Towards maintaining a robust quantum memory on a lattice using local operations

Paul Fitzpatrick

MIT AI Lab

(Dated: December 11, 2002)

Abstract

Surface codes can be used to build a robust quantum memory that preserves a quantum state exponentially well as the size of the lattice grows, if the individual probabilities of errors on qubits and measurements are below a threshold [1]. However, the codes found so far require either non-local classical operations for error correction, or more spatial dimensions than are physically available. This paper examines whether the non-local classical operations can be replaced with an approximation computed in a distributed manner (using local operations), and still have a threshold result. If this computation can be coupled with the quantum operators used to maintain the memory, and performed at the same rate, they could together act as a single combined local Hamiltonian. Such an approximation seems difficult in 2D, but simpler in 1D. The bulk of this paper is devoted to developing a surface code where defects that require correction are guaranteed to occur in pairs along straight lines through the lattice, converting error correction to a 1D matching problem. A small modification to the code developed by Kitaev et al. [1, 2] discovered along the way is also presented that eliminates the need for two families of check operator, potentially simplifying implementation.

I. INTRODUCTION

Kitaev et al. have investigated the use of surface codes to construct a robust quantum memory capable of preserving a quantum state perfectly in the limit of large lattice size if the individual probability of errors on qubits and measurements is below a threshold [1, 3]. The memory has the desirable property that all quantum operations required for its maintenance are local. However the codes found so far require either non-local classical operations for error correction (2D case), or more spatial dimensions than are physically available (4D case). It is currently an open question whether this kind of code can be made to work in a completely local manner with physically realistic connectivity.

One possible way forward would be to take the 2D surface code presented in [1] (which will be referred to here as the "square/cross" code) and replace the non-local classical computation with a local, distributed computation that achieves the same effect given sufficient time. The purpose of the computation is to introduce an effective long-distance attraction between pairs of "defects" that correspond to opposite ends of a chain of erroneous qubits. If error chains grow in random-walk fashion, the expected RMS distance between their endpoints will grow with the square root of the time elapsed. Information can propagate across the lattice linearly in time, so distributed computation can at least transmit information faster than error chains accumulate. However, the force of attraction that could plausibly be simulated between a pair of defects in this way is far too weak, since once a pair separates, the noise of other defects around them effectively masks out their mutual attraction. The non-local algorithm is not sensitive to this, since it can instantaneously match and discount defect pairs belonging to smaller chains, but a distributed computation is continuously disturbed by new errors. Simulated experiments confirm that this effect destroys the exponential improvement in state preservation as the lattice grows in size.

For error chains in the square/cross code, the error syndrome information is not very informative locally. It identifies the ends of the chain, but there is no information at either end about the direction towards the other. Might some other code generate a more useful syndrome? If so, then perhaps an attractive force can be implemented between defect pairs that is less sensitive to the presence of unrelated defects nearby. For a code where operations on the encoded qubits take the form of homologically non-trivial cycles, there are limits to what we can possibly expect to determine about error chains locally. As Figure 1 illustrates,

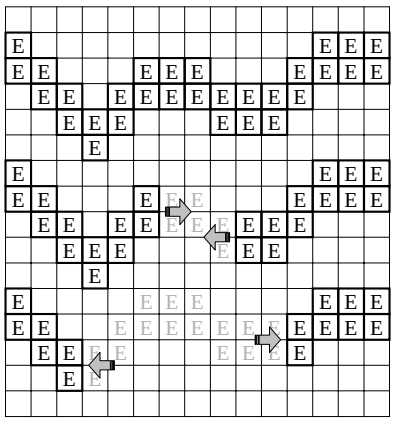

FIG. 1: Suppose the uppermost chain of E's represents a homologically non-trivial cycle on a toric lattice, implementing an encoded operation on a logical qubit. If a small fragment of the chain is removed (grayed-out portion of middle chain), error correction should work to replace it and return to the most plausible encoded state. If more than half the chain is removed (lowest chain), error correction should work to remove it completely. We cannot expect local syndrome information to determine the best direction to move in, since that depends on global knowledge (the length of the chain).

we cannot expect that local syndrome information will allow us to determine uniquely in which direction the error chain extends. There is at least one bit of uncertainty about the direction of the chain. In the square/cross code, the uncertainty is greater than this. But suppose we could find a code where the uncertainty saturates this minimum – so that the local error syndrome is sufficient to determine exactly two directions the chain might extend in – and where error chains cannot change direction without generating defects at the corner. Then we could potentially greatly constrain the attractive forces between defects to propagate along a 1D line rather than through the full 2D area. Is there any reason to believe such codes might exist? The meandering homologically non-trivial cycles that encode operations in the square/cross code can be seen as a product of trivial closed cycles (which can in turn be composed as products of the check operators) and perfectly straight homologically non-trivial cycles. If we could eliminate the trivial closed cycles, we might be able to force all non-trivial cycles to be straight.

For example, suppose we used the check operator shown in Figure 2 – something nice and awkward that makes generating trivial cycles a chore. Here we associate qubits with the

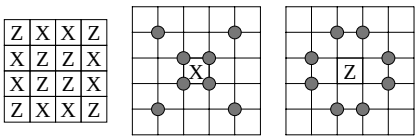

FIG. 2: An example check operator (left). The operator is associated with the vertices of a lattice, and operates on qubits in the squares of the lattice. The defects generated by X and Z errors are shown in the middle and right. A useful property is that along any horizontal or vertical line, the defects are generated in pairs and so preserve the parity of the number of defects on that line.

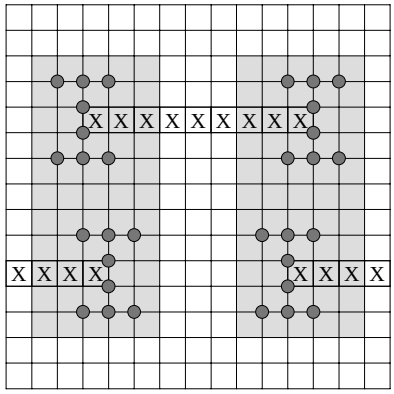

FIG. 3: The syndrome information generated by a chain of X errors is shown here. Suppose we have two such chains and wish to splice them together to form a homologically non-trivial chain that generates no syndrome information. If we choose a neighborhood around a pair of ends, such as the shaded regions shown, then we see that splicing is not possible. The parity of defects is odd along six horizontal lines within these regions. These parities can never be made even by adding X and Z operators within the neighborhood.

squares (plaquettes/faces/cells) of a toric lattice, and check operators with the vertices of this lattice. For this check operator, both X and Z errors anti-commute with even numbers of check operators horizontally and vertically. Chains of X or Z errors leave odd numbers of errors at either end of the chain. There is therefore no way to "splice" the ends of two such chains together and remove all syndrome information in the neighborhood where they meet (Figure 3), unless they are collinear. Homologically non-trivial cycles of this form that generate no defects must be straight, and cannot turn. Furthermore, the pattern of defects at either end of error chains of this form is sufficient to determine the overall direction of the chain (although not, of course, which sense it extends in).

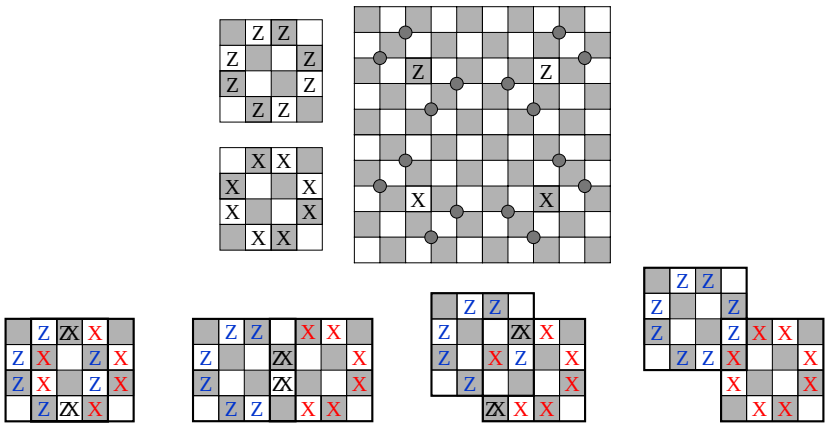

FIG. 4: The two families of check operator are shown on the upper left, shaded to show where they may appear on the lattice, which is visualized as a toroidal chessboard. Each check operator now responds to only one kind of error. The defects generated by X and Z errors are shown on the upper right. A useful property is that along any diagonal, defects are always generated in pairs. The lower row demonstrates that the check operators commute satisfactorily with each other.

For this code, chains exist (such as a chain of Y errors) that leave an even number of defects at either end of the chain and hence can be spliced. In general, there will be some chains that can turn without generating defects, and others that must remain straight. If we can encode information in a manner that is immune to chains that can turn, then we could simply ignore them. Controlling the chains that cannot turn without generating defects is likely to be much simpler than controlling more general chains, as argued earlier.

The check operator used here serves well for the purposes of illustration but in practice it has some undesirable properties. The next section introduces a more tractable version.

II. THE CHESSBOARD CODE

Consider the code defined by the check operators shown in Figure 4. The check operators come in two families, one containing only Z operators, and one containing only X operators. The check operators alternate so that neighboring vertices are associated with opposite families. There are the same number of check operators as qubits. There are two sources of redundancies between the operators. One source is due to products of the check operators taken along a diagonal in either direction, giving a total of $2L$ redundancies. This could let us encode $2L$ logical qubits, but it is unreasonable to do so since they can be affected by

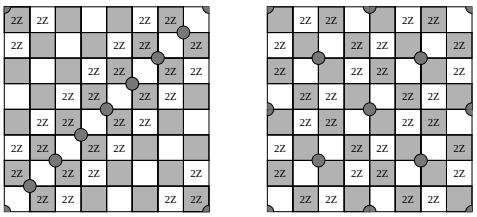

FIG. 5: The lattice width L must be even for the code to be well defined. There are $2L$ redundancies in the check operators associated with diagonals. An example of such a redundancy is shown on the left – if a product of the check operators associated with the set of highlighted vertices is taken, the result is the identity. A more useful redundancy, shown on the right, occurs when L is a multiple of four. There are a pair of redundancies of this form.

very small sets of errors. Another source is due to products of the check operators across the entire lattice, which gives an extra two redundancies (see Figure 5) if L is a multiple of four. For the remainder of this section, L is assumed to be of this form, allowing two logical qubits to be encoded robustly.

This choice for the check operators has the property that X and Z errors anti-commute with even numbers of check operators associated with the vertices along any diagonal, as Figure 4 showed. Certain diagonal chains of errors generate odd syndrome counts at either end, preventing splicing – the chain labelled Z^3 in Figure 6 is a good example. A chain like Z^3 but in the perpendicular direction is possible if it lies on black squares. And there are analogous chains in X which go in the opposite direction to the Z chain on the same colored squares. The chain Z^1 also cannot be spliced (unless it is overlaid with a shifted copy of itself, Z^2). We could hope to prevent such chains from occurring using relatively simple defect matching along the diagonals. Unfortunately, Z^3-style chains extended to be cycles cannot be made to anti-commute with X versions of themselves (they intersect on an even number of squares), and Z^1-style chains can be masked as in Z^2. So we cannot choose a pair of such cycles as the encoded operations on the same qubit. Even if this were possible, other uncontrolled cycles could still anti-commute with them and cause mischief.

But while we cannot protect a qubit completely, we can potentially protect it from either a bit or phase error (but not both). For example, suppose we choose our encoded operation $\bar{X}_{safe}$ on the first logical qubit to be the unusual cycle X^7. Cycles in Z that anti-commute with this small cycle all correspond to the kinds of chains we can control with 1D matching.

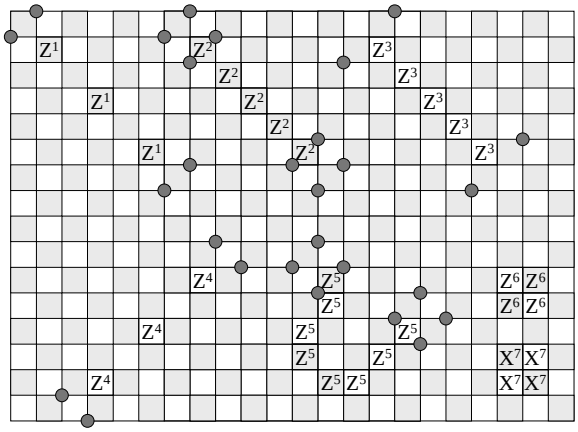

FIG. 6: Several error chains and the defects they generate. Chain Z^1, the chain identified with the superscript 1 on its errors, leaves defects at either end that are odd in the diagonals parallel to the chain. Two such chains can overlap (Z^2) to leave even defects on the diagonals, and we will see this chain can turn. Z^1 and Z^2 lie on black squares; on white squares a chain in the same orientation (Z^3) has no gaps, like Z^2, but leaves odd defects (like Z^1). In the opposite orientation (Z^4) the chain behaves just like Z^1. It has gaps, and can be overlapped and hence can turn (Z^5). Finally, there are very small cycles (Z^6, X^7) that commutes with all the check operators but are independent of them. This can be seen as a by-product of the family of redundancies in the check operators along diagonals.

Hence we can prevent the operation $\bar{Z}_{safe}$ from occurring (of course, $\bar{X}_{safe}$ itself cannot be prevented and requires only four errors to occur). We are clearly free to choose $\bar{X}_{safe}$ in this way since we could put the check matrix for the code in standard form [4, 5], solve for a full set of $\bar{Z}$ and $\bar{X}$ operators, and then find some $\bar{Z}_i$ with which $\bar{X}_{safe}$ anti-commutes. Then we can replace $\bar{X}_i$ with $\bar{X}_{safe}$, and every $\bar{Z}_k$ that also anti-commutes with $\bar{X}_{safe}$ with $\bar{Z}_i\bar{Z}_k$ (other than $\bar{Z}_i$ itself). As a sanity check, I tried this procedure for a 16×16 lattice, and it went through fine (although not every choice of X^7-like cycle anti-commutes with a $\bar{Z}_i$; it is useful to choose a diagonal cycle $\bar{Z}_{safe}$ that anti-commutes with $\bar{X}_{safe}$ and install it first in the same manner before installing $\bar{X}_{safe}$ itself).

This is an encouraging result. Using purely local error correction procedures, Dennis et al. showed that a full qubit could be protected in a 4D lattice, while in a 3D lattice a code could only be found to protect against either bit or phase flips (but not both) [1]. The current result suggests that this latter kind of protection is possible in a 2D lattice if we are

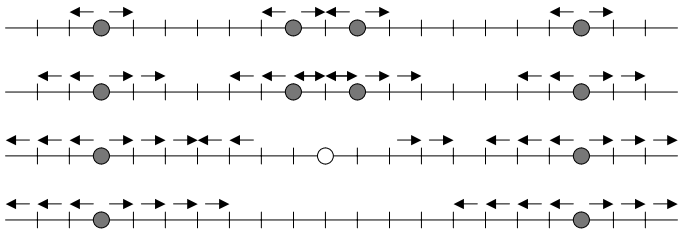

FIG. 7: Error correction on a line. Filled circles represent defects, the empty circle represents a successfully annihilated pair. Each of the four lines represents a successive iteration. Arrows are defect messages being transmitted up and down the line. When these messages lose support, they are destroyed at twice the rate that they propagate.

willing to perform error correction along 1D lines extending out from defects. Perhaps then in a 3D lattice with error correction in a similarly constrained geometry, full protection of a qubit is possible? This is pure speculation of course, but at an intuitive level the quest for full protection in the 2D case is plagued with problems related to geometric intersections – which are clearly sensitive to the physical dimension.

Figure 7 gives a sketch of how error correction can be done along a line (such as the diagonal lines of the chessboard code proposed in this section). Binary message flags declaring the presence of a defect are transmitted outwards at some rate r; messages which lack the support of messages behind them are destroyed at a faster rate. When these messages reach defects, they induce motion towards their source (at a slower rate than r), using the appropriate operators on local qubits. If messages are received from both directions, no motion results. When defects meet, they annihilate. While Figure 7 shows a simple case, the key case to consider is when a pair of defects are far from each other and have been so for a long enough time to dominate their neighborhood with messages, but not long enough for those messages to reach each other. All new, close defects outside the pair will tend to get drawn to them, which is why it is important their messages reach each other faster than they move so they can annihilate before adding on to the error chain. It doesn't matter what new defects appearing inside the pair do – they may be frozen, or move towards one of the pairs, or annihilate. The lattice needs to be larger than it would be for instantaneous non-local error correction, so that the largest chains that correction is intended to cope with have time to discover each other and annihilate before their messages loop back around the torus. But this is just a constant scaling factor in the width of the lattice. This scheme is

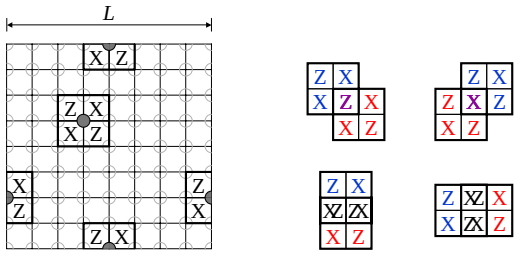

FIG. 8: $L \times L$ qubits are associated with the squares of a toric lattice, with check operators associated with the vertices (circled). Two example instances of the check operator are shown on the left. On the right, overlapping check operators are shown to commute satisfactorily.

inherently robust to sporadic false errors in the syndrome, but caution is required to deal with missed errors. The messages themselves need to be robustly propagated, since they support each other.

III. THE DIAGONAL CODE

This section is somewhat of a tangent to the rest of the paper. It presents an alternative form of the square/cross code used by Kitaev et al. which is particularly regular, and might offer some advantages for implementation. It was found while searching for codes with constrained error chains, which it unfortunately does not have.

Consider an array of $L \times L$ qubits associated with the squares of a toric lattice, as shown in Figure 8. With each vertex of the lattice we can associate the 2×2 check operator shown. Each instance of the operator commutes with every other. There are the same number of operators as qubits. If the width of the lattice is even, then two of the check operators can be expressed as a product of others, as shown in Figure 9. These are the only redundancies, so there are a total of $L^2 - 2$ independent commuting operators. This is two less than the number of qubits, so there are two encoded logical qubits.

The properties of this code are analogous to that of the square/cross code. Defects are created in pairs, and can be separated from each other along a chain (see Figure 10). Closed cycles that are homologically trivial can be shrunk until they disappear, and so clearly have no effect on the encoded state. Cycles that are homologically non-trivial commute with the check operators but are not in the stabilizer they define. Such cycles operate on the encoded state. Figure 11 shows encoded operations suitable for acting on the encoded qubits.

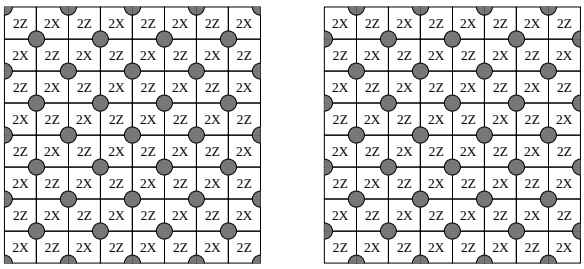

FIG. 9: Redundancies in the check operators. If the operators associated with the vertices circled in either lattice are multiplied together, the result is the identity operator. Hence any one of the operators could be expressed as a product of the remainder in the same set. There are two redundancies if the width of the lattice is even, otherwise there is just one.

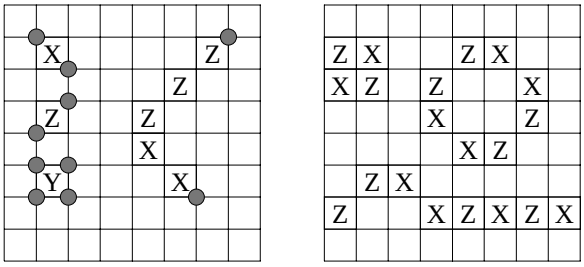

FIG. 10: The behavior of defects is analogous to the square/cross code. They are created in pairs, and can be separated along a chain of errors (shown on left). If defects are brought together, they annihilate, leaving either a homologically trivial cycle that is a product of the check operators, or a homologically non-trivial cycle that is not (shown on right).

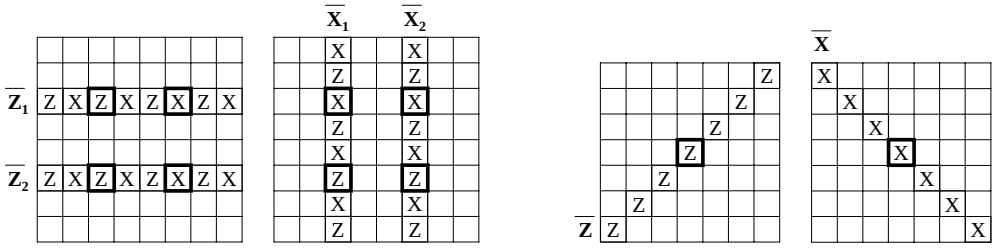

FIG. 11: Encoded operators for two logical qubits (left). The qubits where the encoded $\bar{X}$ and $\bar{Z}$ operators intersect are highlighted. For $\bar{Z}_1$ and $\bar{Z}_2$ with a Z operator leftmost as shown, it is important that one be in an even-numbered row and one be in an odd-numbered row (if not, one needs to start with an X instead). The same applies to the columns of $\bar{X}_1$ and $\bar{X}_2$. The encoded operators on the right apply when the lattice width L is odd and there is a single encoded qubit.

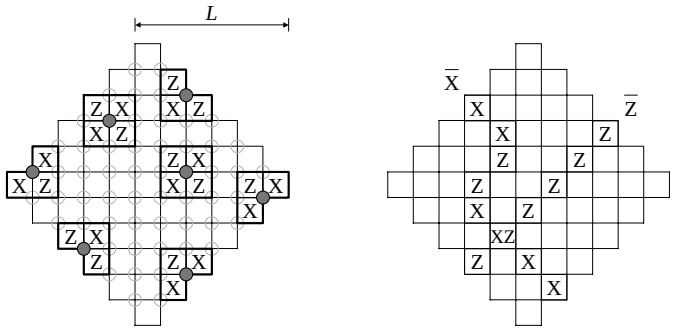

FIG. 12: Planar version of the diagonal code. The border needs to be a staircase, as shown on the left (although it need not be as regular as shown). There are now check operators along the border that act on three qubits rather than four. Edges can be classed as "X receptive" or "Z receptive" (analogous to the rough/smooth distinction in the square/cross code [2]). Defects can now appear singly, from chains terminating in X or Z at an appropriate edge. Encoded operators arise as chains reaching between corresponding edges.

Error correction can follow essentially the same pattern as for the dual-operator code. Pairs of errors can be matched up using the Edmonds algorithm along diagonals, and corrected using the shortest chain between them. Importantly, consideration of whether to apply a correction in X or Z is not required to choose that path – these can be read off trivially from the shape of the correcting path (diagonals in one direction require a Z, diagonals in the other require an X, straight lines alternate X and Z, and the defects at either end of the path determine whether an X or Z is required there).

In the same way as the square/cross code, the diagonal code can be made planar rather than toroidal, as shown in Figure 12. The shape of the border is crucial. One feasible choice is the diamond shape shown. The check operators are as before – unless the corner of an operator lies outside the border, in which case that corner is simply omitted. Operators are associated with every non-trivial vertex in the grid. There are $L^2 + (L-1)^2 = 2L(L-1)+1$ qubits, and $2L(L-1)$ check operators. There is one more qubit than check operator, so there is a single encoded qubit – again, like the planar version of the square/cross code.

It is still possible to interpret defects as anyonic excitations [6–8] for this code, if we associate a chessboard pattern with the *vertices* of the lattice. Defects at white-colored vertices and defects at black-colored vertices have the appropriate Aharonov-Bohm style interaction where moving one defect around the other introduces a global phase.

This code has the advantage of requiring just a single type of check operator, rather than two, so the syndrome measurement is extremely regular. This simplification to the quantum component of the error correction process may be useful in practice. Error correction can also be done in one pass for all errors, rather than in two iterations for X and Z errors separately – but this is not particularly significant, since the vertices segregate out into two non-interacting sets anyway.

IV. CONCLUSIONS

The various check operators presented in this paper are a small subset of those possible, particularly if X and Z operators are mixed. While all the surface codes defined on a 2D lattice have much in common, particular codes can have technical advantages such as the chessboard code presented in Section II. It is interesting that even the simple diagonal code lies outside the formalism developed by Freedman and Meyer [9] which otherwise makes surface codes much easier to develop. There appears to be quite a lot of room for exploration in this area, and Raginsky has argued that the same is true for physical implementation [10].

[1] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, J. Math. Phys. **43**, 4452 (2002), quant-ph/0110143.

[2] S. B. Bravyi and A. Y. Kitaev (1998), quant-ph/9811052.

[3] C. Wang and J. Preskill (2002), quant-ph/0207088.

[4] R. Cleve and D. Gottesman, Phys. Rev. A. pp. 76–82 (1997), quant-ph/9607030.

[5] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, 2000).

[6] A. Y. Kitaev (1997), quant-ph/9707021.

[7] M. H. Freedman, A. Kitaev, M. J. Larsen, and Z. Wang, Bull. AMS (to appear) (2002), quant-ph/0101025.

[8] P. Zanardi and S. Lloyd (2002), quant-ph/0208132.

[9] M. H. Freedman and D. A. Meyer (1998), quant-ph/9810055.

[10] M. Raginsky (2001), quant-ph/0111035.