ancilla state $|A_0\rangle$. Define the ancilla state

$$|A_0\rangle = \frac{1}{2}\sum_{a,b}|S_aS_bS_{ab}\rangle. \tag{5.7}$$

We also define

$$|A_1\rangle = \frac{1}{2}\sum_{a,b}|S_aS_bS_{1-ab}\rangle, \tag{5.8}$$

which is easily convertible to $|A_0\rangle$ by applying NOT on the third block. Note that the state

$$\frac{1}{\sqrt{2}}(|A_0\rangle + |A_1\rangle) = \frac{1}{2\sqrt{2}}(|S_0\rangle + |S_1\rangle)(|S_0\rangle + |S_1\rangle)(|S_0\rangle + |S_1\rangle) \tag{5.9}$$

is easy to construct by preparing three zero states $|S_0\rangle$ and then applying an encoded Hadamard on each block. In order to convert this state to $|A_0\rangle$, we use states of the form

$$|S_{cat}\rangle = \frac{1}{\sqrt{2}}(|S_0\rangle^m + |S_1\rangle^m), \tag{5.10}$$

which we call encoded cat states. An encoded cat state can be achieved by applying m zero-state preparation procedures to get $|S_0\rangle^m$, followed by an encoded Hadamard on the first block, and then CNOT gates from the first block to all other blocks.

We will also make use of the transformation

$$|S_a\rangle^m|A_b\rangle \longmapsto (-1)^{ab}|S_a\rangle^m|A_b\rangle \tag{5.11}$$

for bits a, b. The transformation (5.11) is performed by applying

$$|S_a\rangle|b\rangle|c\rangle|d\rangle \longmapsto (-1)^{a(bc+d)}|S_a\rangle|b\rangle|c\rangle|d\rangle \tag{5.12}$$

on the ith block in the encoded cat state and the ith bit in each of the three blocks of $|A_0\rangle + |A_1\rangle$. By applying this transformation for $1 \le i \le m$ we get

$$|S_a\rangle^m|S_b\rangle|S_c\rangle|S_d\rangle \longmapsto (-1)^{a(bc+d)}|S_a\rangle^m|S_b\rangle|S_c\rangle|S_d\rangle, \tag{5.13}$$

which is exactly the desired transformation of (5.11). Note that transformation (5.12) need not be fault-tolerant. We do not care if one fault ruins the entire block $|S_a\rangle$.

Let us start with the state

$$\frac{1}{\sqrt{2^{r+1}}}(|S_0\rangle^m + |S_1\rangle^m)^r(|A_0\rangle + |A_1\rangle), \tag{5.14}$$

where r will be chosen soon. Now apply transformation (5.11) between each one of the r encoded cat states and the last register, and between every two subsequent applications of transformation (5.11), perform an error correction on the last register.

The reason for the error corrections will become clear shortly. The resulting state would then be

$$\frac{1}{\sqrt{2^{r+1}}}(|S_0\rangle^m + |S_1\rangle^m)^r|A_0\rangle + \frac{1}{\sqrt{2^{r+1}}}(|S_0\rangle^m - |S_1\rangle^m)^r|A_1\rangle. \tag{5.15}$$

We would now like to "measure" (without any measurement) the r cat states, in order to decide (essentially, by taking majority over the r "measurement" results) whether the cat states are in their plus or minus states. If the answer is "plus," we would know that the state of the final three registers is $|A_0\rangle$, the desired state, and if "minus," we would know it is $|A_1\rangle$, in which case we could apply NOT on the third block to get $|A_0\rangle$.

We can now explain why the error corrections were added in between the different applications of transformation (5.11) that led to the state of (5.15): Transformation (5.11) is derived by m applications of transformation (5.13). Hence, a single error in the last register can cause a flip of sign in the encoded cat state, from "plus" to "minus" or vice versa. If this single error is not corrected, the signs will be wrong in all of the remaining cat states, too.

To apply the measurement of the sign of one encoded cat state in a fault-tolerant manner, we observe that applying encoded Hadamard gates on all m blocks in one encoded cat state results in

$$\begin{aligned}\frac{1}{\sqrt{2}}(|S_0\rangle^m + |S_1\rangle^m) &\longmapsto \frac{1}{\sqrt{2^{m-1}}} \sum_{i=0, i\cdot\vec{1}=0}^{2^m} |S_{i_1}\rangle|S_{i_2}\rangle \dots |S_{i_m}\rangle, \\ \frac{1}{\sqrt{2}}(|S_0\rangle^m - |S_1\rangle^m) &\longmapsto \frac{1}{\sqrt{2^{m-1}}} \sum_{i=0, i\cdot\vec{1}=1}^{2^m} |S_{i_1}\rangle|S_{i_2}\rangle \dots |S_{i_m}\rangle.\end{aligned} \tag{5.16}$$

This means that the answer plus or minus is determined by the parity of the strings in the sum. To compute the parity fault-tolerantly, we apply a fault-tolerant decoding procedure on each block, to get a string of m (ideally equal) bits. We then compute the parity transversally: We consider all of the first bits in the blocks (there are m of them) and compute their parity, by using a quantum circuit made of Toffoli, CNOT, and NOT gates. Likewise, we consider all of the second bits and compute their parity, and so on. For each encoded cat state, we therefore get m parity bits. We compare the parity bits of the different encoded cat states transversally by applying m majority votes of the form:

$$|a_1, a_2, \dots, a_r, b\rangle \longmapsto |a_1, a_2, \dots, a_r, b + maj(a_i)\rangle. \tag{5.17}$$

Each such majority vote can be constructed once again from Toffoli, CNOT, and NOT gates, since they are universal for classical computations. Finally, we apply CNOT transversally from the result of the majority vote to the qubits in the third block of $|A_0\rangle$.

It is left to see that the spread of this procedure is 1. We will consider errors at different stages of the procedure. The construction of an encoded cat state allows one fault to propagate to one error since it is composed of the state preparation procedure and CNOTs applied transversally. This is true also for the construction of the state $\frac{1}{\sqrt{2}}(|A_0\rangle + |A_1\rangle)$ because we used only the fault-tolerant state preparation procedure and the transversal Hadamard gate. A more subtle consideration is required for

transformation (5.11). An error during this transformation can cause a whole block in one of the cat states in the state (5.13) to be affected, together with one qubit in each one of the last three blocks. However, one such block can ruin the parity bits of only one encoded cat state. As long as the number of faulty parity bits is less than $r/2$, the majority vote will still give the correct parity. An error in the parity computations or in the majority vote cannot affect more than one qubit in $|A_0\rangle$, since they are done transversally.

We choose r to be $2k_0+1$, where $k_0 = \lfloor q/2 \rfloor$. The reason is that, in our proof, we will distinguish between cases with at most k_0 faults in a rectangle, which we treat as the good case, and other cases which will be shown to be rare. Hence, we would like to be able to tolerate $\lfloor q/2 \rfloor$ faults in a rectangle. Choosing $r = 2k_0 + 1$ will do the trick.

5.4.2. Construction of Toffoli gate given $|A_0\rangle$. The construction of the Toffoli gate given $|A\rangle$ follows Shor's scheme almost exactly, with minor changes due to the fact that measurements are replaced by CNOT gates to additional blank qubits. Also, classical conditioning on the results of the measurements is replaced by unitary gates on the computation qubits and extra blank qubits carrying the results of the measurements. Here is how this is done. We will first generate "half" a Toffoli gate:

$$|S_a\rangle|S_b\rangle|S_0\rangle \longmapsto |S_a\rangle|S_b\rangle|S_{ab}\rangle. \tag{5.18}$$

A Toffoli gate can be generated from transformation (5.18) in the following way. We start with the three blocks on which we want to apply Toffoli: $|S_a\rangle|S_b\rangle|S_c\rangle$. Then we generate $|S_0\rangle$ on an extra block, by using the zero-state preparation procedure. We then apply transformation (5.18) on the first two blocks and the newly generated $|S_0\rangle$. Then we apply an encoded CNOT from our original third block $|S_c\rangle$ to the new block. We finally apply an encoded Hadamard on the original third block. This gives the overall transformation:

$$|S_a\rangle|S_b\rangle|S_c\rangle|S_0\rangle \longmapsto \frac{1}{\sqrt{2}}|S_a S_b S_{c+ab}\rangle(|S_0\rangle + (-1)^c|S_1\rangle). \tag{5.19}$$

Note that if the fourth block was not there, we would be done, because the operation on the first three blocks is exactly the Toffoli gate. We next decode the fourth block. Hence, we would like to apply the operation

$$|S_a\rangle|S_b\rangle|S_c\rangle \longmapsto (-1)^{ab+c}|S_a\rangle|S_b\rangle|S_c\rangle \tag{5.20}$$

on the first three blocks, conditioned that the decoded qubits are 1.

This can be applied transversally, in the following way. First, apply a controlled phase on $|S_c\rangle$ and the decoded qubits, in a transversal manner. This will give the factor $(-1)^c$. To apply $|S_a\rangle|S_b\rangle \longmapsto (-1)^{ab}|S_a\rangle|S_b\rangle$ conditioned on the decoded qubits, apply transversally the operation: $|a\rangle|b\rangle|d\rangle \longmapsto (-1)^{abd}|a\rangle|b\rangle|d\rangle$, where $|d\rangle$ is a decoded qubit. This achieves the correct transformation since we know that the controlled phase can be applied transversally. This operation can be applied by adding a blank qubit $|a\rangle|b\rangle|0\rangle|d\rangle$ and applying a Toffoli gate on the first three qubits, followed by a controlled phase on the last two qubits, and then by a Toffoli gate again on the first three qubits.

It is left to show how to construct transformation (5.18) on two blocks B_1 and B_2. This is done by generating the ancilla state $|A_0\rangle$ as before. Now apply an encoded

CNOT from the first block of $|A_0\rangle$ to the first block B_1 and from the second block of $|A\rangle$ to the second block B_2. This achieves the transformation

$$|S_c\rangle|S_d\rangle|A_0\rangle \longmapsto \sum_{a,b} |S_{a+c}\rangle|S_{b+d}\rangle|S_a\rangle|S_b\rangle|S_{ab}\rangle. \tag{5.21}$$

We note that the last three blocks have a strong connection to the Toffoli gate. More precisely, we note that projecting the above sum on the subspace where the first two blocks are $|S_0\rangle|S_0\rangle$ implies that $a = c, b = d$, and the gate which is achieved on the last three blocks is exactly the desired Toffoli gate. If the first two blocks are the state $|S_0\rangle|S_1\rangle$, this implies that $a = c, b = \neg d$, and so the gate which is achieved is the encoded version of $|c, d\rangle \mapsto |c, \neg d, c\neg d\rangle$. Similarly for the two other possibilities we get the encoded versions of $|c, d\rangle \mapsto |\neg c, d, \neg cd\rangle$ and $|c, d\rangle \mapsto |\neg c, \neg d, \neg c\neg d\rangle$, respectively.

To adjust for these deviations from the Toffoli gate, we apply decoding procedures to the first two blocks and then apply the following corrections transversally on the five blocks:

$$\begin{aligned}
&|0,0\rangle|a,b,c\rangle \longmapsto |0,0\rangle|a,b,c\rangle,\\
&|0,1\rangle|a,b,c\rangle \longmapsto |0,1\rangle\mathsf{NOT}(2)\mathsf{CNOT}(1,3)|a,b,c\rangle,\\
&|1,0\rangle|a,b,c\rangle \longmapsto |1,0\rangle\mathsf{NOT}(1)\mathsf{CNOT}(2,3)|a,b,c\rangle,\\
&|1,1\rangle|a,b,c\rangle \longmapsto |1,1\rangle\mathsf{NOT}(1)\mathsf{NOT}(2)\mathsf{NOT}(3)\mathsf{CNOT}(1,3)\mathsf{CNOT}(2,3)|a,b,c\rangle.
\end{aligned} \tag{5.22}$$

It is easy to check that this achieves the desired corrections. The transformation in (5.22) is a reversible transformation on five qubits and can therefore be constructed by a constant number of classical gates from the set $\mathcal{G}_2$.

The spread of this procedure is 1, since we use the decoding procedure which has spread 1 and everything else is done transversally.

6. Universality of the sets of gates $\mathcal{G}_1$ and $\mathcal{G}_2$. In this section we prove the universality of the sets of gates we use. Roughly, a set of gates is said to be universal if the subgroup generated by the set of gates is dense in the group of unitary operations on n qubits $U(2^n)$ (for p-qudits the group is $U(p^n)$). A beautiful theorem by Kitaev [50, 52] and Solovay and Yao [85] implies that if a set $\mathcal{G}$ is universal, then any quantum circuit that uses arbitrary gates with constant fan-in can be replaced by one that uses only gates from $\mathcal{G}$, such that the overhead in time and space of the new circuit is only polylogarithmic. This theorem gives meaning to the notion of universality, since it implies that restricting the computation to a universal gate set implies only polylogarithmic overhead, and thus one can regard such a set as sufficient for quantum computation.

Our proof of universality of the set $\mathcal{G}_2$, used for CSS codes, is a simple reduction to a set of gates shown to be universal by Kitaev [50]. A similar result was achieved independently by Boykin et al. [26]. The proof that the set of gates $\mathcal{G}_1$ used for polynomial codes is universal is much more complicated. It is based on geometrical arguments on the special unitary group $SU(n)$, together with some basic facts from the theory of finite fields.

We start with a detailed discussion of the notion of universality. We then proceed to prove some geometrical lemmas that we will use in our universality proofs, and, finally, we prove the universality of the gate sets $\mathcal{G}_1$ and $\mathcal{G}_2$.

6.1. Universal sets of gates—basic results.

DEFINITION 19. *Let $p \geq 2$. A set of gates $\mathcal{G}$ on $k \geq 2$ p-qudits is said to be universal if it is closed under inversion and $\mathcal{G} \cup \{e^{i2\pi\theta} I\}_{real\ \theta}$ generates a dense subset in $U(p^k)$.*

The inclusion of the scalars of the form $e^{2\pi i\theta}$ (which are of absolute value 1 and are sometimes called *phases*) does not have any physical effect, since multiplying a gate by such a phase does not change the resulting density matrix.

The fact that this definition indeed captures the notion of universality is summarized by the following theorem.

THEOREM 7. *Consider $\epsilon > 0$, a quantum circuit Q that uses arbitrary gates (of constant fan-in), and a universal set of gates $\mathcal{G}$. The circuit Q can be translated to a circuit Q_ϵ that uses only gates from $\mathcal{G}$ such that the following three conditions hold:*

1. *Q_ϵ computes a function that approximates the function computed by Q to within total variation distance ϵ.*
2. *Q_ϵ is only polylogarithmically larger and deeper than Q.*
3. *The description of Q_ϵ can be efficiently computed given the description of Q.*

The proof of this theorem is well known for the case of qubits, and we extend it here for the case of p-qudits. The proof is based on two results. The first result, known as the Solovay–Kitaev theorem, states that "density implies efficiency": If a set operating on some Hilbert space is universal, then it can be used to approximate any unitary matrix on the same space exponentially rapidly. Moreover, the sequence of gates from $\mathcal{G}$ that achieves the approximation can be found efficiently.

THEOREM 8 (see Kitaev [50, 52] and Solovay [85]). *Let $\mathcal{G}$ be a universal set of gates over $U(p^k)$ for some integers $p, k \geq 2$. Then there exists a Turing machine A that, given a matrix $M \in U(p^k)$ and $\epsilon > 0$, outputs a sequence of gates $g_1, \ldots, g_\ell \in \mathcal{G}$ such that $\|M - g_\ell \cdot g_{\ell-1} \ldots g_1\| < \epsilon$, and both ℓ and the running time of A are polynomial in $\log(1/\epsilon)$.*

The proof of this beautiful and fundamental theorem uses Lie groups and Lie algebras and is beyond the scope of this paper.

To complete the proof of Theorem 7 we need another fact: An operation on any number of qubits (qudits) can be achieved by using gates that operate on only two qubits (qudits). This was proved for the case of qubits by DiVincenzo [40], a proof which was simplified by Barenco et al. [19]. We give here a proof for the general case of qudits, by using similar ideas to those used by Deutsch [39] and Barenco et al. [19].

THEOREM 9. *Let $\mathcal{G}$ be a universal set of gates on $k \geq 2$ p-qudits. Consider the Hilbert space of $m > k$ p-qudits. Then the set of gates achieved by extending the gates in $\mathcal{G}$ to operate on m p-qudits generates a dense subset of $U(p^m)$.*

Proof. We define a generalized Toffoli gate on m qudits $T_m(Q)$ to be a gate which applies Q on the mth qudit conditioned that the first $m-1$ qudits are in the state $p-1$ (see Figure 6.1).

The conditioned Q can be applied on the rth qudit, instead of the mth one, in which case we denote the gate by $T_{m,r}(Q)$.

CLAIM 5. *The set of gates $T_{m,r}(Q)$ can be approximated to within an arbitrary accuracy by using the extensions of gates from G to m qudits.*

Proof. We first show in Figure 6.2 an explicit sequence of generalized Toffoli gates on $m-1$ qudits, $T_{m-1}(Q)$'s, which achieves $T_m(Q)$.

We denote that $V = Q^{\frac{1}{p}}$, and $\oplus$ denotes here the $\mathsf{NOT}_p = \mathsf{NOT}_p(1)$ operation. It is easy to check that the above circuit indeed gives the desired controlled Q, by considering what happens to the basis states, in two cases: All first $m-2$ qudits

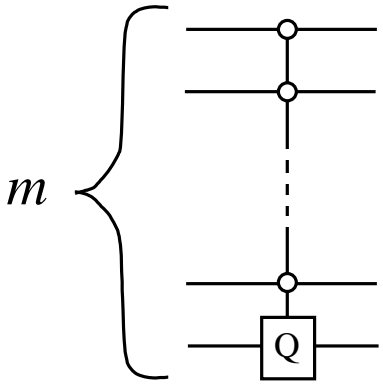


FIG. 6.1. *The controlled operation in F_2 is replaced here by conditioning the application of Q on the fact that the first $m-1$ qudits are in the state $p-1$. Note the difference from the usual conditioning, which for the case of $m=2$ would give CNOT_p. To distinguish it from the standard kind of conditioning, we denote this controlling operation by an empty circle.*

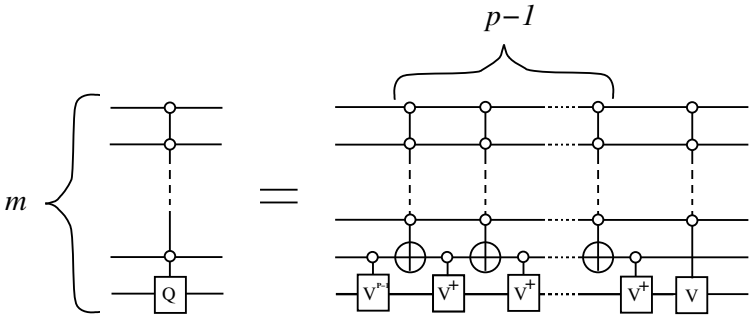


FIG. 6.2. *The gate $T_m(Q)$ can be written as a sequence of gates of the form $T_{m-1}(Q)$. After the first controlled V^{p-1}, we apply $T_{m-1}(\mathsf{NOT}_p)$, followed by a controlled $V^\dagger$, and repeat this pair of gates $p-1$ times. At the end we apply V controlled on the first $p-2$ qudits.*

are equal to $p-1$, or not. By using the above scheme recursively we can construct a circuit which uses k-qudit gates of the form of $T_k(Q)$ and achieves $T_m(Q)$ for any $m > k$ and any one-qudit Q. Since the set $\mathcal{G}$ is universal, the gates of the form $T_k(Q)$ can be approximated. Note that the recursion starts with two-qudit gates, which is the reason why we require $k \geq 2$. The construction is similar for $T_{m,r}(Q)$. □

The gate $T_m(Q)$ can be seen as applying Q on the subspace spanned by the last p basis vectors while applying identity on the rest. The next step is to construct a generalization of the above gate, i.e., a gate which applies $Q \in U(p)$ on the subspace spanned by any p basis vectors $|i_1\rangle, \ldots, |i_p\rangle$ while applying identity on the rest of the basis vectors. Denote this matrix by $T_m(Q, i_1, \ldots, i_p)$. We note that since $\mathcal{G}$ is universal, it can be used to approximate any one-qudit gate, so we are also allowed to use in our construction NOT_p gates.

CLAIM 6. *The gate $T_m(Q, i_1, \ldots, i_p)$ can be constructed by using gates of the form $T_{m,r}(Q)$ together with NOT_p gates.*

Proof. Consider the set of gates $\{T_{m,r}(Q)\}$, where Q runs over all permutation matrices on the Hilbert space of one qudit and r runs over all m qudits. We first prove that this set, augmented with NOT_p gates, generates all permutation matrices

on m qudits. To prove this, it suffices to construct all matrices of the form $\tau_{i,j}$ for two strings $i, j \in \{0, p-1\}^m$ that differ in only one coordinate. The matrix $\tau_{i,j}$ is defined to be the transformation that switches the two basis vectors $|i\rangle, |j\rangle$ and $\tau_{i,j}|k\rangle = |k\rangle$ for any $k \neq i, j$. Let the coordinate on which i, j disagree be the rth coordinate, and let this coordinate be equal to a in the string i and to b in the string j. To construct $\tau_{i,j}$, apply NOT_p gates on all of the coordinates except the rth coordinate, so that all of these coordinates in both strings become equal to $p-1$. Now apply $T_{m,r}(Q)$, with Q on the rth coordinate being the matrix which permutes $|a\rangle$ and $|b\rangle$, leaving the rest of the basis vectors untouched. Then reverse all of the NOT_p gates on all of the coordinates but the rth one. This gives $\tau_{i,j}$ and therefore all permutations on basis vectors. The general $T_m(Q, i_1, \ldots, i_p)$ for any Q can now be achieved by first permuting the basis vectors $i_1, \ldots, i_p$ to the last p vectors, applying $T_m(Q)$, and then permuting back. ∎

The last step is to use $T_m(Q, i_1, \ldots, i_p)$ to construct a general $p^m \times p^m$ unitary matrix U. Let us denote the p^m eigenvectors of U by $|\psi_j\rangle$ with corresponding eigenvalues $e^{i\theta_j}$. U is specified by $U|\psi_j\rangle = e^{i\theta_j}|\psi_j\rangle$. Define

$$U_k|\psi_j\rangle = \begin{cases} |\psi_j\rangle & \text{if } k \neq j, \\ e^{i\theta_k}|\psi_k\rangle & \text{if } k = j. \end{cases} \tag{6.1}$$

Then $U = \Pi_{k=1}^{p^m} U_k$. It is left to show how to construct U_k. For this we will show how to construct a transformation R with the following properties: R takes $|\psi_k\rangle$ to $\lambda|(p-1)^m\rangle$ for some complex number λ of absolute value 1. We don't care what R does to the rest of the vectors. Given such an R, we can use it to construct U_k as follows. We first apply R. We then apply the generalized Toffoli gate which takes $|(p-1)^m\rangle$ to $e^{i\theta_k}|(p-1)^m\rangle$ and does nothing to the rest of the basis states. Then we apply R^{-1} which takes $\lambda|(p-1)^m\rangle$ to $|\psi_k\rangle$. This indeed achieves U_k, as can be easily checked.

To construct R, and similarly R^{-1}, we do the following. We start with $|\psi_k\rangle$, and we first make the coefficient in front of $|0^m\rangle$ zero, by a rotation in the plane spanned by $|0^m\rangle$ and $|(p-1)^m\rangle$. This is a special case of the $T_m(Q, i_1, \ldots, i_p)$ which we have constructed before. Thus, the weight of $|0^m\rangle$ has been shifted to $|(p-1)^m\rangle$. In the same way, the weights in front of all basis vectors, one by one, can be shifted to $|(p-1)^m\rangle$, and this achieves R. ∎

6.2. Useful lemmas for proving universality. We proceed to state two geometrical lemmas. These will be used in the proofs of universality of the sets of gates $\mathcal{G}_1$ and $\mathcal{G}_2$. These lemmas are due to Kitaev, who used them for proving universality of the set of gates in [50].

LEMMA 2. *Let $n \geq 3$. Let $|\alpha\rangle \in \mathcal{C}^n$. Let H be the subgroup in $SU(n)$ which fixes $|\alpha\rangle$, and let $V \in U(n)$ be a matrix which does not leave the subspace spanned by $|\alpha\rangle$ invariant. Then the subgroup generated by the subgroups H and $V^{-1}HV$ is dense in $SU(n)$.*

The proof of this lemma can be found in the solution of Problem 8.11 in [52].

LEMMA 3. *Let U_1 and U_2 be two noncommuting matrices in $SU(2)$ such that their eigenvalues are not integer roots of unity. The subgroup generated by U_1, U_2 is dense in $SU(2)$.*

Proof. If x is an element of $SU(2)$ not of finite order, then the closed subgroup generated by x is connected and of dimension 1. The closed group generated by both U_1 and U_2 is thus connected and is noncommutative. Any connected noncommutative subgroup of $SU(2)$ is *all* of $SU(2)$. ∎

6.3. Universality of the set of gates $\mathcal{G}_1$ for polynomial codes. We would now like to show that the set of gates for polynomial codes $\mathcal{G}_1$ is universal.

THEOREM 10. *Consider p-qudits for a prime $p > 3$. The set of gates $\mathcal{G}_1$ together with all phase factors is universal for $U(p^5)$.*

Let us first show that we have at our disposal all classical gates on two qudits. Let F be a finite field of characteristic p for some prime p and size $q = p^r$ for some r. Consider the basic classical reversible gates on F-valued registers, including addition and multiplication by a nonzero constant, reversible addition, $[a, b] \to [a, a + b]$, and Toffoli $[a, b, c] \to [a, b, c + ab]$. For a characteristic different than 2, we also consider the second basic set, which is the above set of gates but with the Toffoli gate replaced by the squaring gate: $[a, b] \to [a, b + a^2/2]$. We claim the following.

LEMMA 4 (folklore). *Both sets of basic reversible gates, applied on $k \geq 3$ qudits, generate the full permutation group S_{q^k}. In both cases an ancillary register in the state 0 is added to the system.*

Proof. We start with the first set of basic gates. Denote by G_k the group generated by the basic gates on k registers. For a function $g : F^{k-1} \to F$ we define the permutation $\pi_g : F^k \to F^k$ by

$$\pi_g([a_1, \ldots, a_k]) = [a_1 + g(a_2, \ldots, a_k), a_2, \ldots, a_k].$$

The coordinate to which g is added is called the target coordinate. The main result of Ben-Or and Cleve [23] is that, for $k \geq 3$ and any such function g, $\pi_g \in G_k$.

We now use functions of the form π_g to generate all of S_{q^k}. First, recall that any permutation on S_{q^k} can be written as a product of transpositions of the form $\tau_{i,j}$ for $i, j \in F^k$ such that the string i differs from the string j in only one coordinate ($\tau_{i,j}$ leaves all other coordinates unchanged). It thus suffices to generate $\tau_{i,j}$ from functions of the form π_g. To do this, we add one ancillary dit in the state 0. Let us assume that the coordinate in which the strings i, j differ is the rth one. Define $g : F^k \to F$ to be the function that takes a string s to the value of the rth coordinate in $\tau_{i,j}(s)$. We have that $\pi_g([a_0, a_1, \ldots, a_k]) = [a_0 + g(a_1, \ldots, a_k), a_1, \ldots, a_k]$, which operates on an ancillary coordinate (a_0) plus the k original coordinates, is at our disposal.

To achieve $\tau_{i,j}$, we apply π_g. This is followed by π'_g, which we define to be $\pi'_g([a_0, a_1, \ldots, a_k]) = [a_0, a_1, a_{r-1}, a_r - g(a_1, \ldots, a_{r-1}, a_0, a_{r+1}, \ldots, a_k), a_{r+1}, \ldots, a_k]$. We note that

$$[a_1, \ldots, a_k] \to [a_1 - g(a_2, \ldots, a_k), a_2, \ldots, a_k]$$

is in G_k, by applying π_g $q - 1$ times. Hence, by renaming coordinates, π'_g is also at our disposal. By the definition of g we have that $\pi'_g \pi_g [0, a_1, \ldots, a_k] = [g[a_1, \ldots, a_k], a_1, \ldots, a_{r-1}, 0, a_{r+1}, \ldots, a_k]$ which is the desired transposition up to swapping the coordinates.

The proof for the second set of basic reversible gates is similar by using the fact that in this case $\pi_g \in G_k$ for any $k \geq 2$, which was also proved in [23]. ∎

Remark 2. In fact, the same result can be proved without the ancillary qudit, by using results from Coppersmith and Grossman [35]. We do not give details here.

Since $\mathcal{G}_1$ contains all basic gates, this implies that we have at our disposal all classical gates on three (and therefore also on two) qudits. Next, we prove an analogue for F_p of the well-known fact that one-qubit gates and classical two-qubit gates are universal for qubits [19].

LEMMA 5. *The group G generated by the set of gates consisting of all one-qudit gates $SU(C^p)$ and all classical two-qudit gates is the special unitary group on two qudits $SU(C^{p^2})$.*

Proof. We will use the fact that the lemma we are trying to prove is already known for the case of $p = 2$. Let S be the two-dimensional subspace in C^p spanned by the first two basis vectors $|0\rangle$ and $|1\rangle$. Clearly, $SU(S)$ is in $U(C^p)$. Let S_ℓ be the two-dimensional subspace in the Hilbert space of the ℓth qudit, for $\ell \in \{0, 1\}$, and let $A = S_1 \otimes S_2$. Since we have at our disposal all classical gates, we also have all permutations of basis vectors of A. We can use the fact that A is isomorphic to the Hilbert space of two qubits, and Lemma 5 for the case of qubits, and so this implies that $SU(A)$ can be generated. We now augment A by one computational basis state at a time to get the entire space.

Consider $|i_1\rangle, \ldots, |i_{p^2-2}\rangle$ to be a sequence of computational basis vectors which completes the basis of A, $|00\rangle, |10\rangle, |01\rangle, |11\rangle$, to a basis for C^{p^2}. Define $A_j = A \oplus |i_1\rangle \oplus \cdots \oplus |i_j\rangle$. We prove by induction on j that we can generate $SU(A_j)$. Suppose that we can generate $SU(A_{j-1})$. The classical gate which permutes $|i_j\rangle$ and $|i_{j-1}\rangle$, leaving all other states unchanged, is at our disposal. This gate is in $SU(A_j)$ and does not leave the subspace spanned by $|i_j\rangle$ invariant. Hence, we can apply Lemma 2 to get all of $SU(A_j)$. ∎

Lemma 5 implies that it would suffice to show that all one-qudit gates are at our disposal. Denote by Q the one-qudit matrix of the form

$$Q = \begin{pmatrix} w & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix}, \tag{6.2}$$

which applies a phase factor of w on the state $|0\rangle$.

LEMMA 6. *Q and Q^{-1} are in the subgroup generated by $\mathcal{G}_1$ on three qudits.*

Proof. We will generate $Q \otimes I \otimes I$, which applies the following transformation:

$$\begin{aligned} |0\rangle|a\rangle|b\rangle &\longmapsto w|0\rangle|a\rangle|b\rangle, \\ |j\rangle|a\rangle|b\rangle &\longmapsto \quad |j\rangle|a\rangle|b\rangle, \qquad j \neq 0. \end{aligned} \tag{6.3}$$

To achieve this transformation, we view this gate as applying multiplication by w of the second qudit, conditioned that the first qudit is 0. Recall that in our notation P is a one-qudit gate that applies a phase shift $P|a\rangle = w^a|a\rangle$, and B is a one-qudit gate that applies a bit shift $B|a\rangle = |a + 1 \bmod \mathrm{p}\rangle$. Both B and P are in our repertoire. Denote by $\Lambda(B)$ the controlled B, which applies B on the second qudit, conditioned that the first qudit is 0, and applies the identity on the second qudit if the state of the first qudit is anything but 0. $\Lambda(B)$ is also at our disposal by Lemma 4. Now consider the commutator

$$P^{-1} \cdot \Lambda(B)^{-1} \cdot P \cdot \Lambda(B), \tag{6.4}$$

where the P's are applied to the second qudit. This is exactly the gate we want, since, if the first qudit is 0, the matrix which is applied on the second qudit is $P^{-1} \cdot B^{-1} \cdot P \cdot B = wI$. If the first qudit is in a basic state which is not $|0\rangle$, the matrix which is applied on the second qudit is the identity. ∎

We now consider the two commutator matrices in $\langle \mathcal{G}_1 \rangle$, the group generated by $\mathcal{G}_1$:

$$X = HQH^{-1}Q^{-1}, \quad Y = HQ^{-1}H^{-1}Q, \tag{6.5}$$

where H is the generalized Fourier transform. We also define the two-dimensional subspace S:

$$S = span\left\{|0\rangle, \sum_{b\in F_p, b\neq 0} |b\rangle\right\}. \tag{6.6}$$

The claim is that X and Y operate as the identity on the orthogonal subspace to S.

LEMMA 7. *X and Y operate as the identity on $S_i^{\perp}$.*

Proof. It is easy to write down explicitly the matrix elements of $X = HQH^{-1}Q^{-1}$ and $Y = HQ^{-1}H^{-1}Q$:

$$X_{ab} = \begin{cases} \delta_{ab} + \frac{1}{p}(w-1) & \text{if } b \neq 0, \\ (\delta_{ab} + \frac{1}{p}(w-1))w^{-1} & \text{if } b = 0, \end{cases} \tag{6.7}$$

$$Y_{ab} = \begin{cases} \delta_{ab} + \frac{1}{p}(w^{-1}-1) & \text{if } b \neq 0, \\ (\delta_{ab} + \frac{1}{p}(w^{-1}-1))w & \text{if } b = 0. \end{cases}$$

To see that X and Y operate as the identity on the subspace orthogonal to S, consider the matrices $X - I$ and $Y - I$, which satisfy (6.7) if we subtract δ_{ab} from each term. We show that the orthogonal vectors to S are all in the kernel of $X - I$ and $Y - I$. A vector v orthogonal to S satisfies

$$v_0 = 0, \quad \sum_{b,b\neq 0} v_b = 0, \tag{6.8}$$

and thus

$$\sum_{b\in F_p} (X-I)_{ab} v_b = \frac{1}{p}(w-1)\sum_{b\neq 0} v_b = 0, \tag{6.9}$$

$$\sum_{b\in F_p} (Y-I)_{ab} v_b = \frac{1}{p}(w^{-1}-1)\sum_{b\neq 0} v_b = 0. \qquad \square$$

We denote by X' and Y' the two matrices confined to S. We claim that these matrices generate a dense subgroup in the group of 2×2 unitary matrices $U(2)$ operating on S. We will want to use Lemma 3, and the main effort is to prove that the eigenvalues of X' and Y' are not integer roots of unity. The proof of this fact is based on some basic results regarding cyclotomic fields and Galois fields, which can be found in [98].

LEMMA 8. *For $p > 3$, the eigenvalues of X' and Y' are not integer roots of unity.*

Proof. The subspace S is spanned by the orthonormal basis vectors $|0\rangle$ and $|\alpha\rangle = \frac{1}{\sqrt{p-1}} \sum_{b\in F_p, b\neq 0} |b\rangle$. By (6.7) and a little algebra we get

$$X'|0\rangle = \left(1 + \frac{1}{p}(w-1)\right)w^{-1}|0\rangle + \frac{\sqrt{p-1}}{p}(w-1)|\alpha\rangle, \tag{6.10}$$

$$X'|\alpha\rangle = \frac{\sqrt{p-1}}{p}(w-1)w^{-1}|0\rangle + \left(1 + \frac{p-1}{p}(w-1)\right)|\alpha\rangle,$$

and for Y we get the transformation

$$Y'|0\rangle = \left(1 + \frac{1}{p}(w^{-1}-1)\right)w|0\rangle + \frac{\sqrt{p-1}}{p}(w^{-1}-1)|\alpha\rangle, \tag{6.11}$$

$$Y'|\alpha\rangle = \frac{\sqrt{p-1}}{p}(w^{-1}-1)w|0\rangle + \left(1 + \frac{p-1}{p}(w^{-1}-1)\right)|\alpha\rangle.$$

It is easy to verify that X', and similarly Y', have determinant 1. The characteristic polynomial for X' is $\lambda^2 - \mathrm{Tr}(X')\lambda + \mathrm{Det}(X')$, which amounts to

$$f(\lambda) = \lambda^2 - \frac{2 + (p-1)(w + w^{-1})}{p}\lambda + 1. \tag{6.12}$$

Y' has exactly the same characteristic polynomial. We want to show that the roots of this polynomial are not integer roots of unity. Let us assume that one of the roots of the above polynomial is a primitive nth root of unity, denoted by ζ_n, for some integer n. The other solution is the complex conjugate of ζ_n, and we have

$$\frac{2 + (p-1)(w + w^{-1})}{p} = \zeta_n + \zeta_n^{-1}. \tag{6.13}$$

We will first prove that $n = p$ or $2p$. Denote by $Q(w)$ and $Q(\zeta_n)$ the Galois extensions of the field of rationals obtained by adjoining w and ζ_n, respectively, to the field of rationals Q. Also, denote by $Q(w)^+$ the maximal real subfield of $Q(w)$ obtained by extending Q by $w + w^{-1}$, and similarly denote the maximal real subfield of $Q(\zeta_n)$ by $Q(\zeta_n)^+$. By (6.13), $Q(\zeta_n)^+ = Q(w)^+$.

The degree of the extension $\deg(Q(w)/Q(w)^+)$ is exactly 2, since w is a root of the minimal two degree polynomial $x^2 - (w + w^{-1})x + 1$ over the field $Q(w)^+$. Similarly, $\deg(Q(\zeta_n)/Q(\zeta_n)^+) = 2$. On the other hand, $\deg(Q(w)/Q) = p - 1$ and $\deg(Q(\zeta_n)/Q) = \phi(n)$, by Theorem 2.5 in [98]. Now, for three fields, F_1, F_2, and F_3 such that F_3 extends F_2 which extends F_1, we have $\deg(F_3/F_1) = \deg(F_3/F_2)\deg(F_2/F_1)$. It follows that

$$\deg(Q(w)^+/Q) = \frac{p-1}{2}, \quad \deg(Q(\zeta_n)^+/Q) = \frac{\phi(n)}{2}. \tag{6.14}$$

But $Q(w)^+ = Q(\zeta_n)^+$, which implies that the degrees of extensions are equal, so $\phi(n) = p - 1$.

Now if $p > 3$, then $w + w^{-1} \notin Q$, since $\deg(Q(w + w^{-1})/Q) = (p-1)/2 > 1$. Since $w + w^{-1} \in Q(w) \cap Q(\zeta_n)$, we have $Q \neq Q(w) \cap Q(\zeta_n)$. If p and n were relatively prime, we would have $Q = Q(w) \cap Q(\zeta_n)$ (by Proposition 2.4 in [98]), and so p must divide n, say, $n = p^r m$, with m coprime to p. This implies that

$$\phi(n) = p^{r-1}(p-1)\phi(m). \tag{6.15}$$

Since, as we have seen before, $\phi(n) = p - 1$, we have $p^{r-1}\phi(m) = 1$. We deduce that $r = 1$ and $\phi(m) = 1$. This can be satisfied only if $m = 1$ or $m = 2$, namely, $n = p$ or $n = 2p$.

In the first case of $p = n$ we get

$$\frac{1 + (p-1)\cos(2\pi/p)}{p} = \cos(2k\pi/p), \tag{6.16}$$

where we have set $\xi_n = e^{2\pi i k/n}$. If $p > 3$, (6.16) is not satisfied by any integer k, since the left-hand side is a convex combination of $\cos(2\pi/p)$ and 1, and no real part of a pth root of unity lies between these two points.

In the second case of $p = 2n$, we get

$$\frac{1 + (p-1)\cos(2\pi/p)}{p} = \cos(k\pi/p). \tag{6.17}$$

Ruling this out is similar to ruling out (6.16), except that we have to show that the case of $k = 1$ does not hold, namely, $\frac{1+(p-1)\cos(2\pi/p)}{p} \neq \cos(\pi/p)$. This is true because the cosine function is a concave function. The lemma follows. ∎

We can now show that we have at our disposal all one-qudit gates.

CLAIM 7. *For $p > 3$, the set $\mathcal{G}_1$ operating on three qudits generates all one-qudit gates.*

Proof. Since we are allowed to operate on three qudits, we can generate Q by Lemma 6. We can thus generate X and Y. It is easy to see that, if $w \neq \pm 1$ or, equivalently, $p \neq 2$, the off-diagonal terms of $X'Y' - Y'X'$ are not zero, and so X' and Y' do not commute. We can therefore apply Lemma 3, by using the fact that the eigenvalues are not roots of unity, by Lemma 8. We thus have a dense subset of the special unitary group $SU(2)$ on all subspaces S. We now want to augment the subspace S by one basis vector at a time, to get the entire $SU(p)$ group. To do this, we define $S_0 = S$ and, for $p > j \geq 1$, $S_j = S \oplus |1\rangle \oplus \cdots \oplus |j\rangle$. We have seen that we can generate $SU(S_0)$. Assume by induction that we can generate $SU(S_{j-1})$, and let us see that we can generate $SU(S_j)$. Let $|\gamma\rangle$ be the state orthogonal to S_{j-1} inside S_j. If we set $|\delta\rangle = \ sum_{k=j+1}^{p-1}|k\rangle$, then it is easy to check that $|\gamma\rangle$ is a nontrivial combination of $|\delta\rangle$ and $|j\rangle$. We have at our disposal a classical gate which exchanges $|0\rangle$ and $|j\rangle$. Moreover, this gate is in $SU(S_j)$. This gate does not leave the subspace spanned by $|\gamma\rangle$ invariant. Hence we can apply Lemma 2 and get all of $SU(S_j)$. Since $SU(S_{p-1}) = SU(p)$, the claim is proved. ∎

The proof of Theorem 10 now follows easily.

Proof of Theorem 10. We have shown that we have at our disposal all one-qudit gates. Lemma 4 shows that, by working on five qudits, we also have in our repertoire all classical gates on three qudits and, in particular, all classical gates on two qudits which act trivially on the third qudit. Lemma 5 implies that we can generate all two-qudit gates. The theorem follows from Theorem 9, which shows that these matrices can be used to construct all matrices on five qudits $U(p^5)$. ∎

6.4. Universality of the set of gates $\mathcal{G}_2$ used for CSS codes. The set of gates used for CSS codes is shown here to be universal. The theorem is based on an argument by Kitaev [50] which is given here for completeness.

THEOREM 11. *The set of gates $\mathcal{G}_2$ together with all phase factors is universal for $U(2^5)$.*

Proof. We denote by P the gate that takes $|1\rangle$ to $i|1\rangle$ and does nothing to $|0\rangle$. $\Lambda(P)$ is the controlled P, namely, the gate which applies P on a second qubit only if the first qubit is $|1\rangle$ and does nothing otherwise. The proof is based on a result by Kitaev [50], which asserts that the set of gates $\{\Lambda(P), H\}$ is universal. Since the Hadamard gate H is already in our set $\mathcal{G}_\in$, we need only to show how to construct $\Lambda(P)$ from our set of gates. We will denote by $T_{a_1,\ldots,a_k}$ a generalized Toffoli on the k qubits $a_1, \ldots, a_k$. This gate applies NOT on the kth qubit conditioned that the first $k-1$ qubits are in the state $|1\rangle$. We first construct $T_{1,2,4,5}$ out of five qubits. This can be done by using three-qubit Toffoli gates as follows:

$$
\begin{aligned}
a, b, c, d, e \quad &\overset{T_{1,2,3}}{\longmapsto} \\
a, b, c + ab, d, e \quad &\overset{T_{3,4,5}}{\longmapsto} \\
a, b, c + ab, d, e + cd + abd \quad &\overset{T_{1,2,3}}{\longmapsto} \\
a, b, c, d, e + cd + abd \quad &\overset{T_{3,4,5}}{\longmapsto} \\
a, b, c, d, e + abd,
\end{aligned}
\tag{6.18}
$$

which is exactly $T_{1,2,4,5}$. Define $X = P_4^3 T_{1,2,3,4} P_4 T_{1,2,3,4}$, where P_4 applies the gate P to the fourth qubit. X takes $|1110\rangle \longmapsto i|1110\rangle$, $|1111\rangle \longmapsto -i|1111\rangle$ and does nothing

to the other basis states. X^2 is the three-qubit gate which gives $|111\rangle \longmapsto -|111\rangle$ and identity on the rest of the basis vectors, tensored with identity on the fourth qubit. Now, to construct $\Lambda(P)$, we apply $P_3^3 T_{1,2,3} P_3 T_{1,2,3}$ followed by X^2. The first sequence of gates gives $|110\rangle \longmapsto i|110\rangle, |111\rangle \longmapsto -i|111\rangle$ and identity on the rest. By applying X^2 we get $\Lambda(P)$ tensored with identity on the third qubit. The theorem now follows from Kitaev [50].

We now provide Kitaev's argument which uses Lemmas 2 and 3 to show that $\Lambda(P)$ and the Hadamard gate are universal. Denote that

$$X_1 = H_1 \Lambda(P)_{1,2} H_1, \quad X_2 = H_2 \Lambda(P)_{2,1}^{-1} H_2. \tag{6.19}$$

Define $Y_1 = X_1 X_2^{-1}$ and $Y_2 = X_2 X_1^{-1}$. Note that Y_1 and Y_2 both operate as the identity on the two states $|00\rangle$ and $|\eta\rangle = |01\rangle + |10\rangle + |11\rangle$. Denote by L the subspace orthogonal to $|00\rangle$ and $|\eta\rangle$. Then $Y_1\ Y_2 \in SU(L)$. Y_1, and Y_2 do not commute, and their eigenvalues are $\frac{1}{4}(1 \pm \sqrt{15})$. Hence, by Lemma 3 they generate a dense subgroup in $SU(L)$. Now add to Y_1, Y_2 also the gate $\Lambda(P)$ itself. This gate fixes $|00\rangle$ but does not stabilize the space $|\eta\rangle$. We can use Lemma 2 to show that the set $\{Y_1, Y_2, \Lambda(P)\}$ generates a dense subgroup in $SU(L \oplus |\eta\rangle)$. Finally, add H_1 to the set $Y_1, Y_2, \Lambda(P)$. We have seen that $Y_1, Y_2, \Lambda(P)$ generates a dense set in the subgroup that fixes $|00\rangle$, while H_1 is not in this subgroup. Hence, $H_1, Y_1, Y_2, \Lambda(P)$ (which are all gates generated by H and $\Lambda(P)$) generate a dense subgroup of $SU(L \oplus |\eta\rangle \oplus |00\rangle) = SU(4)$. Together with all phase factors, we get the unitary group on two qubits. □

7. Fault tolerance for independent probabilistic noise. In this section we prove our main result: the threshold theorem for fault tolerance in the presence of probabilistic noise with a constant error rate. To do this we use the ingredients we have developed in the previous sections: quantum error correcting codes and fault-tolerant procedures on states encoded by these codes. The construction is based on a simulation of an unreliable circuit M_0 by another circuit M_1, which computes the same function. The new circuit M_1 is deeper and requires more space than M_0 but only by a constant factor. The advantage is that, under certain conditions on the error rate, M_1 is more reliable than the original circuit M_0. In such a case, the simulation can be applied recursively, each time achieving further improvement in the effective error rate. The final quantum circuit M_r, which is only polylogarithmically larger and deeper than M_0, can be shown to be fault-tolerant against a constant error rate. To analyze the propagation of errors in M_r, we define the notion of sparse errors and sparse fault paths, by using hierarchical definitions that fit the hierarchical structure of the construction. The threshold theorem is proved in two parts. First, we show that sparse fault paths are good, meaning that they cause sparse errors which do not affect the final result. Second, we show that nonsparse fault paths are rare, as long as the error rate is below a certain threshold.

For this section, we use the terminology of qubits. Everything works in exactly the same way if the particles are qudits instead, as is required in the scheme which uses the polynomial codes. In this case, the final circuit will of course consist of p-qudits, where p is some finite dimension.[5]

[5] One can of course apply our construction, which uses polynomial codes, such that the final circuit is implemented with qubits. For this, we simply replace each p-qudit in the final circuit with $\lceil \log(p) \rceil$ qubits. The proof that the new circuit is fault-tolerant follows similar ideas as in section 9.

7.1. Quantum computation codes. In order to compute on encoded states, we need a quantum code accompanied with a universal set of gates which can be applied fault-tolerantly (namely, with small spread) on states encoded by C. The code should also be accompanied with fault-tolerant decoding, zero-state preparation, and error correction procedures. We define the following.

DEFINITION 20. *A quantum code is called a quantum computation code with spread* 1 *if it is accompanied with a universal set of gates G with fault-tolerant procedures and with fault-tolerant zero-state preparation, decoding, and correction procedures, all with spread* 1. *Moreover, we require that*

(1) *all procedures use only gates from G, and*

(2) *the correction procedure takes any density matrix to some word in the code.*

The first restriction in the definition of a quantum computation code allows us to use this code in a recursive construction. The second restriction is required for a more subtle reason, as will become clear in the proof of Lemma 9.

Theorem 5 implies that the polynomial QECC accompanied by the set of gates $\mathcal{G}_1$ is a quantum computation code with spread 1. Theorem 6 implies the same for CSS codes over F_2 restricted as in section 5.1, with the set of gates $\mathcal{G}_2$.

In our proof of fault tolerance, we will be interested not in the spread of one procedure but in the spread of a *sequence* of two procedures, namely, an encoded gate preceded by an error correction (this will be referred to as a rectangle). For the case of procedures of spread 1, this does not matter, since by augmenting two such procedures one after the other, the resulting circuit consisting of the pair of procedures also has spread 1.

Unfortunately, this does not hold for the case in which the spread of the procedures we augment is larger than 1. In fact, if the spread of the procedures is ℓ', the spread of a circuit consisting of two of them one after the other might be ℓ'^2. We take this into account in our definition of a quantum computation code which uses procedures of spread larger than 1. A quantum computation code with spread ℓ is defined as in Definition 20, except that we require that the spread of the decoding and zero-state preparation procedure, as well as the spread of a sequence of an encoded gate preceded by an error correction procedure, be ℓ.

Obviously, we must require that the number of errors that the code can correct q be larger than the spread of the code, so that we can tolerate at least one fault in a rectangle. In fact, we require more than that for our proof to work: We need $2\ell \leq q$, as will be seen in Lemma 9.

7.2. Recursive simulations. From now on, fix a quantum computation code C. It encodes one qubit on m qubits, it corrects q errors, and it is accompanied with a universal set of gates $\mathcal{G}$ which can be performed fault-tolerantly. We will fix m to be a constant which does not grow with n. Let M_0 be a quantum circuit using gates from $\mathcal{G}$. We simulate M_0 by a more reliable circuit M_1, as follows. Each qubit is replaced by a block of qubits. Each time step in M_0 transforms in M_1 to a working period, which consists of two stages. In the first stage, an error correction procedure is applied on each block. At the second stage, each gate which operated in the simulated time step in M_0 is replaced in M_1 by its procedure, operating on the corresponding blocks.

The input of M_1 is the input to M_0, where each input bit is duplicated m times. Before any computation is done on this input, we apply in M_1 a zero-state preparation procedure, then apply CNOT_p transversally from the given input string $|a^m\rangle$ to the encoded state $|S_0\rangle$, and discard the m initial dits. At the end of the computation we will again use redundancy: For each block, we apply a decoding procedure which

decodes the state to m copies of the logical bit, and the output of M_1 is defined as the majority of the bits in each block.

The above mapping, denoted by $M_1 = \phi(M_0)$, constitutes one level of the simulation. The mapping ϕ is then applied once again, this time on M_1, to give M_2. We repeat this r levels to get $M_r = \phi^r(M_0)$, an r-simulating circuit of M_0. The number of levels r is $O(\text{polyloglog}(V(M_0)))$, where $V(M_0)$ is the volume of M_0, i.e., the number of locations in M_0 (see Definition 5 of the term *location*). The output of M_r is defined by taking recursive majority on the outputs. This means that first we take the majority in each block of size m, then we take the majority, of m such majority bits, and so on for r levels, to give one output bit.

7.3. Blocks and rectangles. The recursive simulations induce a definition of s-blocks: Every qubit transforms to a block of m qubits in the next level, and this block transforms to m blocks of m qubits, and so on. One qubit in M_{r-s} thus transforms to m^s qubits in M_r. This set of qubits in M_r is called an s-block. This induces a division of the qubits in M_r to s-blocks, and this division is a refinement of the division to $s+1$-blocks. A 0-block in M_r is simply a qubit. In the same way, one can define s-working periods. Each time step in M_0 transforms to w time steps in M_1, and an s-working period is the time interval in M_r which corresponds to one time step in M_{r-s}.

The recursive simulation induces a partition of the set of locations in M_r to generalized rectangles. An r-rectangle in M_r is the set of locations which originated from one location in M_0. This is best explained by an example: Consider a CNOT gate which is applied in M_0 at time t on qubits q_1, q_2. The location $((q_1, q_2), t)$ in M_0 transforms in M_1 to error correction procedures on both blocks, followed by the procedure of the CNOT gate. The set of locations in these three procedures is the 1-rectangle in M_1 which originated from the location $((q_1, q_2), t)$ in M_0. More generally, an s-rectangle in M_r is the set of points in M_r which originated from one location in M_{r-s}. Note that the partition to s-rectangles is a refinement of the partition to $(s+1)$-rectangles. A 0-rectangle in M_r is just one location.

7.4. Sparse errors and sparse faults. In a noiseless scenario, the state of M_r at the end of each r-working period encodes the state of M_0 at the end of the corresponding time step. However, we assume that errors occur in M_r, and we want to analyze those. In order to analyze the propagation of errors in M_r, we need to distinguish between the actual faults that occur during the computation and the errors that are caused in the state. First, we focus on the errors in the states and define a distance between encoded states. The hierarchy of blocks requires a recursive definition.

DEFINITION 21. *Let B be the set of qubits in n r-blocks. An (r,k)-sparse set of qubits A in B is a set of qubits in which, for every r-block in B, there are at most k $(r-1)$-blocks such that the set A in these blocks is not $(r-1,k)$ sparse. A $(0,k)$-sparse set of qubits A is an empty set of qubits.*

Two density matrices ρ_1 and ρ_2 of the set of qubits B are said to be (r,k)-deviated if there exists an (r,k)-sparse set of qubits $A \subseteq B$, with $\rho_1|_{B-A} = \rho_2|_{B-A}$. The deviation satisfies the triangle inequality since the union of two sets which are (r,l_1)- and (r,l_2)-sparse, respectively, is (r,l_1+l_2)-sparse, by induction on r.

We will see that a computation is successful if the error at the end of each r-working period is sparse enough. The question is which fault paths keep the errors sparse. We will show in Lemma 9 that this is guaranteed if the fault path is sparse.

DEFINITION 22. *A set of locations in an r-rectangle is said to be (r,k)-sparse if there are no more than k $(r-1)$-rectangles in which the set is not $(r-1,k)$-sparse. A $(0,k)$-sparse set in a 0-rectangle is an empty set. A fault path in M_r is (r,k)-sparse if, in each r-rectangle, the set is (r,k)-sparse.*

7.5. The good part: Sparse fault paths keep the error sparse. We claim that if the fault path is sparse enough, then the error corrections keep the deviation small. The number of effective errors at any given level is thus constantly maintained below the dangerous zone. This is true at all levels, and indeed we prove it by induction on the level index r.

LEMMA 9. *Let C be a computation code that corrects q errors, with spread ℓ. Let M_r be the r-simulation of M_0 by C. Consider a computation subjected to an (r,k)-sparse fault path with $2k\ell \le q$. At the end of each r-working period the error is $(r,q/2)$-sparse.*

A condition of the form $k\ell \le q$ is natural, since k faults can indeed propagate to $k\ell$ errors if the spread is ℓ, and so we have to require that $k\ell \le q$ for the error correction to work. Our condition is in fact more restrictive, having the additional factor of 2, for technical reasons that will become apparent in the proof.

Proof. It is instructive to first prove this lemma for $r = 1$. This is done by induction on the time step t. For $t = 0$ the deviation is zero. Suppose that the density matrix at the end of the tth working period is $(1,q/2)$-deviated from the correct density matrix. If no errors occur during the tth working period, the error corrections would have corrected the state to $\phi(\rho(t))$, and the procedures would have taken it to the correct state $\phi(\rho(t+1))$. However, k faults did occur in each rectangle. We will use the fact that the rectangle has spread ℓ. Let us add the k faults to the fault path in the rectangle one by one. While adding the ith fault, we have $(q/2 + (i-1))\ell \le q - \ell$, because $i \le k$, and $2k\ell \le q$. Hence, this satisfies the requirement in Definition 17, and so we can deduce that each of the faults we add increases the deviation in the state at the end of the rectangle by at most ℓ qubits in each block. Thus, we have in each block at most $k\ell$ qubits which are affected by the faults, and the final deviation is at most $k\ell \le q/2$. This proves the theorem for $r = 1$.

For general r, we prove two assertions together, by using induction on r. The first assertion implies the desired result.

1. Consider n r-blocks, in a density matrix ρ_r which is $(r,q/2)$-deviated from $\phi^r(\rho_0)$, where ρ_0 is a density matrix of n qubits. At the end of an r-working period which r-simulates the operation of g_0 on ρ_0, with an (r,k)-sparse set of faults, the density matrix is $(r,q/2)$-deviated from $\phi^r(g_0 \circ \rho_0)$.
2. Consider n r-blocks in the state ρ_r such that x of the blocks are in an arbitrary density matrix and y of them are in a density matrix which is $(r,q/2)$-deviated from some word in the code $\phi^r(\rho_0)$. Consider an r-working period which r-simulates the operation g_0 with an (r,k)-sparse set of faults, applied on ρ_r. We claim that at the end of the r-working period the state is the same as if we started with the y blocks as they are and the x blocks corrected to some word in the code. More precisely, at the end of the r-working period the density matrix is $(r,q/2)$-deviated from $\phi^r(g_0 \circ \rho_0')$, where ρ_0' is a density matrix of n qubits which when reduced to the qubits corresponding to the y good blocks is equal to ρ_0.

For $r = 1$ the proof of the first assertion is as before. The second assertion is true because of a similar argument, using the second requirement in the definition of a quantum computation code (Definition 20), namely, that the error correction

procedure takes any density matrix to a word in the code. Let us now assume both claims for r and prove each of the claims for $r+1$.

Proof of Assertion 1. We consider an $(r+1)$-working period operating on ρ_{r+1}, $(r+1)$-simulating the procedure encoding g_0 on ρ_0. The $(r+1)$-simulation working period can be seen as an r-simulation of one working period in M_1 which 1-simulates some computation in M_1, consisting of an error correction procedure followed by an encoded gate.

Consider for a moment the 1-simulation circuit. We know that this circuit has spread ℓ. By using the fact that $k\ell + q/2 \leq q$, we know that if at most k faults occurred in this 1-simulation, and if the input state is $(1, q/2)$-deviated from being correct, then the output state is at most $(1, q/2)$-deviated from the correct state, namely, from $\phi(g_0(\rho_0))$.

We now turn back to the r-simulation of this circuit, namely, to our $(r+1)$-working period. We now assume a wrong assumption: The state at the end of each r-working period (in our $(r+1)$-working period) is $(r, q/2)$-deviated from some word in the code. This means that the $q/2$ problematic input blocks are $(r, q/2)$-deviated from some word in the code and, moreover, that at the end of each of the problematic r-rectangles the state of these blocks is $(r, q/2)$-deviated from some word in the code. In this case the proof of the first assertion follows easily from our first induction assumption, as follows.

Let w be the number of r-working periods in the $(r+1)$-working period. By the induction assumption on the first assertion, we have that at the end of each one of the r-working periods the matrix is $(r, q/2)$-deviated from what it is supposed to be by the computation we are r-simulating, namely, by the 1-simulation with spread ℓ. After w applications of this induction assumption, we get that the matrix at the end is $(r, q/2)$-deviated from a matrix $\phi^r(\rho_1)$, where ρ_1 is $(1, q/2)$-deviated from $g_0(\rho_0)$. This implies that the final density matrix is $(r+1, q/2)$-deviated from the correct matrix $\phi^{r+1}(g_0(\rho_0))$.

To release the wrong assumption, we use the induction hypothesis on the second assertion. We claim that nothing changes in the above argument if we apply, before each r-working period, an error correction procedure on each r-block, which has an (r, k)-sparse set of faults. By the induction assumption on the first assertion, for blocks which are $(r, q/2)$-deviated from some word in the code, this will remain the case after this error correction. Consider a block which is not $(r, q/2)$-deviated from some word in the code. Then the error correction will indeed change it to a block which is $(r, q/2)$-deviated from some word in the code. If the next r-rectangle that operates on that block is bad (namely, the fault path in this r-rectangle is not (r, k)-sparse), then the fact that we have added the correction does not change anything in the argument, since the faults can be chosen adversarially to ruin the entire block. If the next r-rectangle is good (namely, the fault path in it is (r, k)-sparse), then by the induction assumption on the second assertion this correction will happen in any case, and thus does not matter. Hence, the above argument goes through.

Proof of Assertion 2. The proof follows almost exactly the same argument as for the first assertion. We consider again an r-simulation of a 1-simulation. The first stage of the 1-simulation we consider here is a 1-error correction which takes any word to some word in the code, and so it is supposed to take the 1-blocks corresponding to the y-blocks to their state $\phi(\rho_0)$ and the x-blocks to some state. Hence, the 1-simulation is supposed to take the state to $\phi(\rho_0')$ as in the assertion. The remainder of the procedure is supposed to apply $\phi(g_0)$ to this state. Once again impose the wrong assumption as in the proof of the first assertion, and use the fact that the

spread of the 1-circuit is ℓ. By the induction assumption on the first assertion, we have that, at the end of each one of the r-working periods in the r-simulation of the above 1-simulation, the matrix is $(r, q/2)$-deviated from what it is supposed to be. We get that, under this assumption, the final matrix would be $(r, q/2)$-deviated from a matrix $\phi^r(\rho_1)$, where ρ_1 is $(1, q/2)$-deviated from $\phi(\rho')$. This implies that the final matrix is $(r+1, q/2)$-deviated from $\phi^r(\rho'_0)$. The relaxation of the wrong assumption is done exactly as before. □

This lemma implies that, if the faults are sufficiently sparse, the level of deviation can be kept below a certain value for the entire computation, and so at the end of the computation the set of errors is sparse. We need to show that in this case the recursive majority of the output bits indeed gives the correct answer.

LEMMA 10. *Let $2q+1 \leq m$. Let the final density matrix of M_r be $(r, q/2)$-deviated from the correct one. Consider the distribution on n bit strings which is obtained by measuring the output and taking recursive majority on each r-block. This distribution is equal to the output distribution of M_0.*

Proof. Let ρ_r be the correct final density matrix of M_r. Let ρ'_r be the final density matrix which is $(r, q/2)$-deviated from ρ_r. This remains true if we apply a measurement of all of the qubits and also if we discard certain r-blocks which do not correspond to the output of the computation. We can thus assume that ρ_r and ρ'_r are the density matrices of the output qubits, that they are mixtures of basis states, and that ρ'_r is $(r, q/2)$-deviated from ρ_r.

Note that, due to the reading procedure, the correct density matrix ρ_r is in the subspace which is spanned by basis states in which all of the coordinates in one r-block are equal, i.e., are of the form $|i_1^{m^r}, i_2^{m^r}, \ldots, i_n^{m^r}\rangle$. We have to show that, when we measure the qubits as a density matrix which is $(r, q/2)$-deviated from such a matrix, and we take the recursive majority, we get the correct distribution.

The matrix ρ_r can be written as $\{p_i, |\alpha_i\rangle\}$, where $|\alpha_i\rangle = |i_1^{m^r}, i_2^{m^r}, \ldots, i_n^{m^r}\rangle$. All qubits in an r-block are equal, since the matrix is correct. The probability that the recursive majority string is i is exactly p_i. ρ'_r can be written as $\{q_j, |\beta_j\rangle\}$, where $|\beta_j\rangle$ are basis vectors. Let A be a subset of qubits such that $\rho_r|_A = \rho'_r|_A$ and A contains all of the qubits on which ρ_r operates, except an $(r, q/2)$-sparse set. We know that the reduced density matrix of ρ'_r to A is the same as that of ρ_r, and so we have that ρ'_r is supported only by $|\beta_j\rangle$ such that their restrictions to qubits in A are strings which agree on all qubits in each r-block. Moreover, we have

$$\sum_{j, \beta_j|_A = \alpha_i|_A} q_j = p_i. \tag{7.1}$$

It is easy to see that the recursive majority string for $|\beta_j\rangle$ in the above sum is the same as that of $|\alpha_i\rangle$ and therefore is equal to i. Hence, we have that the probability for the recursive majority to be i in ρ'_r is p_i, and this completes the proof. □

7.6. The bad part: Nonsparse fault paths are rare below the threshold. Let us first analyze the improvement in the effective error rate when going from M_0 to M_1. By Lemma 9 we know that, as long as the number of faults k in each rectangle in M_1 satisfies $k \leq q/2\ell$ (where q is the number of errors the code corrects and ℓ is the spread of the code), then the computation at the end is correct. Therefore, the probability for more faults than $q/2\ell$ in one rectangle is a good definition of the effective error rate: If no such event happened, than there is no error. We can easily upper bound this probability from above. We set $k_0 = \lfloor q/2\ell \rfloor$ and denote by A the

number of locations in the largest rectangle in any simulation using C. Then

$$\eta_{\text{eff}} = \binom{A}{k_0+1} \eta^{k_0+1} \tag{7.2}$$

is an upper bound on the probability for more than k_0 faults in one rectangle. We refer to this bound as the effective error rate of the new circuit M_1 (we have obviously overestimated the effective error rate in this definition, but we do not attempt to optimize our analysis here).

DEFINITION 23 (the threshold). *We define the threshold η_c to be the error rate for which $\eta_{\text{eff}} = \eta$:*

$$\eta_c = \frac{1}{\binom{A}{k_0+1}^{1/k_0}}. \tag{7.3}$$

The following claim justifies the term *threshold*.

CLAIM 8. *Suppose that $k_0 \geq 1$. For any η below the threshold $\eta < \eta_c$, we have that the effective error rate is strictly smaller than the actual error rate: $\eta_{\text{eff}} < \eta$.*

Proof. Note that $x = \eta/\eta_c < 1$.

$$\eta_{\text{eff}} = \binom{A}{k_0+1} \eta^{k_0+1} = \binom{A}{k_0+1} \eta_c^{k_0+1} x^{k_0+1} = \eta_c x^{k_0+1} = \eta x^{k_0} < \eta. \qquad \square$$

We now show that, below the threshold, bad (namely, nonsparse) fault paths are rare.

LEMMA 11. *If $\eta < \eta_c$, $\exists \delta > 0$ such that the probability $P(r)$ for the fault path restricted to an r-rectangle to be (r, k_0)-sparse is larger than $1 - \eta^{(1+\delta)^r}$.*

Proof. Let δ be such that

$$\binom{A}{k_0+1} \eta^{k_0+1} < \eta^{1+\delta}. \tag{7.4}$$

Such a δ exists for η below the threshold, by Claim 8.

The proof of the lemma follows by induction on r. The probability for a 0-rectangle, i.e., one location, to have faults which are $(0, k_0)$-sparse, namely, the probability that in this location a fault did not occur, is $1 - \eta$.

Assume for r, and let us prove for $r+1$. For the faults in an $(r+1)$-rectangle not to be $(r+1, k_0)$-sparse, there must be at least $k_0 + 1$ r-rectangles in which the fault path is not (r, k_0)-sparse. So

$$P(r+1) \geq 1 - \binom{A}{k_0+1} (1 - P(r))^{k_0+1} > 1 - \eta^{(1+\delta)^{r+1}},$$

by using the induction assumption and the fact that $\eta^{(1+\delta)^r} < \eta < \eta_c$, and hence (7.4) can be applied with $\eta^{(1+\delta)^r}$ instead of η. $\square$

We can now prove the threshold theorem.

THEOREM 12 (the threshold theorem for probabilistic noise). *Let $\epsilon > 0$. Let C be a computation code using a set of gates $\mathcal{G}$. There exist a constant threshold $\eta_c > 0$ and constants c_1, c_2, c_3 such that the following holds. Let Q be a quantum circuit, with n input qubits (qudits), which operates t time steps, uses s gates from $\mathcal{G}$, and has v locations. There exists a quantum circuit Q' which operates on $n \cdot O(\log^{c_1}(\frac{v}{\epsilon}))$ qubits*

(qudits), for time $t \cdot O(\log^{c_2}(\frac{v}{\epsilon}))$*, and uses* $v \cdot O(\log^{c_3}(\frac{v}{\epsilon}))$ *gates from* $\mathcal{G}$ *such that, in the presence of probabilistic noise with error rate* $\eta < \eta_c$*,* Q' *computes a function which is within* ϵ *total variation distance to that computed by* Q*.*

Proof. We set M_0 to be equal to Q. Let $k_0 = \lfloor q/2\ell \rfloor$ as before. We set δ to be as in (7.4). We choose r such that $v\eta^{(1+\delta)^r} < \epsilon$. We then define Q' to be M_r generated according to the above scheme. By Lemma 11, we have that the probability for a fault path to be bad, i.e., not (r, k_0)-sparse, is smaller than ϵ. By Lemmas 9 and 10, the (r, k_0)-sparse fault paths give correct outputs. The resulting function (or distribution) is thus within ϵ total variation distance from correct.

As for the overhead, the number of qubits in Q' is nm^r, the number of time steps is tw^r, where w is the largest number of time steps in one rectangle, and the number of gates is vA^r. The proof follows from our choice of r, which turns out to be some constant times $\log\log(\epsilon/v)$. ∎

8. The threshold result for general noise. So far, we have dealt only with probabilistic faults. Actually, the circuit generated by the above recursive scheme is robust also against general local noise. This makes the applicability of the result much wider.

We prove this by writing the noise operator on each qubit as the identity plus a small error term. By expanding the error terms in powers of η, we get a sum of terms, each corresponding to a different fault path. The threshold result for general noise is again proved by dividing the faults into good and bad parts. First, we show that the bad part is negligible, i.e., that the norm of the sum of all terms which correspond to nonsparse fault paths is small. The proof that the good faults are indeed good, i.e., that the error in the terms corresponding to sparse fault paths is sparse, is based on the proof for probabilistic errors, together with some linearity considerations.

The threshold one gets in this case is slightly worse.

8.1. Fault paths in the case of general noise. The notion of fault paths is less clear in the case of general noise. To define fault paths, write the final density matrix of the noisy circuit as follows:

$$\rho(t) = \mathcal{E}(t) \cdot \mathcal{L}(t) \cdot \mathcal{E}(t-1) \cdot \mathcal{L}(t-1) \ldots \mathcal{E}(0) \cdot \mathcal{L}(0)\rho(0). \tag{8.1}$$

In the above equation, $\mathcal{E}(t)$ is the noise operator operating at time t, and $\mathcal{L}(t)$ is the computation operator at time t. According to our noise model (2.3), $\mathcal{E}(t)$ can be written as a tensor product of operators, operating on the possible locations of faults at time t, $A_{i,t}$. Each such operator can be written as a sum of two operators, by using (2.4):

$$\mathcal{E}_{A_{i,t}}(t) = (1-\eta)I + \mathcal{E}'_{A_{i,t}}(t), \quad \|\mathcal{E}'_{A_{i,t}}(t)\| \leq 2\eta. \tag{8.2}$$

We can replace all of the error operators in (8.1) by the products of operators of the form (8.2). We get:

$$\begin{aligned}\rho(t) = &\left(\otimes_{A_{i,t}}((1-\eta)I + \mathcal{E}'_{A_{i,t}})\right) \cdot \mathcal{L}(t) \cdot \left(\otimes_{A_{i,t-1}}((1-\eta)I + \mathcal{E}'_{A_{i,t-1}})\right) \cdot \mathcal{L}(t-1) \\ &\ldots \left(\otimes_{A_{i,0}}((1-\eta)I + \mathcal{E}'_{A_{i,t-1}})\right) \mathcal{L}(0)\rho(0).\end{aligned} \tag{8.3}$$

We can open up the brackets in the above expression. We get a sum of terms, where in each term, for each set of qubits $A_{i,t}$ at time t, we operate either $(1-\eta)I$ or $\mathcal{E}'_{A_{i,t}}(t)$. Thus, each term in the sum corresponds to a certain fault path. More

precisely, a fault path is a subset of the locations $\{A_{i,t}\}_{i,t}$, and the term in the sum which corresponds to this fault path is exactly the term in which we apply $\mathcal{E}'_{A_{i,t}}(t)$ on all locations in the fault path and apply $(1-\eta)I$ on the rest. As was done in the probabilistic case, we can now divide the above sum (8.3) into two parts: the sum over the *good* fault paths and the sum over the *bad* fault paths. We define the good fault paths to be those which are (r, k_0)-sparse, and the bad ones are all of the rest. We write

$$\rho(t) = \mathcal{L}_g \cdot \rho(0) + \mathcal{L}_b \cdot \rho(0). \tag{8.4}$$

We will treat each part separately.

8.2. The bad part: Nonsparse fault paths are negligible below the threshold. We show that the trace norm of the *bad* part is negligible, when η is below the threshold for general noise.

DEFINITION 24. *The threshold for general noise $\eta'_c > 0$ is defined to be the error rate η such that*

$$e \begin{pmatrix} A \\ k_0+1 \end{pmatrix} (2\eta)^{k_0+1} = 2\eta, \tag{8.5}$$

where, as before, $k_0 = \lfloor q/2\ell \rfloor$, q is the number of errors which the code corrects, ℓ is the spread of the code, and A is the maximal number of locations in a rectangle. The threshold for general noise for the code C is thus

$$\eta'_c = \frac{1}{2} e^{-k_0} \begin{pmatrix} A \\ k_0+1 \end{pmatrix}^{-k_0}. \tag{8.6}$$

Note that there is a slight difference from the threshold in the case of probabilistic noise. A factor of 2 is added to η, and the factor of e is added to the whole definition. These differences are due to the fact that the norm of the good operators in the general noise case is not smaller than 1 but can also be slightly larger than 1. η'_c is thus taken to be smaller than the threshold for probabilistic noise η_c.

We now prove that the contribution of the bad fault paths is indeed small.

LEMMA 12. *Let $\eta \langle \eta'_c, \epsilon \rangle 0$. Let M_0 use v locations. Let $r = \text{polyloglog}(\frac{v}{\epsilon})$. Then in M_r*

$$\|\mathcal{L}_b\| \leq \epsilon, \quad \|\mathcal{L}_g\| \leq 1 + \epsilon.$$

Proof. We shall rewrite the sum over all fault paths, by collecting together all of the operations according to in which r-rectangles they were done. In other words, we order the r-rectangles in some topological order, and then apply all of the operators that belong to one rectangle, before we start with a different rectangle. We now open up the operators corresponding to one r-rectangle, by using (8.2). We denote by $\mathcal{L}_b(i)$ the sum over all operators on the ith r-rectangle that correspond to errors on a bad fault path in this rectangle. Similarly, $\mathcal{L}_g(i)$ is the sum over all operators on the ith rectangle that correspond to good fault paths. If there are v procedures, we can write

$$\rho(t) = (\mathcal{L}_g(v) + \mathcal{L}_b(v)) \cdot (\mathcal{L}_g(v-1) + \mathcal{L}_b(v-1)) \ldots (\mathcal{L}_g(1) + \mathcal{L}_b(1))\rho(0) \tag{8.7}$$

for $\rho(0)$ being the initial density matrix. We first prove that

$$\forall 1 \leq i \leq v, \quad \|\mathcal{L}_b(i)\| \leq (2\eta)^{(1+\delta)^r}, \quad \|\mathcal{L}_g(i)\| \leq 1 + (2\eta)^{(1+\delta)^r}, \tag{8.8}$$

where δ is defined by

$$e\begin{pmatrix} A \\ k_0+1 \end{pmatrix}(2\eta)^{k_0+1} < (2\eta)^{1+\delta}. \tag{8.9}$$

Such a δ exists because η is below the threshold defined in (8.6). The proof of inequality (8.8) for $\eta < \eta_c'$ follows the lines of Lemma 11.

We use induction on r. It is useful to add the superscript r and denote by $\mathcal{L}_b^r(i)$ the sum over all bad fault paths in an r-rectangle. For $r = 0$, a 0-rectangle is simply one location. Hence $\mathcal{L}_b^0(i)$, which is the sum over all *bad* fault paths in this location, consists of only one term: the gate applied in this location (which could be the identity) followed by one noise operator. By (8.2), and the properties of the norm on superoperators in section 2.9, $\|\mathcal{L}_b^0(i)\| \le 2\eta$ and $\|\mathcal{L}_g^0(i)\| \le 1 + 2\eta$.

We now assume for r and prove for $r+1$. For the faults in an $(r+1)$-rectangle *not* to be $(r+1, k_0)$-sparse, there must be at least k_0+1 r-rectangles in which the fault is not (r, k_0)-sparse. So by the induction assumption on both $\mathcal{L}_b(i)$ and $\mathcal{L}_g(i)$

$$\begin{aligned} \|\mathcal{L}_b^{r+1}(i)\| &\le \begin{pmatrix} A \\ k+1 \end{pmatrix}((2\eta)^{(1+\delta)^r})^{k+1}(1+(2\eta)^{(1+\delta)^r})^{A-k-1} \\ &\le e\begin{pmatrix} A \\ k+1 \end{pmatrix}((2\eta)^{(1+\delta)^r})^{k+1}, \end{aligned} \tag{8.10}$$

where we have used the fact that $(1+(2\eta)^{(1+\delta)^r})^{A-k-1} < e$, since $(2\eta)^{(1+\delta)^r} < 2\eta$ and $2\eta A \le 1$. The right-hand side is $\le (2\eta)^{(1+\delta)^{r+1}}$ by using the fact that $(2\eta)^{(1+\delta)^r} < 2\eta$ and the fact that the threshold condition of (8.6) is satisfied for $\eta < \eta_c'$. This proves the induction step for $\mathcal{L}_b(i)$. By using $\|\mathcal{L}_g(i) + \mathcal{L}_b(i)\| = 1$ we can prove the induction step also for $\mathcal{L}_g(i)$. To prove the statement, we consider bad fault paths, i.e., at least one r-rectangle is bad. If there are v rectangles, we have

$$\|\mathcal{L}_b\| \le v \cdot (1 + (2\eta)^{(1+\delta)^r})^{v-1}(2\eta)^{(1+\delta)^r}. \tag{8.11}$$

Setting $r = \text{polyloglog}(\frac{v}{\epsilon})$ gives the desired result. ∎

8.3. The good part: Sparse fault paths give almost correct outputs.

LEMMA 13. *Let $\eta < \eta_c', \epsilon > 0$. Let M_0 have v locations. Let P_i be the probability to measure the string i in M_0 (operating without noise). Let $r = \text{polyloglog}(\frac{v}{\epsilon})$, as in Lemma* 12*, and let M_r be defined as before. We consider the operator $\mathcal{L}_g$ as in* (8.4)*, and $\rho(0)$ as the input density matrix of M_r. Then*

$$\left|\sum_{j \mapsto i}[\mathcal{L}_g \cdot \rho(0)]_{j,j} - P_i\right| \le \epsilon,$$

where $j \mapsto i$ means that taking recursive majority on the string j results in the string i.

Proof. The measurement of the output qubits of any density matrix of nm^r qubits induces some probability distribution over n-bit strings by taking the recursive majority. For any density matrix ρ which is $(r, q/2)$-deviated from the correct final density matrix of M_r, this probability distribution is the same one as the output distribution of M_0. In other words, the probability for each n-bit string is the same as that of M_0.

Since $\mathcal{L}_g$ corresponds to sparse fault paths, we would now like to apply Lemma 9 to show that $\mathcal{L}_g\rho(0)$ is $(r, q/2)$-deviated from the correct final matrix. However,

since we are dealing with general noise, the matrix $\mathcal{L}_g\rho(0)$ is not necessarily a density matrix or even a positive-semidefinite matrix. This is because the operators $\mathcal{E}'_{A_{i,t}}$ that it involves are not necessarily completely positive.

To bypass this technical issue, we recall that

$$\mathcal{E}'_{A_{i,t}} = \mathcal{E}_{A_{i,t}} - (1-\eta)I, \tag{8.12}$$

which is a weighted sum of physically admissible operators ($\mathcal{E}_{A_{i,t}}$ and I). Hence, we can write $\mathcal{L}_g$ as a weighted sum of (r, k_0)-sparse fault paths that *do* correspond to physically admissible operators:

$$\mathcal{L}_g \cdot \rho(0) = \sum_f \lambda_f \mathcal{L}_f \cdot \rho(0), \tag{8.13}$$

where each term in the above sum $\mathcal{L}_f \cdot \rho(0)$ corresponds to an evolution of the initial density matrix subject to physically admissible faults operating in a set of locations which is (r, k_0)-sparse. Lemma 9 thus applies to each term in the sum. We get that each density matrix $\mathcal{L}_f \cdot \rho(0)$ is $(r, q/2)$-deviated from correct. This means that

$$\sum_{j\mapsto i}[\mathcal{L}_g \cdot \rho(0)]_{j,j} = \sum_f \lambda_f \sum_{j\mapsto i}[\mathcal{L}_f \cdot \rho(0)]_{j,j} = \left(\sum_f \lambda_f\right) P_i. \tag{8.14}$$

However, by using (8.13), we have

$$\left|\sum_f \lambda_f - 1\right| = |\operatorname{Tr}(\mathcal{L}_g\cdot\rho(0)) - 1| = |\operatorname{Tr}((\mathcal{L}_g+\mathcal{L}_b)\cdot\rho(0)) - 1 - \operatorname{Tr}(\mathcal{L}_b\cdot\rho(0))| = |\operatorname{Tr}(\mathcal{L}_b\cdot\rho(0))|. \tag{8.15}$$

By using Property 1 of the superoperator norm (section 2) and Lemma 12, we have

$$|\operatorname{Tr}(\mathcal{L}_b \cdot \rho(0))| \le \|\mathcal{L}_b\|\|\rho(0)\| \le \epsilon, \tag{8.16}$$

which completes the proof. ∎

8.4. The threshold theorem for general noise. We can now prove the threshold result for general noise.

THEOREM 13 (the threshold theorem for general noise). *Let $\epsilon > 0$. Let C be a computation code with gates $\mathcal{G}$. There exist a threshold $\eta'_c > 0$ and constants c_1, c_2, c_3 such that the following holds. Let Q be a quantum circuit, operating on n qubits for t time steps, which uses s gates from $\mathcal{G}$, and has v locations. There exists a quantum circuit Q' which operates on $n\log^{c_1}(\frac{v}{\epsilon})$ qubits, for time $t\log^{c_2}(\frac{v}{\epsilon})$, and uses $v\log^{c_3}(\frac{v}{\epsilon})$ gates from $\mathcal{G}$ such that, in the presence of general noise with error rate $\eta < \eta'_c$, Q' computes a function which is within ϵ total variation distance from that computed by Q.*

Proof. We construct Q' to be the r-simulation of Q, where r is chosen such that the requirements of Lemma 12 are satisfied with $\epsilon' = \epsilon/2$. We now estimate the total variation distance between the output distribution of Q' (after taking recursive majority) and that of Q. As before, we denote the probability to measure i at the

output of Q by P_i. The total variation distance is then

(8.17)
$$\frac{1}{2}\sum_i \left| \sum_{j\mapsto i} \rho(t)_{j,j} - P_i \right| = \frac{1}{2}\sum_i \left| \sum_{j\mapsto i} ([\mathcal{L}_b \cdot \rho(0)]_{j,j} + [\mathcal{L}_g \cdot \rho(0)]_{j,j}) - P_i \right|$$
$$\leq \frac{1}{2}\left(\sum_i \left| \sum_{j\mapsto i} [\mathcal{L}_b \cdot \rho(0)]_{j,j} \right| + \sum_i \left| \sum_{j\mapsto i} [\mathcal{L}_g \cdot \rho(0)]_{j,j} - P_i \right| \right).$$

Lemma 13 implies that the second term is at most ϵ'. The first term is bounded from above by the trace norm of $\mathcal{L}_b \cdot \rho(0)$ and hence, by Lemma 7.6, by ϵ'. The sum of the two gives $2\epsilon' = \epsilon$. □

9. Fault tolerance with any universal set of gates. So far, the reliable circuits which we have constructed can use only universal set of gates associated with a quantum computation code, such as the sets $\mathcal{G}_1$ and $\mathcal{G}_2$. This is an undesirable situation, both theoretically and practically. Theoretically, we would like to be able to show that the fault-tolerance result is robust, meaning that fault-tolerant quantum computation can be performed regardless of the universal set of gates which we use. Practically, it is likely that the sets of gates $\mathcal{G}_1$ or $\mathcal{G}_2$ are difficult to implement in the laboratory. We would like to be able to implement fault-tolerant quantum computation by using the gates that are most readily available to us. Indeed, in this section we provide the desired generalization and show that the threshold result holds for any universal set of gates $\mathcal{G}$. In other words, starting from a quantum circuit which uses an arbitrary universal set of gates $\mathcal{K}$, we can implement it fault-tolerantly with any universal set of gates $\mathcal{G}$ of our choice. We require only that $\mathcal{G}$ contains a gate which discards a qubit (qudit) and a gate which adds a blank qubit (qudit) to the circuit.

The idea of the proof is that we design the final circuit in three stages: We start from the original circuit which uses gates from $\mathcal{K}$ and simulate it by a circuit which uses one of the sets of gates for which a quantum computation code exists, e.g., $\mathcal{G}_1$. We then apply our fault-tolerant construction and get a circuit M_r which uses once again gates from the same set associated with the computation code, say, $\mathcal{G}_1$. Finally, we replace every gate g in M_r by a sequence of gates from $\mathcal{G}$ which approximates the gate g to within a constant μ. The final circuit is denoted M'_r. We now prove that this construction works, if μ is chosen correctly.

In the following, we assume that we use the fault-tolerant construction with the set of gates $\mathcal{G}_1$. The discussion can be easily changed to use the gate set $\mathcal{G}_2$.

DEFINITION 25. *For any gate g in $\mathcal{G}_1$, consider the smallest circuit (in terms of number of locations) that uses gates from $\mathcal{G}$ and approximates g to within accuracy μ. Denote this circuit by $Q(g,\mu)$. Let $S(\mu)$ be the maximum number of locations in $Q(g,\mu)$ taken over all g in $\mathcal{G}_1$.*

CLAIM 9. *Fix a gate $g \in \mathcal{G}_1$. Consider the superoperator $Q'(g,\mu)$ which corresponds to applying the circuit $Q(g,\mu)$ in the presence of general noise of error rate η. We claim that, if $\eta < 1/2S(\mu)$, $\|g - Q'(g,\mu)\| \leq 3S(\mu)\eta + \mu$.*

Proof. By the definition of $Q(g,\mu)$, it suffices to prove that $\|Q(g,\mu) - Q'(g,\mu)\| \leq 3S(\mu)\eta$. We write down the superoperator corresponding to $Q'(g,\mu)$, as is done in

(8.3). By using the triangle inequality, and (2.4), we have

$$\begin{aligned}\|Q(g,\mu)-Q'(g,\mu)\| &\le S(\mu)(2\eta)(1-\eta)^{S(\mu)-1} \\ &\quad + \binom{S(\mu)}{2}(2\eta)^2(1-\eta)^{S(\mu)-2}+\cdots+(2\eta)^{S(\mu)} \\ &= (1+\eta)^{S(\mu)}-(1-\eta)^{S(\mu)}.\end{aligned} \tag{9.1}$$

Simple combinatorics now implies the result, taking into account that $S(\mu)\eta \le 0.5$. ∎

We view the circuit M'_r as composed of generalized locations, where each generalized location corresponds to the set of locations arising from the approximation of one gate in M_r. Let us compare M'_r in the presence of noise, viewed in this resolution of generalized locations, to the circuit M_r operating without noise. Claim 9 implies that at each generalized location the error in M'_r, compared to the correct gate in M_r, is at most $3S(\mu)\eta+\mu$. We would like to apply our result from section 8, and so we need to require that $3S(\mu)\eta+\mu<\eta'_c$, the threshold for general noise. By optimizing for μ we get the following definition for the threshold:

$$\eta''_c = \min\left\{\max_{\mu<\eta'_c}\frac{\eta'_c-\mu}{3S(\mu)}, \frac{1}{2S(\mu)}\right\}. \tag{9.2}$$

If η is smaller than this threshold, then Claim 9 applies, and we get that the effective error rate $\eta_{\text{eff}} = 3S(\mu)\eta+\mu$ is indeed below the threshold for general noise. This allows us to prove the threshold theorem by using any universal set of gates in the presence of general noise.

THEOREM 14 (the threshold result in full generality). *Let $\epsilon>0$. Let $\mathcal{K}$ and $\mathcal{G}$ be two universal sets of quantum gates. There exist a constant $\eta''_c>0$ and constants c_1,c_2,c_3 such that the following holds. Given any quantum circuit Q with n input qubits, which operates for t time steps, uses s gates from $\mathcal{K}$, and has v locations, there exists a corresponding quantum circuit Q' which operates on $nO(\log^{c_1}(\frac{v}{\epsilon}))$ qubits, for time $tO(\log^{c_2}(\frac{v}{\epsilon}))$, and uses $vO(\log^{c_3}(\frac{v}{\epsilon}))$ gates from $\mathcal{G}$ such that, in the presence of general noise with error rate $\eta<\eta''_c$, Q' computes a function which is within ϵ total variation distance from the function computed by Q.*

Proof. We first approximate Q by a circuit M_0 which uses only gates from the set of gates $\mathcal{G}_1$ of a computation code C. Our new circuit M_0 computes the same function as Q up to total variation distance $\epsilon/2$. We do this by approximating every one of the s gates in Q by a sequence of gates from $\mathcal{G}$ that approximates the gate to within ϵ/s. Due to the Kitaev–Solovay theorem (see section 6), the number of gates required to replace each gate in Q is $\text{polylog}(s/\epsilon)$.

We now construct M_r, the r-simulation of M_0, which again uses gates from $\mathcal{G}_1$. This is done as in section 7, except that the choice of r is taken to suit the error rate $\eta_{\text{eff}} = 3S(\mu_0)\eta+\mu_0$.

To construct Q', we replace each location in M_r by a circuit of $S(\mu_0)$ locations, by using gates from $\mathcal{G}$ that approximate the gate performed in the location to within μ_0.

We would now like to prove that the final circuit is fault-tolerant against general noise of error rate $\eta<\eta''_c$ by using essentially the same lines as in section 8. To be able to apply this proof, we do the following. We group the locations in the final circuit M'_r to groups of $S(\mu_0)$ locations, where each group corresponds to the location from which it originated in M_r. Consider the superoperator $\mathcal{L}$ associated with the application of the gates and noise operators in one generalized location corresponding

to the gate g. By Claim 9, $\|\mathcal{L}-g\| \leq \eta_{\text{eff}}$. Hence, $\|g^{-1}\mathcal{L}-I\| \leq \eta_{\text{eff}}$. We can therefore consider an equivalent circuit to M'_r, where for each generalized location, described by a superoperator $\mathcal{L}$ which consists of a sequence of gates and noise operators, we instead put the gate g followed by a noise operator $g^{-1}\mathcal{L}$. The proof now follows from the proof of Theorem 13, by using $\ell = S(\mu_0)$ for the spread of the procedures. □

The threshold value we get for arbitrary sets of gates (9.2) is of course worse than the threshold values for the fault-tolerant constructions that use gates of computation codes. The value of the threshold depends on the exact set of gates $\mathcal{G}$, because it depends on the number of gates required to approximate the gates of the computation code.

10. Robustness against exponentially decaying correlations. We would now like to show how the above results hold also in the case of exponentially decaying correlations between the noise processes, in both space and time.

10.1. Adding correlations to the noise model. Two very important assumptions were made when introducing our general model of noise.

- Locality: No correlations between environments of different qubits, except through the gates.
- The Markovian assumption: The environment is renewed at each time step, and hence no correlations between the environments at different time steps.

Both of these assumptions can be slightly released, to allow exponentially decaying correlations in both space and time, while the results of this paper still hold.

To add exponentially decaying correlations to the probabilistic noise model, we generalize it in the following way. Instead of considering independent probabilities for error in each location, we require that the probability for a fault path which contains k locations is bounded by some constant times the probability for the same fault path in the independent errors model:

$$\Pr(\text{fault path with } k \text{ errors}) \leq c\eta^k(1-\eta)^{v-k}, \tag{10.1}$$

where v is the number of locations in the circuit. Then an adversary chooses a noise operator which operates on all of the qubits in the fault path, without any restrictions. The most general case is as follows. The adversary adds some blank qubits, which are called the environment, at the beginning of the computation. At each time step, the adversary can operate a general operator on the environment and the set of qubits in the fault path at that time step. This model allows correlations in space, since the noise operator need not be of the form of a tensor product of operators on different locations. Correlations in time appear because the environment that the adversary added in is not renewed at each time step, so noise operators of different time steps are correlated. Note that the independent probabilistic noise process is a special case of this process.

10.2. Proof of the threshold theorem with exponentially decaying correlations. We observe that all of the lemmas that we have used in order to prove the threshold theorem for probabilistic noise hold in this case, except for one step which fails. It is the step that shows that bad fault paths are rare, in the case of probabilistic noise (Lemma 11). The proof of this lemma relies on the independence of faults. We observe that the proof of Lemma 11 is actually a union bound. In this section we show that the same threshold as is used for probabilistic noise (Definition 23) guarantees that the bad fault paths are rare also in the presence of probabilistic noise with exponentially decaying correlations.

We use a union bound argument, as follows. Consider fault paths in v r-rectangles. If a fault path is bad, there must be at least one r-rectangle in which it is bad, namely, not (r, k_0)-sparse. Let us concentrate on this r-rectangle. We first count the number of bad fault paths in this r-rectangle that have a minimal number of faulty locations. To do this, denote by F_j the number of minimal fault paths in a j-rectangle. We have

$$F_1 \leq \begin{pmatrix} A \\ k_0+1 \end{pmatrix} \tag{10.2}$$

and

$$F_{j+1} \leq \begin{pmatrix} A \\ k_0+1 \end{pmatrix} F_j^{k_0+1}. \tag{10.3}$$

We can solve the recursion to get

$$F_r \leq \begin{pmatrix} A \\ k_0+1 \end{pmatrix}^{\frac{(k_0+1)^r-1}{k_0}}. \tag{10.4}$$

A minimal bad fault path contains exactly $(k_0+1)^r$ locations. Now, v r-rectangles contain vA^r locations. We can bound the number of bad fault paths in v r-rectangles, that consist of $(k_0+1)^r + i$ locations, as follows. We first choose one r-rectangle (this gives a factor of v). In this rectangle we pick one of the possible minimal bad fault paths (this gives a factor of F_r). We can then choose the rest of the locations arbitrarily. This gives that the number of bad fault paths in v r-rectangles consisting of $(k_0+1)^r + i$ locations is at most

$$v \begin{pmatrix} A \\ k_0+1 \end{pmatrix}^{\frac{(k_0+1)^r-1}{k_0}} \begin{pmatrix} vA^r - (k_0+1)^r \\ i \end{pmatrix}. \tag{10.5}$$

By using (10.1), we have that the overall probability of the bad fault paths is at most

(10.6)

$$\begin{aligned} &\sum_{i=0}^{vA^r-(k_0+1)^r} v \binom{A}{k_0+1}^{\frac{(k_0+1)^r-1}{k_0}} \binom{vA^r-(k_0+1)^r}{i} c\eta^{(k_0+1)^r+i}(1-\eta)^{vA^r-(k_0+1)^r-i} \\ &\leq cv \left(\binom{A}{k_0+1}^{\frac{1}{k_0}} \eta \right)^{(k_0+1)^r} \sum_{i=0}^{vA^r-(k_0+1)^r} \binom{vA^r-(k_0+1)^r}{i} \eta^i (1-\eta)^{vA^r-(k_0+1)^r-i} \\ &= cv \left(\binom{A}{k_0+1}^{\frac{1}{k_0}} \eta \right)^{(k_0+1)^r}. \end{aligned}$$

The expression above decays exponentially fast to zero with r, if η is strictly below the threshold for probabilistic noise (Definition 23). This completes the proof. □

11. Fault tolerance in a *d*-dimensional quantum computer. We now proceed to our final generalization of the threshold theorem. So far, we allowed a gate to operate on any set of qubits, regardless of the actual location of these qubits in space. Here we consider quantum systems with geometrical constraints: The qubits

are embedded in space, in a one-, two-, or three-dimensional grid, and gates can be applied only to nearest-neighbor qubits. In the case of dimensionality higher than 1, and gates of more than two qubits, we also require that all qubits in a gate lie on the same line. We show that the threshold result holds in full generality for d-dimensional quantum computers, for any $d \geq 1$.

In physical systems with geometrical constraints, discarding and adding qubits is a questionable process, unless we allow empty spots. To avoid this complication, we replace these two operations by one gate called RESTART, which is constructed by discarding a qubit and then adding a blank qubit instead of the discarded qubit. This allows us to remove entropy out of the system while maintaining the geometrical structure. Since the gates are restricted to operate on nearest neighbors, we also need to require that the set of gates we use contains the SWAP gate, so that, when we want to apply a gate on qubits that are not nearest neighbors, we can bring them closer together.

THEOREM 15 (threshold theorem for d-dimensional circuits). *Let $\epsilon > 0$. Let $d \geq 1$. Let $\mathcal{G}'$ and $\mathcal{G}''$ be two universal sets of quantum gates. There exist a threshold $\eta_c''' > 0$ and constants c_1, c_2, c_3 such that the following holds. Let Q' be a d-dimensional quantum circuit, with n input qubits, which operates t time steps, uses s gates from $\mathcal{G}'$, and has v locations. There exists a d-dimensional quantum circuit Q'' which operates on $nO(\log^{c_1}(\frac{v}{\epsilon}))$ qubits, for time $tO(\log^{c_2}(\frac{v}{\epsilon}))$, and uses $vO(\log^{c_3}(\frac{v}{\epsilon}))$ gates from $\mathcal{G}'' \cup \{\mathsf{SWAP}, \mathsf{RESTART}\}$ such that, in the presence of general noise with error rate $\eta < \eta_c'''$, Q'' computes a function which is ϵ-close to that computed by Q'.*

Proof. To prove the theorem, we need to modify the fault-tolerant procedures so that they apply gates on nearest neighbors only and also all of the qubits remain in the circuit throughout the computation.

Here is how one level of the simulation is done. We first pick a preferred direction, and each qubit will be extended to an array of qubits lying in that direction. The simulation will "stretch" the simulated circuit only in the preferred direction, by a constant factor. Let a be the maximal number of ancilla qubits used in any of the procedures of the computation code $\mathcal{G}$. Let m be the size of the block. A qubit in Q' will be replaced by $m + a$ qubits, placed in a line along the preferred direction. The ancilla qubits will serve as a working space, but we will also SWAP computational qubits with ancilla qubits, in order to bring computation qubits closer and operate gates on them.

The fault-tolerant procedures are thus modified as follows. First, instead of adding ancilla qubits during the procedure, we use only the ancilla qubits that are already there and apply a RESTART gate on an ancilla qubit one step before we use it in the procedure. Also, any gate g in the original procedure that operates on qubits that are far apart is replaced by a sequence of SWAP gates which bring the qubits that g operates on to nearest neighbor sites, followed by g, followed by another sequence of SWAP gates which bring the qubits back to their original sites. Since the simulated circuit Q' applies gates only on nearest neighbors, say, on $s \in \{1, 2, 3\}$ qubits in a row, the number of SWAP gates used in the above modification per gate g is at most $2(s-1)(m+a)$, i.e., a constant. This means that the modified procedure is larger than the original one by a constant factor (both in time and in space).

The claim is that the procedure is still fault-tolerant. This might seem strange since the SWAP gates operate on many qubits and seem to help in the propagation of errors. However, note that a SWAP gate which operates on a faulty qubit and an unaffected qubit does not propagate the error to the two qubits but keeps it confined to the original qubit, which is now in a new site. Hence, a SWAP gate which is not

faulty does not cause a propagation of error. If a fault does occur in a SWAP gate, then the two qubits participating in it are contaminated. If the fault occurred before the application of the gate g, then one of the qubits the gate g operates on is faulty, and hence this also causes the contamination of all of the qubits on which g operates (in the worst case). So an error in a SWAP gate is equivalent to an error in all of the original sites of the qubits participating in the gate and also the final site of the other qubit participating in the SWAP gate. This adds a factor of 2 at most to the original spread of the procedure. All other aspects of the theorem remain the same. □

12. Threshold estimations. Finally, it is left to estimate the exact threshold value for fault-tolerant quantum computation. However, there is not one such value: This value depends on many things such as which variants of the threshold theorem we use (probabilistic noise, general local noise, d-dimensional circuits) as well as the choice of the computation code and, mainly, the exact assumptions we make on the quantum system, most importantly, whether we allow classical computation in the middle of the quantum process or not. Since our results are mainly proofs of existence, where we have not attempted to optimize the threshold, we do not attempt to provide exact values of the threshold in all of these cases. Nevertheless, we give here a rough estimation of the threshold value in one of the simpler cases: the case of probabilistic independent noise, with no geometrical constraints on the system, and under the assumption that infinitely fast classical operations are allowed during the computation.

To estimate this threshold, we examine the formula for the threshold value in the case of independent probabilistic noise with no geometrical constraints, which is given by Definition 23. The threshold value depends on two parameters: A, the number of locations in the biggest rectangle in the simulation, and $k_0 = \lfloor q/2\ell \rfloor$, where we recall that q is the number of errors that the code can correct, and ℓ is its spread, which is 1 in both our constructions. We use polynomial codes of degree $d = 4$, i.e., length $m = 13$, so that $q = 2$, and so $k_0 = 1$. The threshold in our case is thus $\binom{A}{2}^{-1}$.

The parameter A, the size of the largest rectangle, is estimated by counting the number of locations where quantum (rather than classical) gates and qubits are involved. The bottleneck in our construction using polynomial codes, namely, the largest rectangle, is the one corresponding to applying transversal operation on two blocks, preceded by error correction on both of these blocks. The reason for this is as follows. First, we need to consider only two-qubit encoded gates because we use the squaring gate instead of the Toffoli gate in the set $\mathcal{G}_1$ (see subsection 4.2). Second, one might think that the largest rectangle involves the ancilla state preparation for the degree reduction as well. However, to avoid such large rectangles, we consider each step in the ancilla preparation as a rectangle of its own.

It remains to estimate the size of the above rectangle. For each error correction we need two zero-state preparations, and so altogether we need four zero-state preparations plus transversal operations. Since we use $q = 2$, every zero-state preparation requires 25 preliminary zero-state preparations (not done fault-tolerantly) plus transversal operations. We estimate the size of the rectangle in this case to be of the order of 10^3 locations, which implies a threshold of the order of $\simeq 10^{-6}$.

The threshold value in other cases, and under different sets of assumptions on the noise and on the constraints of the quantum system, can be estimated by using (8.6), (9.2), and the definition of η_c''' as is defined in section 11, where the estimation of the parameters involved should depend on the assumptions being used.

Acknowledgments. We thank Peter Shor for discussions about his result, Thomas Beth for helping to construct the fault-tolerant Toffoli gate without measurements for the CSS codes, Richard Cleve for asking the question of nearest neighbors and suggesting the solution, Noam Nisan for a fruitful discussion regarding the correct model to use, and Prasad Gospel, Denis Gaitsgori, Erez Lapid, and Bob Solovay for helpful discussions regarding the proofs of universality. We are grateful to Hoi-Kwong Lo for correcting an error in the definition of polynomial codes. Finally, we thank Alesha Kitaev for a very helpful remark regarding the generalization to the general noise model and Daniel Gottesman for very helpful remarks and corrections regarding the final draft of this paper.

REFERENCES

[1] D. Aharonov, *Quantum Computation*, Annual Reviews of Computational Physics 6, D. Stauffer, ed., World Scientific, Singapore, 1998.

[2] D. Aharonov, *A quantum to classical phase transition in noisy quantum computers*, Phys. Rev. A, 62 (2000), 062311.

[3] D. Aharonov and M. Ben-Or, *Fault-tolerant quantum computation with constant error*, in Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC), El Paso, TX, 1997, pp. 176–188.

[4] D. Aharonov and M. Ben-Or, *Fault-Tolerant Quantum Computation with Constant Error Rate*, preliminary extended version of [3], available online at http://arxiv.org/abs/quant-ph/9906129 (1999).

[5] D. Aharonov and M. Ben-Or, *Polynomial simulations of decohered quantum computers*, in Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS), IEEE, Los Alamitos, CA, 1996, pp. 46–55.

[6] D. Aharonov, A. Y. Kitaev, and N. Nisan, *Quantum circuits with mixed states*, in Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC), Dallas, TX, 1998, pp. 20–30.

[7] D. Aharonov, M. Ben-Or, R. Impagliazzo, and N. Nisan, *Limitations of Noisy Reversible Computation*, available online at http://arxiv.org/abs/quant-ph/9611028 (1996).

[8] D. Aharonov and D. Gottesman, *Accuracy Thresholds: Can We Beat* 10^{-4}*?* Presentation at ITP Conference of Quantum Information, 2001, available online at http://online.itp.ucsb.edu/online/qinfo_c01/aharonov/.

[9] D. Aharonov, A. Y. Kitaev, and J. Preskill, *Fault tolerant quantum computation with long-range correlated noise*, Phys. Rev. Lett., 96 (2006), 050504.

[10] C. S. Ahn, *Extending Quantum Error Correction: New Continuous Measurement Protocols and Improved Fault-Tolerant Overhead*, Ph.D. thesis, Caltech, Pasadena, CA, 2004.

[11] R. Alicki, M. Horodecki, P. Horodecki, and R. Horodecki, *Dynamical description of quantum computing: Generic nonlocality of quantum noise*, Phys. Rev. A, 65 (2002), 062101.

[12] R. Alicki, D. A. Lidar, and P. Zanardi, *Internal consistency of fault-tolerant quantum error correction in light of rigorous derivations of the quantum Markovian limit*, Phys. Rev. A, 73 (2006), 052311.

[13] P. Aliferis and A. Cross, *Subsystem fault tolerance with the Bacon–Shor code*, Phys. Rev. Lett., 98 (2007), 220502.

[14] P. Aliferis, D. Gottesman, and J. Preskill, *Quantum accuracy threshold for concatenated distance-3 codes*, Quant. Inf. Comput., 6 (2006), pp. 97–165.

[15] P. Aliferis, D. Gottesman, and J. Preskill, *Accuracy threshold for postselected quantum computation*, Quant. Inf. Comput., 8 (2008), pp. 181–244.

[16] D. Bacon, *Operator quantum error correcting subsystems for self-correcting quantum memories*, Phys. Rev. A, 73 (2006), 012340.

[17] D. Bacon, J. Kempe, D. Lidar, and B. K. Whaley, *Universal fault-tolerant computation on decoherence-free subspaces*, Phys. Rev. Lett., 85 (2000), pp. 1758–1761.

[18] J. Kempe, D. Bacon, D. Lidar, and B. K. Whaley, *Theory of decoherence-free fault-tolerant universal quantum computation*, Phys. Rev. A, 63 (2001), 042307.

[19] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, *Elementary gates for quantum computation*, Phys. Rev. A, 52 (1995), pp. 3457–3467.

[20] A. Barenco, A. Ekert, K. A. Suominen, and P. Torma, *Approximate quantum Fourier transform and decoherence*, Phys. Rev. A, 54 (1996), pp. 139–146.
[21] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, *Strengths and weaknesses of quantum computing*, SIAM J. Comput., 26 (1997), pp. 1510–1523.
[22] C. H. Bennett, D. P. DiVincenzo, J. A. Smolin, and W. K. Wootters, *Mixed state entanglement and quantum error correction*, Phys. Rev. A, 54 (1996), pp. 3824–3851.
[23] M. Ben-Or and R. Cleve, *Computing algebraic formulas using a constant number of registers*, SIAM J. Comput., 21 (1992), pp. 54–58.
[24] M. Ben-Or, S. Goldwasser, and A. Wigderson, *Completeness theorems for fault-tolerant distributed computing*, in Proceedings for the 20th Annual ACM Symposium on Theory of Computing (STOC), Chicago, 1988, pp. 1–10.
[25] A. Berthiaume and G. Brassard, *Oracle quantum computing*, in Proceedings of the Workshop on Physics of Computation: PhysComp '92, IEEE Computer Society Press, Los Alamitos, CA, 1992, pp. 60–62.
[26] P. O. Boykin, T. Mor, M. Pulver, V. Roychowdhury, and F. Vatan, *A new universal and fault-tolerant quantum basis*, Inform. Process. Lett., 75 (2000), pp. 101–107.
[27] S. Bravyi and A. Y. Kitaev, *Universal quantum computation with ideal Clifford gates and noisy ancillas*, Phys. Rev. A, 71 (2005), 022316.
[28] A. R. Calderbank and P. W. Shor, *Good quantum error-correcting codes exist*, Phys. Rev. A, 54 (1996), pp. 1098–1105.
[29] M. D. Choi, *Completely positive linear maps on complex matrices*, Linear Algebra Appl., 10 (1975), pp. 285–290.
[30] I. L. Chuang, R. Laflamme, P. W. Shor, and W. H. Zurek, *Quantum computers, factoring, and decoherence*, Science, 270 (1995), pp. 1633–1635.
[31] I. L. Chuang, W. C. D. Leung, and Y. Yamamoto, *Bosonic quantum codes for amplitude damping*, Phys. Rev. A, 56 (1997), pp. 1114–1125.
[32] B. Cirel'son (Tsirelson), *Reliable Storage of Information in a System of Unreliable Components with Local Interactions*, Lecture Notes in Math. 653, Springer-Verlag, New York, 1978, pp. 15–30.
[33] R. Cleve, D. Gottesman, and H.-K. Lo, *How to share a quantum secret*, Phys. Rev. Lett., 83 (1999), pp. 648–651.
[34] C. Cohen-Tanoudji, *Quantum Mechanics*, Wiley, New York, 1977.
[35] D. Coppersmith and E. Grossman, *Generators for certain alternating groups with applications to cryptography*, SIAM J. Appl. Math., 29 (1975), pp. 624–627.
[36] C. Crépeau, D. Gottesman, and A. Smith, *Secure multi-party quantum computation*, in Proceedings of the 34th Annual ACM Symposium on Theory of Computing, Montreal, Quebec, Canada, 2002, pp. 643–652.
[37] E. Dennis, A. Y. Kitaev, A. Landahl, and J. Preskill, *Topological quantum memory*, J. Math. Phys., 43 (2002), pp. 4452–4505.
[38] D. Deutsch, *Quantum theory, the Church-Turing principle, and the universal quantum computer*, Proc. R. Soc. Lond. Ser. A, 400 (1985), pp. 97–117.
[39] D. Deutsch, *Quantum computational networks*, Proc. R. Soc. Lond. Ser. A, 425 (1989), pp. 73–90.
[40] D. P. DiVincenzo, *Two-bit gates are universal for quantum computation*, Phys. Rev. A, 51 (1995), pp. 1015–1022.
[41] L.-M. Duan and G.-C. Guo, *Reducing decoherence in quantum computer memory with all quantum bits coupling to the same environment*, Phys. Rev. A, 57 (1998), pp. 737–741.
[42] P. Gács, *Self correcting two dimensional arrays*, in Advances in Computing Research: A Research Annual: Randomness and Computation, S. Micali, ed., JAI Press, Greenwich, CT, 1989, pp. 223–326 (see, in particular, pp. 240–241 and 246–248).
[43] C. W. Gardiner, *Quantum Noise*, Springer, Berlin, 1991.
[44] D. Gottesman, *A theory of fault-tolerant quantum computation*, Phys. Rev. A, 57 (1998), pp. 127–137.
[45] D. Gottesman, *Stabilizer Codes and Quantum Error Correction*, Ph.D. thesis, Caltech, Pasadena, CA; available online at http://arxiv.org/abs/quant-ph/9705052 (1997).
[46] D. Gottesman, *Fault-tolerant quantum computation with higher-dimensional systems*, Chaos Solitons Fractals, 10 (1999), pp. 1749–1758.
[47] D. Gottesman, *Fault-tolerant quantum computation with local gates*, J. Modern Opt., 47 (2000), pp. 333–345.
[48] D. Gottesman and I. Chuang, *Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations,* Nature, 402 (1999), pp. 390–393.
[49] L. A. Khalfin and B. S. Tsirelson, *Quantum/classical correspondence in the light of Bell's inequalities*, Found. Phys. 22 (1992), pp. 879–948.

[50] A. Y. Kitaev, *Quantum computations: Algorithms and error corrections*, Russian Math. Surveys, 52 (1997), pp. 1191–1249.
[51] A. Y. Kitaev, *Fault-tolerant quantum computation by anyons*, Ann. Physics, 303 (2003), pp. 2–30.
[52] A. Y. Kitaev, A. H. Shen, and M. N. Vyalyi, *Classical and Quantum Computation*, Grad. Stud. Math. 47, American Mathematical Society, Providence, RI, 2002.
[53] E. Knill, *Non-Binary Unitary Error Bases and Quantum Codes*, available online at http://arxiv.org/abs/quant-ph/9608048 (1996).
[54] E. Knill, *Quantum computing with very noisy devices*, Nature, 434 (2005), pp. 30–44.
[55] E. Knill and L. Viola, *Theory of quantum error correction for general noise*, Phys. Rev. Lett., 84 (2000), pp. 2525–2528.
[56] E. Knill, R. Laflamme, and W. Zurek, *Resilient quantum computation*, Science, 279 (1998), pp. 342–345.
[57] E. Knill, R. Laflamme, and W. H. Zurek, *Resilient Quantum Computation: Error Models and Thresholds*, available online at http://arxiv.org/abs/quant-ph/9702058 (1997).
[58] E. Knill and R. Laflamme, *Concatenated Quantum Codes*, available online at http://arxiv.org/abs/quant-ph/9608012 (1996).
[59] D. W. Kribs, R. Laflamme, D. Poulin, and M. Lesosky, *Operator quantum error correction*, Quantum. Inf. Comput., 6 (2006), pp. 383–399.
[60] K. Hellwig and K. Kraus, *Operations and measurements*. II, Comm. Math. Phys. 16 (1970), pp. 142–147.
[61] K. Kraus, *States, Effects and Operations. Fundamental Notions of Quantum Theory*, Springer-Verlag, Berlin, 1983.
[62] R. Landauer, *Is quantum mechanics useful?*, Philos. Trans. Roy. Soc. London Ser. A, 353 (1995), pp. 367–376.
[63] D. A. Lidar, L. C. Isaac, and B. K. Whaley, *Decoherence free subspaces for quantum computation*, Phys. Rev. Lett., 81 (1998), pp. 2594–2597.
[64] J. H. van Lint, *An Introduction to Coding Theory*, 2nd ed., Springer-Verlag, New York, 1992.
[65] C. Miquel, J. P. Paz, and R. Perazzo, *Factoring in a dissipative quantum computer*, Phys. Rev. A, 54 (1996), pp. 2605–2613.
[66] C. Miquel, J. P. Paz, and W. H. Zurek, *Quantum computation with phase drift errors*, Phys. Rev. Lett., 78 (1997), pp. 3971–3974.
[67] J. von Neumann, *Probabilistic logic and the synthesis of reliable organisms from unreliable components*, in Automata Studies, C. E. Shanon and J. McCarthy, eds., Princeton University Press, Princeton, NJ, 1956, pp. 43–98.
[68] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, Cambridge, UK, 2000.
[69] A. Osterloh, L. Amico, G. Falci, and R. Fazio, *Scaling of entanglement close to a quantum phase transition*, Nature, 416 (2002), pp. 608–610.
[70] T. Osborne and M. Nielsen, *Entanglement in a simple quantum phase transition*, Phys. Rev. A, 66 (2002), 032110.
[71] G. M. Palma , K.-A. Suominen, and A. K. Ekert, *Quantum computers and dissipation*, Proc. R. Soc. Lond. Ser. A, 452 (1996), pp. 567–584.
[72] A. Peres, *Quantum Theory: Concepts and Methods*, Kluwer Academic Press, Dordrecht, The Netherlands, 1993.
[73] J. Preskill, *Fault tolerant quantum computation*, in Introduction to Quantum Computation and Information, H.-K. Lo, S. Popescu, and T. P. Spiller, eds., World Scientific, River Edge, NJ, 1998, pp. 213–269.
[74] J. Preskill, *Reliable quantum computation*, Proc. Roy. Soc. London Ser. A, 454 (1998), pp. 385–410.
[75] R. Raussendorf and J. Harrington, *Fault tolerant quantum computation with high threshold in two dimensions*, Phys. Rev. Lett., 98 (2007), 190504.
[76] B. W. Reichardt, *Error-Detection-Based Quantum Fault Tolerance Against Discrete Pauli Noise*, Ph.D. thesis, University of California, Berkeley, CA, 2006. Available online at http://arxiv.org/abs/quant-ph/0612004.
[77] B. W. Reichardt, *Improved Ancilla Preparation Scheme Increases Fault-Tolerant Threshold*; available online at http://arxiv.org/abs/quant-ph/0406025 (2004).
[78] B. W. Reichardt, *Threshold for a Distance-Three Quantum Code*; available online at http://arxiv.org/abs/quant-ph/0509203 (2005).
[79] J. J. Saqurai, *Modern Quantum Mechanics*, revised ed., Addison–Wesley, Reading, MA, 1994.
[80] B. W. Schumacher and M. A. Nielsen, *Quantum data processing and error correction*, Phys. Rev. A, 54 (1996), pp. 2629–2635.

[81] A. Shamir, *How to share a secret*, Comm. ACM, 22 (1979), pp. 612–613.
[82] P. W. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM J. Comput., 26 (1997), pp. 1484–1509.
[83] P. W. Shor, *Scheme for reducing decoherence in quantum computer memory*, Phys. Rev. A, 52 (1995), pp. 2493–2496.
[84] P. W. Shor, *Fault tolerant quantum computation*, in Proceedings of the 37th Symposium on the Foundations of Computer Science, IEEE Press, Los Alamitos, CA, 1996, pp. 56–65.
[85] R. Solovay and A. C.-C. Yao, manuscript, 1996.
[86] D. A. Spielman, *Highly fault-tolerant parallel computation*, in Proceedings of the 37th Annual IEEE Conference on Foundations of Computer Science, 1996, pp. 154–163.
[87] A. Steane, *Quantum computing and error correction*, in Decoherence and Its Implications in Quantum Computation and Information Transfer, A. Gonis and P. E. A. Turchi, eds., IOS Press, Amsterdam, 2001, pp. 284–298.
[88] A. Steane, *Error correcting codes in quantum theory*, Phys. Rev. Lett., 77 (1996), pp. 793–797.
[89] A. Steane, *Multiple particle interference and quantum error correction*, Proc. R. Soc. Lond. Ser. A, 452 (1996), pp. 2551–2577.
[90] A. Steane, *Active stabilisation, quantum computation and quantum state synthesis*, Phys. Rev. Lett., 78 (1997), pp. 2252–2255.
[91] A. M. Steane, *Space, time, parallelism and noise requirements for reliable quantum computing*, Fortschr. Phys., 46 (1998), pp. 443–458.
[92] A. M. Stephens, A. G. Fowler, and L. C. L. Hollenberg, *Universal fault tolerant quantum computation on bilinear nearest neighbor arrays*, Quant. Inf. Comput., 8 (2008), p. 330–344.
[93] A. Stern, Y. Aharonov, and Y. Imry, *Phase uncertainty and loss of interference: A general picture*, Phys. Rev. A, 41 (1990), pp. 3436–3448.
[94] A. Stern, Y. Aharonov, and Y. Imry, *Dephasing of interference by a back reacting environment*, in Quantum Coherence, J. Anandan, ed., World Scientific, Singapore, 1991, p. 201.
[95] K. M. Svore, D. P. Divincenzo, and B. M. Terhal, *Noise threshold for a fault-tolerant two-dimensional lattice architecture*, Quantum Inf. Comput., 7 (2007), pp. 297–318.
[96] B. Terhal and G. Burkard, *Fault-tolerant quantum computation for local non-Markovian noise*, Phys. Rev. A, 71 (2005), 012336.
[97] W. G. Unruh, *Maintaining coherence in quantum computers*, Phys. Rev. A, 51 (1995), pp. 992–997.
[98] L. C. Washington, *Introduction to Cyclotomic Fields*, Grad. Texts in Math. 83, Springer-Verlag, New York, 1982.
[99] W. K. Wootters and W. H. Zurek, *A single quantum cannot be cloned*, Nature, 299 (1982), pp. 802–803.
[100] A. C.-C. Yao, *Quantum circuit complexity*, Proceedings of the 33th Annual Symposium on Foundations of Computer Science (FOCS), IEEE, Los Alamitos, CA, 1993, pp. 352–361.
[101] P. Zanardi, *Dissipation and decoherence in a quantum register*, Phys. Rev. A, 57 (1998), pp. 3276–3284.
[102] P. Zanardi and M. Rasseti, *Noiseless quantum codes*, Phys. Rev. Lett., 79 (1997), pp. 3306–3309.
[103] W. H. Zurek, *Decoherence and the transition from quantum to classical*, Physics Today, 44 (1991), pp. 36–44.