ANALYSIS OF NOISY ENTANGLING GATES IN MINIMAL SURFACE CODES

ANDREAS PETER

SUPERVISORS: JAMES WOOTTON AND DANIEL LOSS

Abstract. In order to build a useful quantum computer, the problem of noise has to be tackled. A promising way to get noise under control is the use of surface codes. The smallest useful surface code is $\mathcal{S}_{17}$, which we will analyse in this work. Using tensor network methods, we analyse the connection between the fidelity of the CNOT-gates in a code with the performance of the code itself for one round of error correction. Two scenarios were calculated for different CNOT inspired by quantum dot implementations, one starting in a product state with one measurement round and final measurement and one starting in a stabilizer state with one measurement round. For purely decoherent noise we find that the code performance can be predicted well using the channel fidelity for both scenarios. For more complicated unitary gates that approximate a CNOT and are derived from floating gate implementations we find that for very small intervals of channel infidelity there is a large spread in code performance. This shows that we can not predict the performance of a surface code that uses specific entangling gates well if we only know the channel infidelity of those entangling gates.

Contents

Date: May 2017.

1. Introduction

Quantum computers — computers that are built to *use* quantum effects in computations — are fascinating machines predicted to offer as of yet unachieved performance in specific mathematical problems such as factorization [34], promising to break most of our current methods of encryption or modest but remarkable speed-ups for more general problems such as searching unstructured data [15], which can be generalized to finding inversions of function-value pairs.

An obstacle in building a useful quantum computer that has yet to be overcome is the effect of the outside world on the computer which presents itself as *noise*.

Noise can change the state of the computer in such a way that it renders it practically useless, destroying the quantum properties of the system.

Protecting the system from outside disturbance is thus of central importance to the field. A proposed solution is the use of *error correcting codes*, i.e. rules that specify how to encode a state carrying information in a system in such a way, that it can be recovered even if there is some noise acting on the system.

Surface codes are the most prominent members of the error correcting codes due to their good performance in simulations and reliance on only nearest-neighbor interactions.

Surface codes have been analysed in various different settings by means of simulation. In this work we want to expand on that analysis with a focus on the two-qubit entangling gates that are used.

We will look at whether the performance of a surface code is linked to the performance of the two-qubit gates it uses or not for the smallest useful code: $\mathcal{S}_{17}$. Since already the 17 qubits used in $\mathcal{S}_{17}$ make an exact simulation impractical on consumer hardware, we will rely on *tensor networks* as a tool to represent the code and calculate the quantities of interest.

In section 2 we will introduce four postulates of quantum mechanics that are sufficient for the rest of the thesis. This introduction includes mention of the necessary resources for doing calculations in quantum mechanics which we will use later and presents the $\mathcal{O}$-notation to quantify the resource use in a practical way. Given these basics, we can sketch the idea of quantum computation and establish requirements for its realization.

Afterwards, in section 3, we will look at tensor networks, their uses and advantages, and a graphical language used thereafter for quantum mechanics in terms of tensor networks. The postulates of section 2 are repeated in the graphical language to serve as an example of its usage.

After introducing open quantum systems in section 4, we can look at the effects of a system being open by considering outside influences as noise in section 5.

To fight noise, section 6 presents the basics of quantum error correction and introduces the repetition and surface code. Among the surface codes we single out a particularly promising code for early experimental realization, $\mathcal{S}_{17}$, which we investigate in detail.

The physical platform our calculations are inspired by is that of lateral quantum dots presented in section 7, where three different implementations of the two-qubit gate for $\mathcal{S}_{17}$ are presented: two based on electrostatic interaction, one based on tunneling.

Having set up all the ingredients, we explicitly cast $\mathcal{S}_{17}$ with the entangling gates derived in section 7 into a tensor network in section 8, where we also establish how we calculate the gate fidelities. This enables us to look at the performance of $\mathcal{S}_{17}$ for different two-qubit gates in subsection 8.2.

The results are interpreted in a larger context in section 9.

2. Quantum Mechanics

We can present the core of quantum mechanics in terms of four postulates [25]:

Postulate 1: The state of an isolated physical system is completely described by its *state vector*, a complex unit vector in the *Hilbert space*. A Hilbert space is a complex vector space with inner product, associated with the physical system.

Postulate 2: Evolution in isolated physical systems is described by *unitary* transformations $\boldsymbol{U}$, i.e. transformations that conserve the inner product between state vectors. The state vector $\boldsymbol{\psi}$ of a system at time t is related to the state vector of the system $\boldsymbol{\psi'}$ at time t' by

$$\boldsymbol{\psi'} = \boldsymbol{U}\boldsymbol{\psi}. \tag{1}$$

Postulate 3: Measurements are described by sets of *measurement operators* $\boldsymbol{M}_m$, where each $\boldsymbol{M}_m$ is an operator on the Hilbert space of the system being measured. Each index m corresponds to a measurement outcome whose probability is given by

$$p(m) = \langle \boldsymbol{\psi} | \boldsymbol{M}_m^\dagger \boldsymbol{M}_m \boldsymbol{\psi} \rangle \,, \tag{2}$$

where $\boldsymbol{\psi}$ is the state being measured and $\langle \,|\, \rangle$ is the inner product on the Hilbert space.

The state of the system after the measurement is

$$\frac{\boldsymbol{M}_m \boldsymbol{\psi}}{\sqrt{p(m)}}. \tag{3}$$

The measurement operators satisfy the *completeness equation*,

$$\sum_m \boldsymbol{M}_m^\dagger \boldsymbol{M}_m = 1, \tag{4}$$

implying that a measurement must have an outcome.

Postulate 4: The Hilbert space of a composite system is the tensor product of the Hilbert spaces of its component systems. This structure transfers to the state vectors, i.e. the state vector of the composite system can be written as a sum over tensor products of the components' state vectors.

From Postulate 1 it follows that for a given state space we can find a basis since the Hilbert space is simply a vector space with additional structure. Thus we can expand state vectors and unitary operators acting on them in a basis.

Unitary operators, now expanded as matrices in the basis of the state vectors, obey

$$\langle Uv, Uw\rangle = \langle v, U^\dagger U w\rangle \stackrel{!}{=} \langle v, w\rangle. \tag{5}$$

Due to the spectral theorem, U is diagonalizable, which means that for a suitable basis — the basis of eigenstates of U — U is a diagonal matrix with entries of the form $\exp(\mathrm{i}\phi)$, where ϕ is a real phase.

Since U is diagonalizable, we can write any U as $\exp(-\mathrm{i}Ht)$ where H can be diagonalized to yield exactly the phases ϕ of U on its diagonal, i.e. H is a matrix with real eigenvalues: A Hermitian matrix.

Thus every unitary evolution can be written in terms of a Hermitian operator H which we call the Hamiltonian. We can now define the eigenstates of the Hamiltonian as the eigenstates of the system and identify the eigenvalues of the eigenstates with their energy.

The Hamiltonian defines how the unitary operators U from Postulate 2 act on the system and thus gives us a complete description of its behaviour. For finite systems we can simply expand states and Hamiltonians in a basis of the system, exponentiate the Hamiltonian to get the unitary operator for time evolution and directly calculate how a system evolves by simple matrix multiplication, or how a system behaves at low energies by looking at its ground state, i.e. the lowest energy eigenstate.

2.1. **Curse of Dimensionality.** While the description above enables us to calculate all desired quantities of a small system rather easily, problems appear when the system size increases.

Generally, the number of basis states and thus of coefficients we have to keep track of grows exponentially as d^N, where N is the number of composite d-level systems, due to the tensor-product structure of composite systems.

Already for small systems with $d = 2$ and $N \approx 200$, writing down states or operators becomes impossible as both practical — computer memory, paper on earth — and theoretical — atoms in the universe — resources are not sufficient for the task.

This is the *curse of dimensionality*: With growing system size the parameter space grows exponentially, rendering the above approach completely useless for systems above a certain threshold but way below realistic systems.

2.2. **The Cost of Doing Physics.** Above in subsection 2.1 we have introduced a notion of *cost* of saving a state as the number of complex numbers we need to write down. In doing so, we ignored the actual way of saving complex numbers and the details of any computer actually doing so, because the details do not seem too important, the scaling is.

We borrow a concept from computer science to characterize requirements we have for *computational resources*, namely time and space, in terms of *asymptotic notation*.

We will use $\mathcal{O}$-notation or *big oh*-notation [2]. $\mathcal{O}(g(n))$ describes a class of functions that are upper bound by $cg(n)$ for some constant $c > 0$ for all $n > n_0 > 0$. Formally:

$$\mathcal{O}(g(n)) = \{\, f(n) \mid \exists c, n_0 \colon 0 \leq f(n) \leq cg(n)\ \forall n \geq n_0 \,\}. \tag{6}$$

If a function belongs to $\mathcal{O}(g(n))$, its behaviour for large n is upper bound by $cg(n)$ for some c and we also assume that it is well characterized by that bound.

We use this to describe the computational requirements of doing physics. We say saving a state requires e.g. $\mathcal{O}(d^N)$ spatial resources, meaning that the cost — at least for large N — is upper bound by some multiple of d^N or that we need exponentially many resources (in N)..

For the rest of this thesis, we will use asymptotic notation to characterize and compare the space requirements to describe states in different representations and time requirements of calculations.

An important distinction is that between *efficient* and *intractable*. We say something is intractable if there is no algorithm whose resources are in $\mathcal{O}(N^k)$ where k is an arbitrary coefficient, i.e. a polynomial, and efficient otherwise. The straightforward representation of a system of N d-level subsystems in a basis of the Hilbert space is intractable in N.

2.3. **Quantum Computation.** In the 1990s efficient algorithms were discovered that used quantum effects to solve problems previously thought to be intractable or offer a significant speedup over all known algorithms so far. The two most prominent of those algorithms deal with integer factorization [34] and searching of unstructured databases [15].

Computational machines that use quantum effects to implement aforementioned algorithms are *quantum computers*.

For quantum computing, information can be considered to be encoded into *two-level quantum systems* as *qubits*. Even if the physical implementation does not use qubits, a description in terms of qubits is always possible.

Qubits are to quantum computing what bits are to classical computing: the basic carriers of information. While a bit has two possible states and occupies either of those with certainty – although we might lack the knowledge which – the qubit can be in a superposition of states.

We can express qubits in the *computational basis* $|0\rangle$,$|1\rangle$ or in any rotation of that basis, e.g. the σ_{x} eigenstates $|+\rangle$ and $|-\rangle$. The coefficients of the two basis states are complex numbers constrained by the state having to be a unit vector. Qubits can thus be described in terms of three real parameters.

A general algorithm on a quantum system corresponds to a unitary operator since it takes a state vector for the input to a state vector for the output. To build a quantum computer, we must be able to implement such unitaries in a system. While the fact that unitaries are continuous as opposed to classical logical circuits might imply that we have to implement arbitrary many quantum operations for a quantum computer, it turns out that [25]

(1) Arbitrary unitary operators can be decomposed *exactly* into two-qubit operators.
(2) Arbitrary two-qubit operators can be decomposed *exactly* into single-qubit operators and CNOT gates.
(3) Arbitrary single-qubit operators can be approximated with arbitrary precision using a limited set of gates.

Thus a quantum computer need only be able to apply a limited set of single-qubit operators and a CNOT gate, which is written as

$$\mathrm{CNOT}_{\mathrm{c,t}} = |0\rangle\langle 0|_{\mathrm{c}} \otimes \mathbb{1}_{\mathrm{t}} + |1\rangle\langle 1|_{\mathrm{c}} \otimes \sigma_{\mathrm{x,t}}, \tag{7}$$

where c and t stand for *control* and *target*. The set of single qubit operators required is not unique but e.g. the *Hadamard* gate H and the $\frac{\pi}{8}$ gate T, defined as

$$H = |+\rangle\langle 0| + |-\rangle\langle 1| \tag{8}$$

and

$$T = |0\rangle\langle 0| + \exp(\mathrm{i}\tfrac{\pi}{8})|1\rangle\langle 1|, \tag{9}$$

together with a CNOT gate constitute a set of gates that is *universal* for quantum computation, i.e. all unitaries can be approximated arbitrarily well using only such gates.

In 2000, DiVincenzo formulated five requirements for the *physical implementation of a quantum computer* [5], that take the requirement of scalability into account. These five requirements, that have become known as *DiVincenzo's criteria*, are (as taken from the original paper):

(1) A scalable physical system with well characterized qubits
(2) The ability to initialize the state of the qubits to a simple fiducial state, such as $|000\ldots\rangle$
(3) Long relevant decoherence times, much longer than gate operation time
(4) A universal set of quantum gates
(5) A qubit-specific measurement capability

3. Tensor Network Theory

3.1. **Tensor.** For this thesis, a tensor can be thought of as a multidimensional array of complex numbers. Tensors are written as A_{ijk} where A is the name of the tensor and i,j and k are its indices. If the indices are fixed, e.g. A_{123}, the object is simply a complex number. We follow [28] and call the number of indices the *valence*, e.g. A_{ijk} is a valence-3 tensor, instead of the more common *rank*, since the rank of a matrix is an invariant, while the valence, as used here, is arbitrary. Valence-0, valence-1 and valence-2 tensors correspond to scalars, vectors and matrices respectively.

The indices all have a limited domain and we denote the size of that domain ξ_i for index i and call it the *dimension* of i. Thus A_{ijk} has $\xi_i\xi_j\xi_k$ entries.

To manipulate tensors we will use five operations: Taking the product between tensors, contracting tensors, grouping and splitting indices, and the singular value decomposition.

3.1.1. *Product.* The product of two tensors A_{ij}, B_k is simply the tensor product, i.e.

$$C_{ijk} = A_{ij}B_k. \tag{10}$$

Note that the resulting tensor has

$$\text{valence}(C) = \text{valence}(A) + \text{valence}(B). \tag{11}$$

To explicitly evaluate a product of tensors, e.g. constructing the tensor C_{ijk} above, we have to calculate all entries of the resulting tensor. The entries are products of entries of the tensors involved. Thus the cost of evaluating the product above is $\mathcal{O}(\xi_i\xi_j\xi_k)$, since for each of the $\xi_i\xi_j\xi_k$ entries of C a product has to be evaluated.

In general, the cost of a product of tensors is proportional to the product of the dimensions of the tensors involved.

3.1.2. *Contraction.* To contract two tensors A_{ij}, B_k, an index of each tensor with equal dimension has to be specified. Given those indices, we take the product of the two tensors $A_{ij}B_k$ and set the two specified indices, e.g. j and k with $\xi_j = \xi_k$, equal and sum over the — now common — index of the two tensors.

This results in a new tensor

$$C_i = \sum_{l=1}^{\xi_l} A_{il}B_l. \tag{12}$$

We can often omit writing the sum and, following the Einstein summation convention, sum over repeated indices in an expression, i.e.

$$C_i = A_{il} B_l. \tag{13}$$

Note that above

$$\text{valence}(C) = \text{valence}(A) + \text{valence}(B) - 2, \tag{14}$$

i.e. the valence of the resulting tensor is decreased by the number of indices that are contracted.

The resulting tensor C_i has ξ_i entries and for each entry we have to sum over ξ_l products of entries in A and B. In general, the cost of a tensor contraction over an index l with only index i in the result (an assumption we can make due to grouping, see below), is $\mathcal{O}(\xi_i \xi_l)$.

This also implies that evaluating the tensor product explicitly and then summing over the common index is a bad strategy, increasing the cost to $\mathcal{O}(\xi_i \xi_l^2)$.

A special case of contraction is the *trace*, a contraction of a tensor with itself. Prototypical is the matrix-trace which, for a rank-2 tensor or matrix A_{ij} reads

$$C = A_{ii}, \tag{15}$$

where C is a valence-0 tensor. The trace has a cost $\mathcal{O}(\xi_i)$, being only a sum over ξ_i entries.

3.1.3. *Grouping and Splitting.* Grouping and splitting are methods to change the indices of a tensor. Grouping refers to the grouping of multiple indices into one index, such that the valence decreases. E.g. introducing an index $m = i + j\xi_i$, we can write A_{ij} as A_m with $\xi_m = \xi_i \xi_j$.

If we want to show explicitly that a tensor has grouped indices, we can enclose them in parentheses, writing A_m as $A_{(ij)}$.

Splitting refers to the inverse operation, namely taking an index and separating it into two or more indices. Reversing the grouping above, we define

$$i = m \mod \xi_i \tag{16}$$

$$j = \left\lfloor \frac{m}{\xi_i} \right\rfloor, \tag{17}$$

to transform A_m back to A_{ij}.

The assignment above reverses the grouping and more generally allows to split an index into an arbitrary number of indices with the constraint that the overall number of entries — the product of all dimensions — is conserved. Splitting all indices to two-dimensional indices allows representation of arbitrary states in terms of two-level systems.

Since grouping and splitting merely change the representation of a tensor, not its entries, we take the cost of those operations to be $\mathcal{O}(1)$.

3.1.4. *Singular Value Decomposition.* The *singular value decomposition* (SVD) is a decomposition of a matrix A into unitary matrices U,V and a diagonal matrix D, such that

$$A = UDV^\dagger. \tag{18}$$

The matrix D contains the *singular values* on its diagonal which are real numbers greater than zero and writing $D = \text{diag}(\lambda_1, \lambda_2, \ldots)$, we assume the singular values λ_i to be ordered, such that $\lambda_i \geq \lambda_j$ iff $i < j$.

Since we can group and split the indices of a tensor, we can always choose the grouping and splitting such that we get a valence-2 tensor which we can treat as a matrix and thus apply an SVD on.

E.g. a tensor C_{ijk}'s indices might be grouped as $C_{(ij)k}$ — resulting in a valence-2 tensor. We can then apply an SVD to get matrices U,V and D, such that

$$C = UDV^\dagger. \tag{19}$$

Looking at the matrices as tensors again, we get

$$\begin{aligned} C_{(ij)k} &= U_{(ij)l} D_{ll'} (V_{kl'})^\dagger \\ &= U_{(ij)l} D_{ll'} V^*_{l'k} \end{aligned} \tag{20}$$

and splitting (ij) on both sides yields

$$C_{ijk} = U_{ijl} D_{ll'} V^*_{l'k}, \tag{21}$$

which is a decomposition of C in terms of two tensor contractions over three tensors. The cost associated with the SVD of an m by n matrix is $\mathcal{O}(\min(mn^2, m^2n))$ [17].

The SVD is not only useful to decompose a tensor into multiple tensors, its main value stems from it being useful for *low matrix-rank approximation*.

Taking the Frobenius Norm as a distance measure, the SVD enables us to approximate a matrix C by *trimming the singular values* and this approximation is ideal as shown by Eckart and Young [7].

Explicitly: Given a matrix $C = UDV^\dagger$, we can approximate it by setting all but the k largest singular values in D to zero, getting a matrix $C^{[k]} = UD^{[k]}V^\dagger$. The distance between C and $C^{[k]}$ is minimal in the Frobenius norm over all possible rank-k matrices.

This result also translates to tensors. To approximate a tensor, we simply follow the above procedure to apply an SVD and then trim the singular values, contract only up to some specified index k to get an approximation with lower *dimension* of the connecting indices and thus less entries.

In the case of Equation 21, we can truncate the summation over l, l' by setting the smallest singular values to zero and get a new, approximate description:

$$C^{[t]}_{ijk} = U_{ijm} D^{[t]}_{mm'} V^*_{m'k}, \tag{22}$$

where $\xi_m = \xi_l - t$, so U, D and V have less entries than before yet still represent or optimally approximate the same tensor C.

3.2. **Quantum Mechanics With Tensors.** Here we revisit the ingredients of quantum mechanics, sketched in section 2, and formulate them in terms of tensors.

3.2.1. *States with Tensors.* Consider a state $\boldsymbol{\psi}$ in a d-dimensional Hilbert space. Using the basis of $N = \log_2(d)$ spin-$\frac{1}{2}$ particles, i.e. a basis $|i_1 i_2 \cdots i_N\rangle = |i_1\rangle \otimes |i_2\rangle \otimes \cdots \otimes |i_N\rangle$ where $i \in \{0, 1\}$ we can expand $\boldsymbol{\psi}$ as

$$\boldsymbol{\psi} = \underbrace{\langle i_1 i_2 \cdots i_N | \psi \rangle}_{\psi_{i_1 i_2 \cdots i_N}} |i_1 i_2 \cdots i_N\rangle, \tag{23}$$

where summation over repeated indices is implied.

The *tensor* $\psi_{i_1 i_2 \cdots i_N}$ contains all information about our state but requires $\mathcal{O}(2^N)$ complex numbers to be specified, i.e. exponential resources in terms of the system size N.

We proceed by *grouping* indices i_2 to i_N, getting a valence-2 tensor $\psi_{i_1 (i_2 \cdots i_N)}$. Using the SVD, we can write the state above as

$$\boldsymbol{\psi} = A^{(1)}_{i_1 \alpha_1} s^{(1)}_{\alpha_1 \alpha'_1} \psi_{\alpha'_1 (i_2 \cdots i_N)} |i_1 i_2 \cdots i_N\rangle, \tag{24}$$

where A and ψ are unitary matrices and s is the diagonal matrix with singular values on its diagonal and $A^{(1)}_{i_1 \alpha_1} s^{(1)}_{\alpha_1 \alpha'_1} \psi_{\alpha'_1 (i_2 \cdots i_N)} = \psi_{i_1 (i_2 \cdots i_N)}$. Using that s is diagonal and has $\xi^\star_{\alpha_1} \leq \xi_{\alpha_1}$ nonzero singular values, we can set $\alpha_1 = \alpha'_1$ and run the sum up

to $\xi^\star_{\alpha_1}$, i.e. truncate the involved tensors' dimensions. For further convenience we can absorb $s^{(1)}$ into $A^{(1)}$ by contracting the two tensors.

We end up with

$$|\psi\rangle = A^{(1)}_{i_1\alpha_1}\psi_{\alpha_1(i_2\cdots i_N)}|i_1 i_2\cdots i_N\rangle. \tag{25}$$

To specify $|\psi\rangle$ we need $2\xi^\star_{\alpha_1}$ and $\xi^\star_{\alpha_1}2^{N-1}$ complex numbers for A and ψ respectively with $\xi^\star_{\alpha_1} \leq 2$ since $\xi^\star_{\alpha_1}$ is upper bound by the smaller of the dimensions of $\psi_{i_1(i_2\cdots i_N)}$.

The indices of tensors that are not associated with a basis element, e.g. α_1, are called *virtual* indices. In contrast, those that are associated with a basis element such as i_1 are called *physical* indices.

Repeating the above procedure, we can split the indices of $\psi_{\alpha_1(i_2\cdots i_N)}$ to get $\psi_{\alpha_1 i_2\cdots i_N}$ and group the indices to get the valence-2 tensor $\psi_{(\alpha_1 i_2)(i_3\cdots i_N)}$. Application of an SVD as above leads to

$$|\psi\rangle = A^{(1)}_{i_1\alpha_1}A^{(2)}_{(\alpha_1 i_2)\alpha_2}\psi_{\alpha_2(i_3\cdots i_N)}|i_1 i_2\cdots i_N\rangle, \tag{26}$$

where again we absorbed the singular value matrix $s^{(2)}$ into $A^{(2)}$ and sum over α_2 up to the number of nonzero singular values $\xi^\star_{\alpha_2}$. Splitting the indices of $A^{(2)}$, we get

$$|\psi\rangle = A^{(1)}_{i_1\alpha_1}A^{(2)}_{\alpha_1 i_2\alpha_2}\psi_{\alpha_2(i_3\cdots i_N)}|i_1 i_2\cdots i_N\rangle. \tag{27}$$

Now, we need to specify three tensors to describe $|\psi\rangle$, namely $A^{(1)}$, $A^{(2)}$ and ψ with $2\xi^\star_{\alpha_1}$, $2\xi^\star_{\alpha_1}\xi^\star_{\alpha_2}$ and $\xi^\star_{\alpha_2}2^{(N-2)}$ entries respectively.

Following the above recipe N times allows us to split off N tensors $A^{(i)}$ that carry a physical index each and, depending on whether they are in the bulk or at the boundary, one or two virtual indices α_i:

$$|\psi\rangle = A^{(1)}_{i_1\alpha_1}A^{(2)}_{\alpha_1 i_2\alpha_2}A^{(3)}_{\alpha_2 i_3\alpha_3}\cdots A^{(N-1)}_{\alpha_{N-2}i_{N-1}\alpha_{N-1}}A^{(N)}_{\alpha_{N-1}i_N}|i_1 i_2\cdots i_{N-1}i_N\rangle. \tag{28}$$

Taking $\xi = \max_i \xi^\star_{\alpha_i}$, the associated cost of the state in memory is $\mathcal{O}(2N\xi^2)$, since we have to specify N tensors with $\mathcal{O}(2\xi^2)$ entries each. Ignoring possible N-dependence of ξ, this only scales *linearly* in the system size.

But in general, following the argument for $\xi^\star_i$ scaling above, ξ is $\mathcal{O}(2^N)$: an indication that the procedure is not useful for just *any* state.

What if we limit ξ? For the extreme case $\xi = 1$, we recover all product states $|\psi\rangle$ with

$$\begin{aligned}|\psi\rangle &= A^{(1)}_{i_1}A^{(2)}_{i_2}A^{(3)}_{i_3}\cdots A^{(N-1)}_{i_{N-1}}A^{(N)}_{i_N}|i_1 i_2\cdots i_{N-1}i_N\rangle \\ &= A^{(1)}_{i_1}|i_1\rangle A^{(2)}_{i_2}|i_2\rangle A^{(3)}_{i_3}|i_3\rangle\cdots A^{(N-1)}_{i_{N-1}}|i_{N-1}\rangle A^{(N)}_{i_N}|i_N\rangle.\end{aligned} \tag{29}$$

To interpret values for ξ greater than 1, we can look at the *entropy of entanglement* (EOE) of a bipartition of a state.

For the EOE, we construct a density operator

$$\begin{aligned}\rho = A^{(1)}_{i_1\alpha_1}A^{*(1)}_{i'_1\alpha'_1}A^{(2)}_{\alpha_1 i_2\alpha_2}A^{*(2)}_{\alpha'_1 i'_2\alpha'_2}\cdots A^{*(N)}_{\alpha'_{N-1}i'_N}A^{(N)}_{\alpha_{N-1}i_N} \\ |i_1 i_2\cdots i_N\rangle\langle i'_1 i'_2\cdots i'_N| \\ = B^{(1)}_{i_1 i'_1\alpha''_1}B^{(2)}_{i_2 i'_2\alpha''_1\alpha''_2}\cdots B^{(N)}_{i_N i'_N\alpha''_{N-1}}|i_1 i_2\cdots i_N\rangle\langle i'_1 i'_2\cdots i'_N|,\end{aligned} \tag{30}$$

where $B^{(k)} = A^{(k)}A^{*(k)}$ with the indices $\alpha_k\alpha'_k$ grouped into α''_k.

We choose to split the state into two parts at an index $1 \leq k \leq N$, i.e. we define two tensors $\rho_{\mathrm{L}\alpha''_k}$ and $\rho_{\mathrm{R}\alpha''_k}$ as

$$\rho_{\mathrm{L}\alpha''_k} = B^{(1)}_{i_1 i'_1\alpha''_1}\cdots B^{(k-1)}_{i_{k-1}i'_{k-1}\alpha''_{k-1}\alpha''_k}|i_1\cdots i_{k-1}\rangle\langle i'_1\cdots i'_{k-1}| \tag{31}$$

and

$$\rho_{\mathrm{R}\alpha''_k} = B^{(k)}_{i_k i'_k \alpha''_k \alpha''_{k+1}} \cdots B^{(N)}_{i_N i'_N \alpha''_{N-1}} |i_k \cdots i_N\rangle\langle i'_k \cdots i'_N|, \tag{32}$$

and write

$$\rho = \rho_{\mathrm{L}\alpha''_k} \rho_{\mathrm{R}\alpha''_k} \tag{33}$$

where the summation over α''_k goes from 1 to $\xi^{\star 2}_{\alpha_k}$.

To get the reduced density operator of either part, we take the trace over the other. So for ρ_{L}, we get

$$\rho_{\mathrm{L}} = \rho_{\mathrm{L}\alpha''_k} \operatorname{Tr} \rho_{\mathrm{R}\alpha'' k} \tag{34}$$

where due to the trace, only terms with $i_j = i'_j$ in ρ_{R} contribute, i.e. there are only $\xi^{\star 2}_{\alpha_k}$ terms.

Recognizing that each value in $\rho_{\mathrm{L}\alpha''_k}$ with fixed α''_k corresponds to a weighted density operator of a pure state, we see that the rank of ρ_{L} is limited by $\xi^{\star 2}_{\alpha_k}$.

The EOE of ρ_{L} is defined as $Tr \rho_{\mathrm{L}} \log \rho_{\mathrm{L}}$ which is upper bounded by $2 \log \xi^{\star}_{\alpha_k}$.

This is an important insight: The number of nonzero singular values or the *virtual dimension* ξ is connected to the entanglement in a state. Increasing ξ allows us to represent states with higher entanglement but states with low entanglement might be represented well with small ξ leading to a memory cost which is linear in the system size.

3.2.2. *Operators.* Next, we look at unitary operators $\boldsymbol{U}$ on a d-dimensional Hilbert space, with d being a power of 2, which we can represent in terms of $N = \log_2 d$ spin-$\frac{1}{2}$ particles.

Expanding $\boldsymbol{U}$ in the product basis of the subsystems yields

$$\boldsymbol{U} = \underbrace{\langle i_1 i_2 \cdots i_N | \boldsymbol{U} | i'_1 i'_2 \cdots i'_N \rangle}_{U_{i'_1 i'_2 \cdots i'_N i_1 i_2 \cdots i_N}} |i_1 i_2 \cdots i_N\rangle\langle i'_1 i'_2 \cdots i'_N|. \tag{35}$$

Now we can apply the procedure from the section above with a slight modification: Instead of grouping all but one physical index of a tensor we intend to decompose together, we group the two indices associated with a specific spin-$\frac{1}{2}$ Hilbert space together. The rest of the procedure — applying SVDs and repeated grouping and splitting of indices — is the same as before and ultimately yields

$$\boldsymbol{U} = O^{(1)}_{i_1 i'_1 \alpha_1} O^{(2)}_{i_2 i'_2 \alpha_1 \alpha_2} \cdots O^{(N)}_{i_N i'_N \alpha_{N-1}} |i_1 i_2 \cdots i_N\rangle\langle i'_1 i'_2 \cdots i'_N|. \tag{36}$$

Since we group two physical degrees of freedom each, the number of entries for the whole expression is upper bounded by $4N\xi^2$ where ξ is the maximum of all virtual dimensions which is upper bounded by $4^{N/2}$.

The full expansion of $\boldsymbol{U}$ requires $\mathcal{O}(2^{2N})$ memory resources for a general operator $\boldsymbol{U}$.

Applying $\boldsymbol{U}$ to a state $\boldsymbol{\psi}$ works by contracting their representations on each site over a physical degree of freedom:

$$\begin{aligned} \boldsymbol{U}\boldsymbol{\psi} &= O^{(1)}_{i_1 i'_1 \alpha_1} O^{(2)}_{i_2 i'_2 \alpha_1 \alpha_2} \cdots O^{(N)}_{i_N i'_N \alpha_{N-1}} A^{(1)}_{i''_1 \alpha'_1} A^{(2)}_{\alpha'_1 i''_2 \alpha'_2} \cdots A^{(N)}_{\alpha'_{N-1} i''_N} \\ &\qquad |i_1 \cdots i_N\rangle \underbrace{\langle i'_1 \cdots i'_N | i''_1 \cdots i''_N \rangle}_{\delta_{i'_1 i''_1} \cdots \delta_{i'_N i''_N}} \\ &= \left(O^{(1)}_{i_1 i'_1} A^{(1)}_{i'_1} \right)_{\alpha''_1} \left(O^{(2)}_{i_2 i'_2} A^{(2)}_{i'_2} \right)_{\alpha''_1 \alpha''_2} \cdots \left(O^{(N)}_{i_N i'_N} A^{(N)}_{i'_N} \right)_{\alpha''_{N-1}} |i_1 \cdots i_N\rangle, \end{aligned} \tag{37}$$

where $\alpha''_j = (\alpha_j \alpha'_j)$. Thus by applying the operator $\boldsymbol{U}$ we get a new expression where tensors $A^{(j)}$ are replace by a contraction of $O^{(j)} A^{(j)}$ and the virtual indices of the state increase by a factor $\xi_U = \max \xi^{\star}_{\alpha_i}$.

The cost of applying an operator $\boldsymbol{U}$ in the tensor picture is thus $\mathcal{O}(4N\xi_U^2\xi_\psi^2)$ since for N sites we contract two tensors with $4\xi_U^2$ and $2\xi_\psi^2$ entries respectively over common *physical* indices of dimension 2.

Taking $\boldsymbol{\psi}$ to be a product state, the interpretation of the virtual dimension ξ_U of $\boldsymbol{U}$ is straightforward: Applying it entangles the state where the resulting entropy of entanglement in the system is upper bounded by $2\log\xi_U$.

For already entangled states the story is a bit more complicated. First, note that by applying $\boldsymbol{U}$ k times to a product state, the virtual dimension of the state is ξ_U^k which for k large soon means that our representation is even less efficient than writing the tensor of the whole state.

This implies the need for a simplification procedure which we have luckily already encountered: the SVD.

Working from a representation

$$\boldsymbol{U}\boldsymbol{\psi} = A^{(1)}_{i_1\alpha_1}A^{(2)}_{\alpha_1 i_2\alpha_2}\cdots A^{(N)}_{\alpha_{N-1}i_N}|i_1 i_2\cdots i_N\rangle, \tag{38}$$

we can take the tensor $A^{(j)}_{i_j\alpha_{j-1}\alpha_j}$, group indices to get $A^{(j)}_{(i_j\alpha_{j-1})\alpha_j}$ and then apply a SVD such that

$$A^{(j)}_{(i_j\alpha_{j-1})\alpha_j} = A'^{(j)}_{(i_j\alpha_{j-1})\alpha'_j}s_{\alpha'_j\alpha'_j}\tilde{A}^{(j)}_{\alpha'_j\alpha_j} \tag{39}$$

where the summation over α'_j only goes over the non-zero singular values. Contracting $s_{\alpha'_j\alpha'_j}\tilde{A}^{(j)}_{\alpha'_j\alpha_j}$ with $A^{(j+1)}_{\alpha_j i_{j+1}\alpha_{j+1}}$ leads to a new representation where the virtual index α'_j between $A^{(j)}$ and $A^{(j+1)}$ is guaranteed to be limited by the system size by construction. Going back and forth over the system applying this procedure yields a representation that has strict bounds on the virtual dimensions.

Alternatively, we can contract two neighboring tensors $A^{(j)}$, $A^{(j+1)}$ into a tensor $A^{(j,j+1)}_{\alpha_{j-1}i_j i_{j+1}\alpha_{j+1}}$ and use the SVD to split that tensor into two. Again, the number of nonzero singular values will limit the virtual dimension ξ — often to manageable size.

While the procedure just described is an *exact* simplification, we can also *approximate* a state. As we learned in subsubsection 3.1.4, we can use that truncating singular values corresponds to an optimal lower rank approximation.

In other words: The SVD allows us to approximate a state by setting the smallest singular values of the SVD to zero and thus decreasing the virtual dimension. It follows that we can decrease the dimension of the indices that connect the decomposition of a tensor with the SVD and thus the memory requirements of saving the state.

The error introduced due to this truncation is on the order of $\sum_{i>k}\lambda_i$ where k is the cutoff and λ_i are the ordered singular values.

3.2.3. *Measurements.* Measurements can be formulated in terms of an inner product and application of operators, i.e. $p(m) = \langle\boldsymbol{\psi}|\boldsymbol{M}^\dagger\boldsymbol{M}_m\boldsymbol{\psi}\rangle$. In the preceding section we already went over operator application thus the missing element is the inner product.

Let us look at a general inner product of two states $\boldsymbol{\psi}$, $\boldsymbol{\phi}$ in the same Hilbert space where

$$\boldsymbol{\psi} = A^{(1)}_{i_1\alpha_1}A^{(2)}_{\alpha_1 i_2\alpha_2}\cdots A^{(N)}_{\alpha_{N-1}i_N}|i_1 i_2\cdots i_N\rangle \tag{40}$$

and

$$\boldsymbol{\phi} = C^{(1)}_{i'_1\alpha'_1}C^{(2)}_{\alpha'_1 i'_2\alpha'_2}\cdots C^{(N)}_{\alpha'_{N-1}i'_N}|i'_1 i'_2\cdots i'_N\rangle. \tag{41}$$

Then

$$\begin{aligned}(42)\qquad \langle \boldsymbol{\psi}|\boldsymbol{\phi}\rangle &= A^{*(1)}_{i_1\alpha_1} C^{(1)}_{i'_1\alpha'_1} A^{*(2)}_{\alpha_1 i_2\alpha_2} C^{(2)}_{\alpha'_1 i'_2\alpha'_2}\cdots \\ &\qquad A^{*(N)}_{\alpha_{N-1}i_N} C^{(N)}_{\alpha'_{N-1}i'_N} \langle i_1 i_2\cdots i_N|i'_1 i'_2\cdots i'_N\rangle \\ &= A^{*(1)}_{i_1\alpha_1} C^{(1)}_{i_1\alpha'_1} A^{*(2)}_{\alpha_1 i_2\alpha_2} C^{(2)}_{\alpha'_1 i_2\alpha'_2}\cdots A^{*(N)}_{\alpha_{N-1}i_N} C^{(N)}_{\alpha'_{N-1}i_N}.\end{aligned}$$

To get the actual value of the inner product — a scalar — all indices have to be contracted. For a state as above, we can do this starting by contracting the tensors of the left-most site $A^{*(1)}$ and $C^{(1)}$ into a new tensor $T^{(1)}$. The cost of this operation is $\mathcal{O}(2\xi_\psi\xi_\phi)$.

Now the new tensor $T^{(1)}$ can be contracted with $A^{*(2)}$ if ξ_ϕ is larger than ξ_ψ or with $C^{(2)}$ if it is not. The cost of both contractions is $\mathcal{O}(2\xi_\phi\xi_\psi)$.

The resulting Tensor, let us call it $T^{(2)}$, can then be contracted with the remaining tensor on the same site for $\mathcal{O}(2\xi_\phi^2\xi_\psi)$, where it was assumed that $\xi_\phi < \xi_\psi$. This procedure, moving from left to right, can be applied to all sites and at no point is it necessary to use more memory than $\mathcal{O}(2\xi_\psi\xi_\phi)$ and the whole procedure takes $\mathcal{O}(N\xi_\phi^2\xi_\psi)$ operations.

3.2.4. *Composite systems.* As already seen, a tensor product corresponds to a virtual dimension $\xi = 1$, which is a trivial summation. Taking a tensor product thus corresponds to simply writing the coefficients next to each other without any common indices.

The cost of this operation is $\mathcal{O}(1)$ since no actual calculations are required.

3.3. **Tensor Networks.** The example above shows how one can go about decomposing a tensor into many tensors of possibly smaller dimensions. This is the basic principle that is used in tensor networks: We describe a tensor as a contraction of other tensors, needing only to specify those smaller tensors and how to contract them. This is where the *network* in tensor networks comes from. We describe a tensor in terms of a network with edges and vertices. The vertices correspond to the smaller tensors and each leg of a vertex is associated with an index. Edges connecting those legs then correspond to contractions and the *value* of the tensor network, that is, what it represents, is recovered by calculating all those contractions.

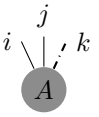
In a way, a tensor network is like a set of ingredients and a recipe of what to do with them to recover a tensor.

For the example of the inner product above, the order of contraction was simple: just contract neighbours with each other in a chain of length N. In general, the structure of a tensor network can be much more complicated which is both a strength and a challenge since many actions on tensor networks correspond to known hard problems of graph-theory requiring intractable algorithms.

3.4. **Penrose Graphical Notation.** One of the challenges we face when working with tensor networks is that it is very easy to lose track of all the indices and what structure they imply. If the tensors in the expression above were not written in order, it would be hard to see that they essentially correspond to a chain of tensors. This is an issue of the algebraic representation: The geometric properties of a decomposition are not obvious.

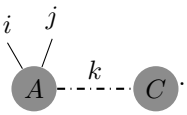
To make the geometry explicit, we introduce a *graphical language* that facilitates reasoning about tensor networks and their structure.

The basic building block is a single tensor, which we write as a circle with legs — one leg for each index. We can label the tensor to make the correspondence between algebraic and graphical representation more explicit, e.g. for a tensor A_{ijk} we write

(43) $A_{ijk}|ij\rangle =$ 

In Equation 43 the legs are not all drawn equal but the leg for the k index stands out. We do this to distinguish between physical and virtual indices. Physical indices are drawn as — and virtual indices are drawn as - - -. The basis elements are *implicit* in the graphical notation; each physical leg stands for the basis of a Hilbert space.

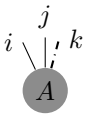
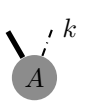
Contractions are written as connections between indices, e.g.

(44) $A_{ijk}C_k =$ 

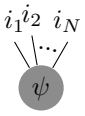
Note that while we explicitly labeled the virtual index k in Equation 44, it does not provide any useful information: k is a bound variable. In the future we will omit labeling contractions unless it seems necessary.

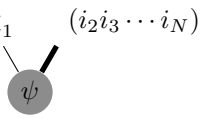
Note also how the structure is apparent on the right side of Equation 44: We have two tensors, one valence-3, the other valence-1, that contract to a valence-2 tensor with two physical indices. Especially for larger algebraic expressions, the graphical notation will help us identifying key features of a state.

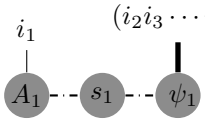
Finally, we can indicate grouping of indices by drawing the leg of a tensor a bit thicker, irrespective of whether the indices are virtual or physical. e.g.

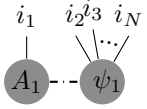
(45) $A_{ijk} =$  $\rightarrow A_{(ij)k} =$ 

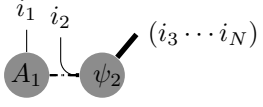
We will now construct the central elements of subsection 3.2 using the graphical language, starting with state vectors

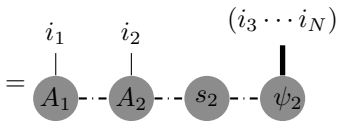
(46) $\boldsymbol{\psi} =$ 

$\rightarrow$  group indices

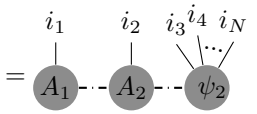
$=$  apply a SVD to the tensor ψ

$\rightarrow$  split indices, contract ψ_1 with s_1

$\rightarrow$  group indices



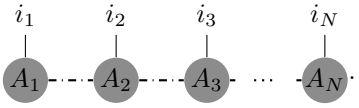
split and apply SVD to ψ_1



contract and split

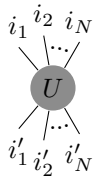
Repeat procedure of grouping, SVD, splitting and contracting for a total of N times to get

(47)

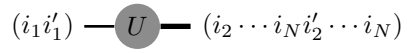


Representing unitaries works similarly by first expanding the operator in terms of local tensors:

(48) $\boldsymbol{U} =$

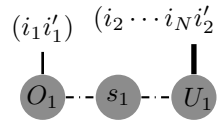


$=$



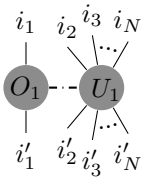
group indices

$=$



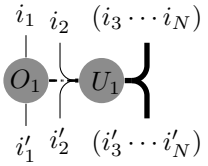
apply a SVD to the tensor U

$=$



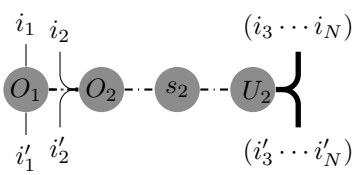
split indices, contract U_1 with s_1

$=$

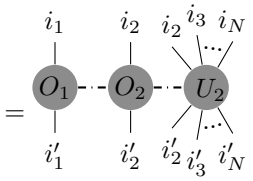


group indices

$=$

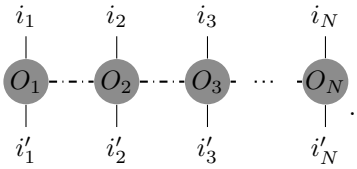


apply SVD to U_1

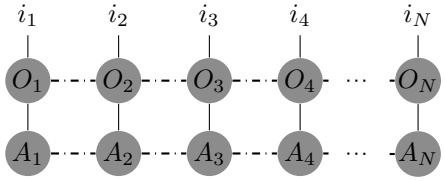


contract and split

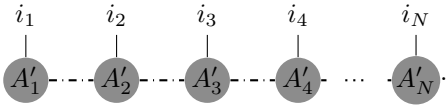
Repeat procedure of grouping, SVD, splitting and contracting

(49) $\boldsymbol{U} =$ 

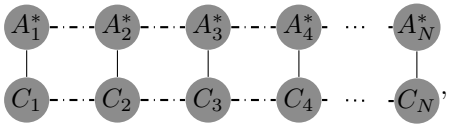
The operator can then be applied straightforward to a state in the same Hilbert space as

(50) $\boldsymbol{U}\boldsymbol{\psi} =$ 

and defining A'_i as the contraction of A_i with O_i

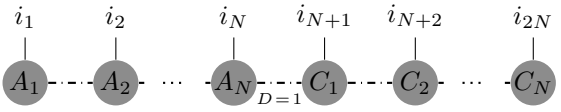
(51) $\boldsymbol{U}\boldsymbol{\psi} =$ 

The inner product can be written as

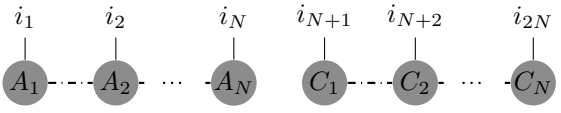
(52) $\langle\boldsymbol{\psi}|\boldsymbol{\phi}\rangle =$ 

and note that if $\boldsymbol{\phi} = \boldsymbol{M}\boldsymbol{\psi}$ this just corresponds to the expectation value if $\boldsymbol{M} = \boldsymbol{M}_m^\dagger \boldsymbol{M}_m$ for some measurement operator $\boldsymbol{M}_m$.

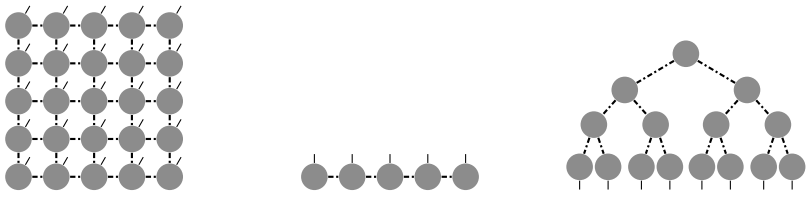
Finally, composite systems are represented either as

(53) $\boldsymbol{\psi} \otimes \boldsymbol{\phi} =$ 

or simply

(54) $=$ 

3.5. **Classes of Tensor Networks.** In subsection 3.2, we outlined how to decompose the tensor describing a quantum state into a chain of tensors with a physical index each. This decomposition is not unique since the actual sequence of physical indices we split off was completely arbitrary. Decompositions that do not lead to such chain-like representations are possible too: There are whole classes of tensor



(A) PEPS (B) MPS (C) TTN

FIGURE 1. Three classes of tensor networks are shown: PEPS, MPS and TTN or *tree tensor networks*. Note that the structure is obvious when looking at the graphical representation while if the tensors were written algebraically, the structure would be hard to make out.

networks with different properties which are due to their *structure*, i.e. the shape of the network.

In Figure 1 three decompositions of states ψ are shown. In this section, we will look at two classes of tensor networks, which are the most useful so far in research: *Matrix Product States* (MPS) and *Projected Entangled Pair States* (PEPS).

3.5.1. *Matrix Product States.* The decomposition of a tensor as outlined in subsection 3.2 naturally leads to an MPS. The name stems from the fact that for any basis element $|i_1 i_2 \cdots i_N\rangle$, the coefficient is given by a product over matrices $A^{(1)}_{i_1} A^{(2)}_{i_2} \cdots A^{(N)}_{i_N}$. A typical MPS is shown in Figure 2.

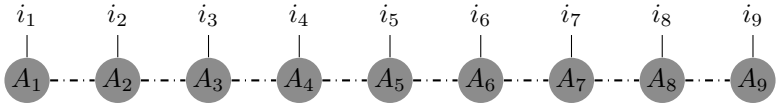


FIGURE 2. An example for an MPS made up of 9 tensors.

The role of the virtual dimension ξ was already outlined in subsection 3.2 where we saw that for any bipartition of a state into two blocks, if we choose ξ such that the EOE is smaller than $2 \log \xi$, we can represent the state exactly as an MPS or to be more concise: A state can be represented well as an MPS if the EOE follows an *area law* of the form

$$S(L) \propto \log \xi, \tag{55}$$

where L is the length of the block considered.

Turning the argument around, we see that an MPS with virtual dimensions ξ can represent any state whose EOE for bipartitions along the MPS is upper bounded by $2 \log \xi$.

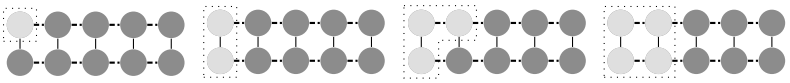
For a random state, ξ generally scales exponentially in the system size. But some physically interesting systems, such as low energy states of gapped Hamiltonians, show exponentially decaying singular values, which in turn implies they are well approximated by MPS with modest ξ [41].

An important part of the efficient decomposition is that the structure of the MPS must *mirror the interactions of the physical system.* That is, if the Hamiltonian is local, neighbours in the physical system should be mapped onto neighbours in the MPS.

The observation of exponentially decaying singular values has been made more precise by the mathematical derivation of *area laws for gapped one-dimensional quantum systems* which implies that MPS can well approximate such states [16].

Some other properties of MPS[27]:

- MPS allow enforcing translational invariance on the tensors and it is possible to take the thermodynamic limit of such tensors.
- MPS are dense, as was shown in subsection 3.2 where increasing ξ increases the subspace of a Hilbert space that can be covered by such an MPS until it covers the whole subspace for exponentially large ξ in the system size.
- MPS with nonexponential virtual dimensions are finely correlated, i.e. the correlation function always decays exponentially with separation distance.
- Expectation values of MPS, i.e. inner products, can be calculated exactly and efficiently if the MPS itself is an efficient representation as was outlined in subsubsection 3.2.3. The standard contraction is indicated in Figure 3.



(A) (B) (C) (D)

FIGURE 3. The inner product of two MPS with virtual dimensions upper bound by ξ can be contracted along the MPSs. For each step, the light nodes that are surrounded by a dotted shape are contracted to a single tensor. Note that at no point the contracted tensor has valence ≥ 3 which implies that in memory it is $\mathcal{O}(2\xi^2)$. Steps (A) to (D) have to repeated $N/2$ times, leading to time requirement of $\mathcal{O}(2N\xi^2)$.

Operators on MPS are called *Matrix Product Operators* or MPOs. We have already met the generic form of an MPO in subsubsection 3.2.2 or in the graphical representation in subsection 3.4.

We have derived MPS by using splitting, grouping and repeated SVDs but we started with a complete description of a state. But usually we use MPS for systems which we could not represent in terms of a single tensor. Instead, we construct an MPS either by just figuring out the tensors of an interesting state, such as a GHZ- or W-state, by starting with a known state and applying MPOs or by starting with a random state and minimizing some quantity that depends on the state — such as the energy — with respect to some Hamiltonian to arrive at a more complex state.

MPS are used in research for e.g. variational renormalization of one-dimensional systems [40], classifying quantum phases [31], to simulate one-dimensional many-body systems [41] or to solve master-equations in one-dimensional systems [23].

3.5.2. *Projected Pair Entanglement States.* PEPS are the prototypical two-dimensional tensor network and an example is shown in Figure 4. As for the MPS, the idea is that the structure of the tensor network *mirrors* the structure of the physical system it represents.

We can look at the EOE of a block in a PEPS. Consider a block of dimension L by L: Changing the state to a density operator, we can trace over one of the systems — we choose the outer — and look at the maximal rank the reduced density operator of the block might have. The reduced density operator of the block is shown in Figure 5.

Counting the virtual indices that are summed over, it is clear that the rank of the density operator is upper bounded by ξ^{4L} and the EOE obeys

$$S(L) \propto L \log \xi, \tag{56}$$

which is the PEPS area law. There is no firm mathematical basis to think that many systems obey this area law as is the case for the one-dimensional case. Yet in

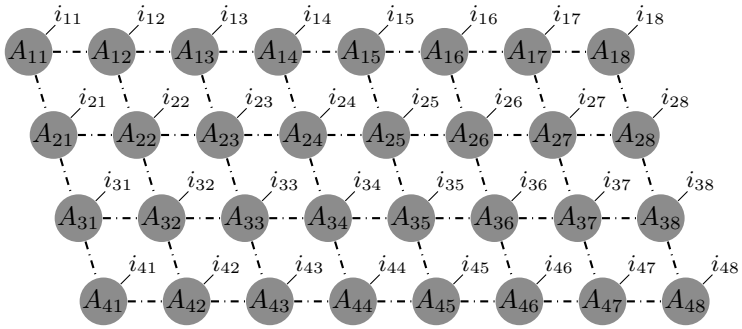


FIGURE 4. Example of a PEPS made up of 32 tensors.

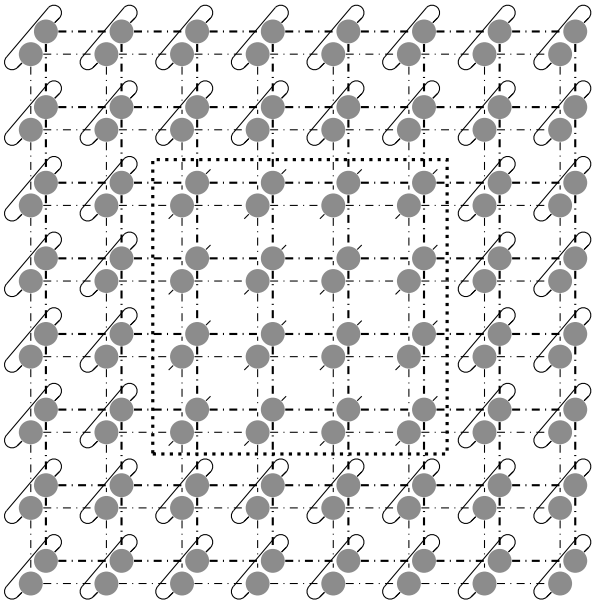


FIGURE 5. The reduced density operator of a block of 4 by 4 tensors, indicated by a dotted rectangle, in a PEPS. The trace is taken over all degrees of freedom outside the block and the maximum rank of the reduced density operator is given by the dimensions of the virtual indices connecting the block with the rest of the PEPS.

practice, many interesting systems seem to obey Equation 56 and can be represented well by a PEPS as e.g. shown in [18].

Some other properties of PEPS[27]:

- PEPS allow enforcing translational invariance on tensors and it is possible to take the thermodynamic limit of such tensors.
- PEPS are dense. Increasing ξ with the system size increases the subspace of a Hilbert space that can be covered by such PEPS where exponential scaling is required to cover the whole space.
- PEPS can handle polynomially-decaying correlations, as opposed to the exponential decay of correlations in MPS. Such correlations can already appear for the smallest nontrivial $\xi = 2$.
- Expectation values of PEPS, i.e. inner products, cannot be calculated both exactly and efficiently. This is because during the contraction of a PEPS,

one necessarily arrives at a point where a tensor with valence $\sqrt{N} = 2^{L/2}$ has to be constructed. See Figure 6 for an illustration.

- Finding the optimal contraction sequence is an intractable problem.

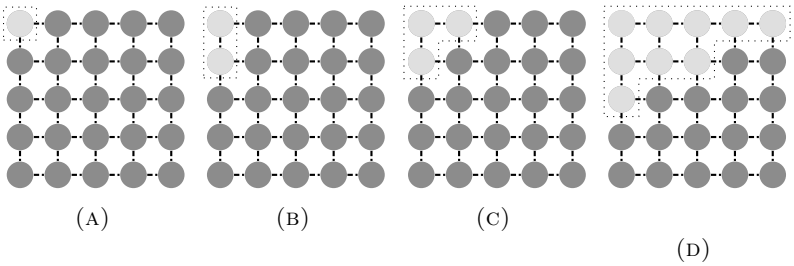


FIGURE 6. (A) to (C) show three consecutive steps in the contraction of the inner product of a PEPS, after the tensors on all sites have already been contracted. (D) shows the inevitable situation where the valence of the contracted tensor is $\mathcal{O}(\sqrt{N})$, i.e. memory resources grow exponentially in the system size.

An operator on a PEPS is called a *Projected Entangled Pair Operator* (PEPO). A PEPO needs to mirror the structure of a PEPS and can be applied by contracting each site that corresponds to the same Hilbert space of the PEPO and the PEPS.

An example of a small PEPO is shown in Figure 7.

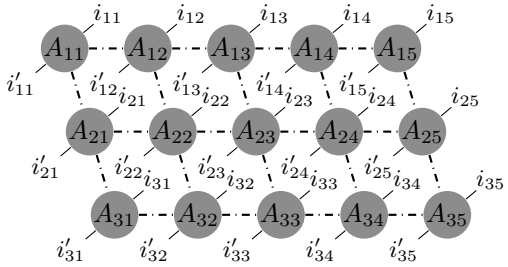


FIGURE 7. Example of a PEPO on 15 sites.

As for the MPS, we often cannot start with a single tensor describing a state and then derive the PEPS but instead we have to construct it by minimization, application of PEPOs or clever choice of the tensors. An explicit construction of a PEPS can be found in subsection 8.3.

PEPS are used e.g. for the simulation of surface codes [3] (as in this thesis), to investigate phases in lattice gauge theories [6, 35], to characterize ground state topology generated by symmetry [30] or to find ground states of strongly correlated quantum lattice models [39].

4. Open Quantum Systems

4.1. **Introduction.** Quantum mechanics as outlined in section 2 deals with *closed* systems which are governed by unitary evolution. Describing a quantum computer as a closed system is a useful simplification, since in order to reason about algorithms and their performance we may ignore all non-essential aspects of the computer.

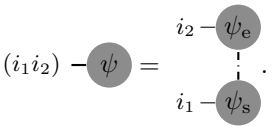
But a quantum computer that is actually a closed system is useless. Both to provide it with an input and read the output, we need to interact with it from

outside. So we need to consider the quantum computer embedded in or linked with another system: the *environment*.

By choosing the environment large enough, presumably the size of the universe should suffice, we can always get to a closed system that follows the nice, familiar rules of unitary evolution of state vectors in a Hilbert space.

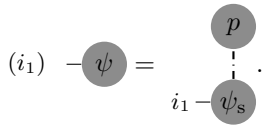
But when describing a quantum computer we do not want and can not take the whole universe into account. Thus we need the theory of *open quantum systems*; a set of tools that allows us to describe an open system embedded in an environment without worrying about the state the environment is in at any time.

4.2. **Quantum Mechanics in Open Systems.** Consider a composite system of two Hilbert spaces: a *system* and an *environment*. The state vector $\boldsymbol{\psi}$ of the composite system can be decomposed into tensors $\boldsymbol{\psi}_\mathrm{e}$ and $\boldsymbol{\psi}_\mathrm{s}$ belonging to the environment and system respectively:

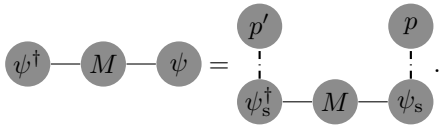
(57) 

If we only want to describe the system, we need to completely erase the environment from our description. The entanglement with the environment allows us to access information about it via measurement of the system. Thus, in order to eliminate the environment from our description, the entanglement has to be erased which introduces *classical* uncertainty about the state of the system.

So say we replace $\boldsymbol{\psi}_\mathrm{e}$ by a tensor representing a probability distribution p, i.e.

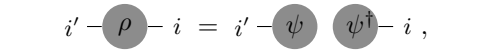
(58) 

Now consider the expectation value of an operator $\boldsymbol{M}$ as

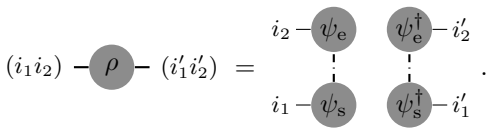
(59) 

But in Equation 59 there are still contributions to the result where the virtual index of p and p' are not equal. In other words: there is *interference* and consequently the claim that we represent *classical* probabilities in Equation 58 is wrong.

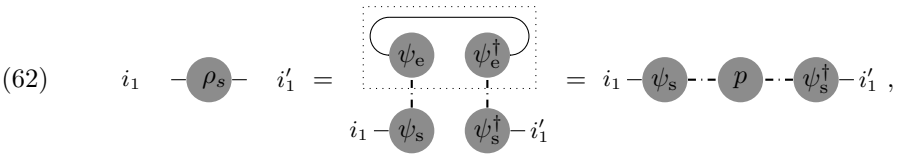
Integrating classical probabilities into our description requires the introduction of the *density operator*. The density operator for a pure state $\boldsymbol{\psi}$ is defined as

(60) 

so our state in Equation 57 in terms of a density operator is written as

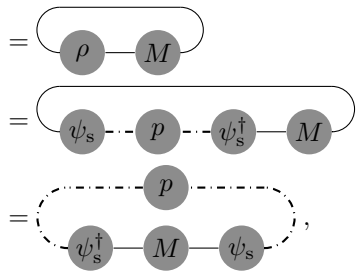
(61) 

We can now eliminate the environment and replace it with a probability distribution p by taking the *partial trace*, i.e. the trace over the environment:

(62) 

where w.l.o.g. p can be considered diagonal. The expectation-value of an operator $\boldsymbol{M}$ given a density operator $\boldsymbol{\rho}$ is then

(63) $$\langle \boldsymbol{M} \rangle = \operatorname{Tr} \boldsymbol{\rho} \boldsymbol{M}$$



where in the last line we just moved the tensors around the contraction and found that the expectation value of $\boldsymbol{M}$ given $\boldsymbol{\rho}$ is simply a sum of expectation values for pure states in $\boldsymbol{\rho}$ weighted by probabilities in p.

Density operators $\boldsymbol{\rho}$ can be defined by three conditions [25]:

(1) $\boldsymbol{\rho}$ is Hermitian
(2) $\operatorname{Tr} \boldsymbol{\rho} = 1$
(3) $\boldsymbol{\rho}$ is a positive operator

Since $\boldsymbol{\rho}$ is Hermitian, it can be diagonalized and has real eigenvalues: $\boldsymbol{\rho} = \sum_i p_i |i\rangle\langle i|$. By the argument above, the p_i can be interpreted as classical probabilities of the states $|i\rangle$. Being probabilities, the p_i have to sum to 1 and since the trace is basis independent, that has to be the case for all representations of $\boldsymbol{\rho}$, thus the second condition. The third condition stems from the requirement that all measurement probabilities are positive. I.e. $\boldsymbol{\rho}$ must not have any negative eigenvalues.

Under unitary evolution, $\boldsymbol{\rho}$ evolves simply by evolution of a decomposition into pure states but to describe non-unitary evolution, a more general formalism has to be used.

4.3. **CPTP maps.** The evolution of an open quantum system is called a *quantum channel* and for fixed time intervals it is described as a *completely positive map* (CP-map).

They need to preserve the properties of a density operator that are needed to fulfil the conditions outlined in subsection 4.2. A map $\mathcal{E}$ is *positive* iff it preserves the positivity of an operators spectrum and *completely positive* iff the composite map $\mathbb{1} \otimes \mathcal{E}$ is positive where $\mathbb{1}$ is the identity channel on a space of density operators with dimensions greater than or equal to the dimension of the space on which $\mathcal{E}$ acts. Note that the spectrum of a density operator is the set of its eigenvalues which correspond to probabilities of pure states. Preserving the positivity of the spectrum thus implies that the spectrum can still be interpreted as probabilities, although possibly not normalized. This is to enable CP-maps to describe measurement.

To guarantee correct normalization, i.e. that the eigenvalues of the density operator sum to one and density operators are mapped to density operators, a further condition is necessary: *Trace preservation.* A CP-map which is trace preserving, i.e. $\operatorname{Tr}[\mathcal{E}(\rho)] = \operatorname{Tr}[\rho]$, is called a *completely positive trace preserving map* (CPTP-map).

Since the trace of a density operator is always one, trace preservation guarantees that quantum channels map valid density operators to valid density operators.

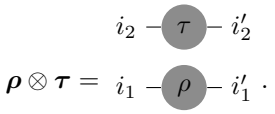
4.4. **Representations.** There are different representations for CPTP-maps but here we will focus on the *system-environment*, the *Kraus* and the *Choi-Matrix* representation.

The system-environment representation is useful if we want to take the physics of the system-environment interaction into account explicitly.

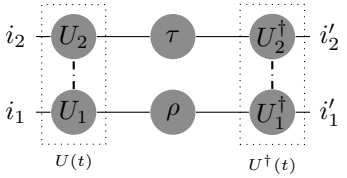
The Kraus representation is useful for a more phenomenological approach, i.e. if we only consider the effects of the environment without their cause and it is mathematically useful because it is easy to check for CPTP in the Kraus representation.

Finally the Choi-Matrix is the form in which CPTP-maps are implemented in subsection 8.3, so we will see how to construct it from the Kraus and System-Environment representations.

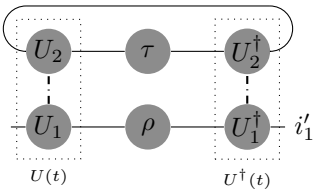
4.4.1. *System-Environment Representation.* Consider a system $\boldsymbol{\rho}$ and an environment $\boldsymbol{\tau}$, that are initially factorized:

(64)


Together they obey unitary evolution

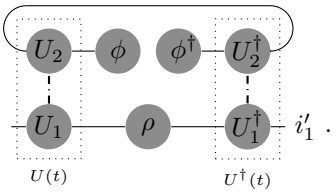
(65)


To get a description of the system state $\boldsymbol{\rho}$ alone, we trace over the environment. This leads to the system-environment representation:

(66)


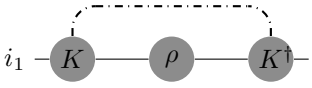
Without loss of generality, we can assume that $\boldsymbol{\tau}$ is a pure state. If it is not, we can always compose it with an additional system and *purify* it.

So

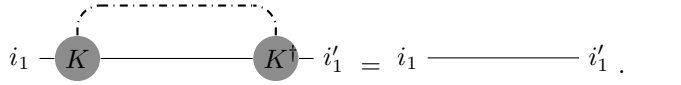
(67)


This is the system-environment representation.

4.4.2. *Kraus Representation.* The Kraus representation ignores the environment itself and only concerns itself with the *effect* of it. We write

(68) $\mathcal{E}(\boldsymbol{\rho}) =$  ,

where each tensor K with fixed virtual index is a *Kraus operator*. In order to be CPTP, the Kraus operators satisfy two criteria. Let $E^{(i)}$ be the Kraus operators for a fixed virtual index i. Then $E^{(i)}$ have to be Hermitian *and* $\sum_i E^{(i)} E^{\dagger(i)} \stackrel{!}{=} \mathbb{1}$ which translates to

(69)  .

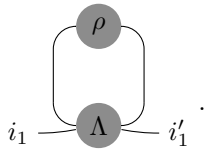
Examples for Kraus operators are given in section 5.

4.4.3. *Choi Matrix-Representation.* Given a channel $\mathcal{E}$, we can construct the *Choi-Tensor* or Choi-Matrix-Representation Λ as

$$\Lambda_{ii'jj'} = |i\rangle\langle i'| \otimes \mathcal{E}(|i\rangle\langle i'|)_{jj'}. \tag{70}$$

Λ is a valence-4 tensor that maps coefficients of states $|i\rangle\langle i'|$ to coefficients of $\mathcal{E}(|i\rangle\langle i'|)$.

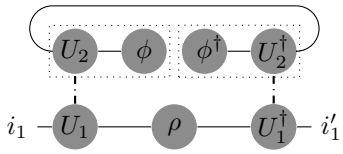
It can be applied as

(71) $\mathcal{E}(\boldsymbol{\rho}) =$  .

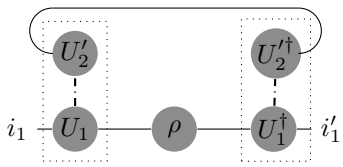
We will show explicitly how to construct CPTP maps in section 5.

4.5. **Conversion to Choi Representation.** Here we quickly sketch how to convert our representations to the Choi representation used in the computational part.

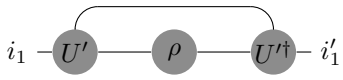
To convert the system-environment to the Kraus representation, we begin with the representation shown above:

(72) $\boldsymbol{\rho}(t) =$ 

contract ϕ, U_2 and $\phi^\dagger$, $U_2^\dagger$ respectively

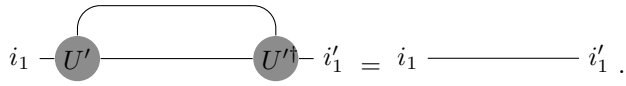
$=$ 

contract U_1, U_2 and $U_1^\dagger$, $U_2^\dagger$ respectively and get

(73) $\boldsymbol{\rho}(t) =$  .

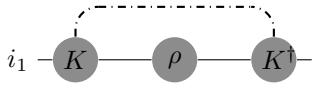
This is the Kraus representation except that the summation is over a physical instead of a virtual degree of freedom. In this case, the distinction is not important

since the indices are bound. The tensors U' and $U'^\dagger$ are valid Kraus tensors as can be seen by noting that

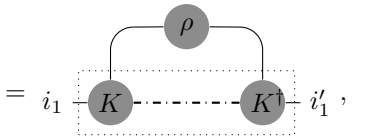
(74) 

due to the structure of U',$U'^\dagger$ in the system-environment representation.

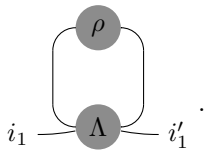
We proceed with the Kraus representation from above:

(75) $\mathcal{E}(\boldsymbol{\rho}) =$ 

Simply changing positions of the tensors leads to



and contracting K, $K^\dagger$ leaves us with

(76) $\mathcal{E}(\boldsymbol{\rho}) =$ 

So we get the Choi representation simply by contracting the Kraus operators over their virtual degree of freedom.

5. Quantum Noise

5.1. **Effects of Noise.** In section 4, we introduced open quantum systems and how to model them in terms of different operator representations. In the context of quantum computation, any effects from the environment outside of our direct control must be interpreted as *noise*; the environment changes our carefully crafted quantum states in unwanted ways.

Here we consider the effects of different kinds of noise on *single qubits*: bit- and phase-flip noise, depolarizing noise, amplitude- and phase damping, and leakage.

We will describe the types of noise in term of their Kraus operators and consider the effect of it on a general one-qubit state.

5.2. One-Qubit Errors.

5.2.1. *Bit- and Phaseflip.* An error that is well known in classical computing is the *bitflip error*, causing a bit to flip its state with a probability p or stay undisturbed with a probability $1-p$. In the quantum version, the bit is flipped in the computational basis.

In terms of Kraus operators, this error is expressed as

$$K_1 = \sqrt{1-p}\,\mathbb{1} \tag{77}$$

$$K_2 = \sqrt{p}\,\sigma_{\mathrm{x}}, \tag{78}$$

defining the channel

$$\mathcal{E}(\rho) = (1-p)\rho + p\sigma_{\mathrm{x}}\rho\sigma_{\mathrm{x}}. \tag{79}$$

It is straightforward to interpret the channel in Equation 79 as a bitflip-channel as described above with e.g. $|0\rangle\langle 0|$ being changed into $|1\rangle\langle 1|$ with probability p or staying the same with probability $1-p$.

An error that does not have a classical counterpart is the *phaseflip error*. Qubits, as opposed to bits, can exist in superpositions and carry information in their relative phase. A phaseflip is then a change of the sign of that relative phase with a probability p or no change with a probability $1-p$.

From the description its clear that the phase flip will change $|+\rangle$ to $|-\rangle$ and vice versa. We note that the phaseflip is simply a bitflip in the σ_{x} basis and thus we find the Kraus operators by a basis change, i.e. conjugation by Hadamards of the operators in Equation 77 and Equation 78:

$$K_1 = \sqrt{1-p}\,\mathbb{1} \tag{80}$$

$$K_2 = \sqrt{p}\,\sigma_{\mathrm{z}} \tag{81}$$

5.2.2. *Depolarization.* Depolarization describes a type of error that completely erases *all* information about a state with a certain probability p, i.e. it takes arbitrary input and with a probability p it returns the completely mixed state $\frac{1}{2}\mathbb{1}$:

$$\mathcal{E}(\rho) = (1-p)\rho + \tfrac{p}{2}\mathbb{1}. \tag{82}$$

Using that for any density operator ρ,

$$\tfrac{1}{2}\left(\rho + \sigma_{\mathrm{x}}\rho\sigma_{\mathrm{x}} + \sigma_{\mathrm{y}}\rho\sigma_{\mathrm{y}} + \sigma_{\mathrm{z}}\rho\sigma_{\mathrm{z}}\right) = \mathbb{1}, \tag{83}$$

we can write Equation 82 in terms of Kraus operators

$$K_1 = \sqrt{1-\tfrac{3}{4}p}\,\mathbb{1} \tag{84}$$

$$K_2 = \sqrt{\tfrac{p}{4}}\,\sigma_{\mathrm{x}} \tag{85}$$

$$K_3 = \sqrt{\tfrac{p}{4}}\,\sigma_{\mathrm{y}} \tag{86}$$

$$K_4 = \sqrt{\tfrac{p}{4}}\,\sigma_{\mathrm{z}} \tag{87}$$

5.2.3. *Amplitude Damping.* Amplitude damping describes an error where a qubit in the computational basis state $|1\rangle$ decays into $|0\rangle$ with a probability γ.

Its Kraus operators are

$$K_1 = \tfrac{1}{2}\left(\left(1+\sqrt{1-\gamma}\right)\mathbb{1} + \left(1-\sqrt{1-\gamma}\right)\sigma_{\mathrm{z}}\right) \tag{88}$$

$$K_2 = \tfrac{1}{2}\sqrt{\gamma}(\sigma_{\mathrm{x}} + \mathrm{i}\sigma_{\mathrm{y}}). \tag{89}$$

The result of repeated application of the channel defined by Equation 84 will inevitably lead to a state $|0\rangle\langle 0|$ because the Kraus operators only describe processes that take a state $|1\rangle$ to a state $|0\rangle$ but not vice versa.

To get a more general case where both transitions are possible, albeit with possibly different probabilities, we consider *generalized amplitude damping* defined by Kraus operators

$$K_1 = \tfrac{1}{2}\sqrt{p}\left(\left(1+\sqrt{1-\gamma}\right)\mathbb{1} + \left(1-\sqrt{1-\gamma}\right)\sigma_{\mathrm{z}}\right) \tag{90}$$

$$K_2 = \tfrac{1}{2}\sqrt{p}\sqrt{\gamma}\left(\sigma_{\mathrm{x}} + \mathrm{i}\sigma_{\mathrm{y}}\right) \tag{91}$$

$$K_3 = \tfrac{1}{2}\sqrt{1-p}\left(\left(1-\sqrt{1+\gamma}\right)\mathbb{1} + \left(1+\sqrt{1-\gamma}\right)\sigma_{\mathrm{z}}\right) \tag{92}$$

$$K_4 = \tfrac{1}{2}\sqrt{1-p}\sqrt{\gamma}\left(\sigma_{\mathrm{x}} - \mathrm{i}\sigma_{\mathrm{y}}\right), \tag{93}$$

where γ is the probability of a transition and p is the probability of that transition being form $|0\rangle$ to $|1\rangle$ and $1-p$ is the probability of the reverse.

5.2.4. *Dephasing.* Dephasing describes an error where *phase information* is lost. We consider it in the computational basis where superpositions of the basis states turn into mixtures of the basis states with a probability α.

Analyzing the error shows that dephasing is actually described by the same Kraus operators as the *phaseflip error*

$$K_1 = \sqrt{1-\alpha}\mathbb{1} \tag{94}$$

$$K_2 = \sqrt{\alpha}\sigma_z. \tag{95}$$

5.2.5. *Leakage.* So far we have assumed that qubits are actual two-level systems. But that need not be the case. We can use the ground and first excited state of a system as the two level system ignoring all higher order states. Depending on the system, this approximates a two-level system very well but there might still be the possibility that during a computation a qubit is excited to a higher-energy state than the first excited state, effectively leaving the computational subspace.

This kind of error is referred to as *leakage* [11].

Leakage can be modelled by considering instead of a qubit a 3-level system whose additional level serves as a leaked state. Leakage can then be understood as generalized amplitude damping with a probability that a computational state is excited into the leakage state on which the quantum operations do not act.

5.3. **Master Equations.** To arrive at a description of the noise from physical considerations as opposed to the purely phenomenological approach or the full system description in subsection 5.2 we can use *quantum master equations.* Master equations model the evolution of a system in terms of (possibly integro-) differential equations. Starting with the Liouville equation that governs both system *and* environment

$$\partial_t \rho = L\rho, \tag{96}$$

with $LA = \mathrm{i}[A, H]$ where H is the Hamiltonian governing the evolution of both system and environment, we wish to trace out the environment to get a channel $\mathcal{E}$ that describes the evolution of the system alone:

$$\rho_s(t) = \mathcal{E}(\rho_s(t_0)). \tag{97}$$

Getting the channel $\mathcal{E}$ usually requires strong assumptions about the nature of the environment and its entanglement with the system. Two of the most common assumptions are the *Born* and the *Markov* assumption [43].

The Born assumption asserts that the environment and the system are only weakly coupled. This is used to argue that the environment is not affected too much by the system and allows us to assume that the joint state of the system and environment is very close to a product state.

The Markov assumption asserts that the environment is much bigger than the system such that the system interacts with a different part of the environment at any time. This is used to argue that the system does not affect itself via the environment or that the environment does not have memory from the point of view of the system.

These two assumptions can be used to derive a *Born-Markov master equation*

$$\partial_t \rho = \mathcal{L}\rho, \tag{98}$$

where $\mathcal{L}\rho$ must be of the form given by the *Lindblad equation.*

5.4. **Lindblad Equation.** The Lindblad Equation is written as:

$$\partial_t \rho = -\mathrm{i}[H, \rho] + \sum_{k=1}^{K} \mathcal{D}(L_k)\rho, \tag{99}$$

where H is the Hamiltonian governing the evolution of the system alone, $\mathcal{D}(A)\rho = 2A\rho A^\dagger - (A^\dagger A\rho + \rho A^\dagger A)$ and the L_k are called *Lindblad operators*. The Lindblad operators need to be such that $\sum_k L_k^\dagger L_k$ is bounded but are arbitrary otherwise.

The form of Equation 99 ensures that $\mathcal{E}(\,)$ is CPTP [43].

5.4.1. *Longitudinal Coupling.* Consider the Lindblad equation with but one Lindblad operator, $L = \sqrt{\phi}\sigma_\mathrm{z}$, undergoing evolution with the Hamiltonian $H = \mathbb{1}$. This Lindblad operator can be derived from a system environment, where the system couples to the environment via σ_z interaction.

The Lindblad equation reads

$$\begin{aligned} \partial_t \rho &= -\mathrm{i}[\mathbb{1}, \rho] + \phi\mathcal{D}(\sigma_\mathrm{z})\rho \\ &= \phi\left(2\sigma_\mathrm{z}\rho\sigma_\mathrm{z}^\dagger - (\sigma_\mathrm{z}^\dagger\sigma_\mathrm{z}\rho + \rho\sigma_\mathrm{z}^\dagger\sigma_\mathrm{z})\right). \end{aligned} \tag{100}$$

Solving this differential equation yields

$$\rho(t) = p_0(t)\rho_0 + p_z(t)\sigma_\mathrm{z}\rho\sigma_\mathrm{z}, \tag{101}$$

with $p_z(t) = \frac{1}{2}(1 - \exp(-\phi t))$ and $p_0(t) = 1 - p_z(t)$.

In Equation 101 it is apparent that the solution can be expressed in terms of Kraus operators $\sqrt{p_0(t)}\mathbb{1}$ and $\sqrt{p_z(t)}\sigma_\mathrm{z}$ which are the operators expressing pure dephasing or phaseflip-noise, as shown in subsection 5.2.

We conclude that the Lindblad operator $\sqrt{\phi}\sigma_\mathrm{z}$ encodes pure dephasing and generalize that to the case of nontrivial Hamiltonians.

5.4.2. *Transversal Coupling.* Next we consider Lindblad operators $L_i = \{\gamma_1\sigma_+, \gamma_2\sigma_-\}$ where $\sigma_\pm = \frac{1}{2}(\sigma_\mathrm{x} \pm \mathrm{i}\sigma_\mathrm{y})$.

The Lindblad equation reads

$$\begin{aligned} \partial_t \rho = -\mathrm{i}[\mathbb{1}, \rho] &+ \gamma_1^2\left(2\sigma_+\rho\sigma_- - \sigma_+\sigma_-\rho - \rho\sigma_+\sigma_-\right) \\ &+ \gamma_2^2\left(2\sigma_-\rho\sigma_+ - \sigma_-\sigma_+\rho - \rho\sigma_-\sigma_+\right). \end{aligned} \tag{102}$$

Solving this equation yields

$$\begin{aligned} \rho(t) = \frac{1}{2}\Bigg(\mathbb{1} &+ \exp\left[-\left(\gamma_1^2 + \gamma_2^2\right)t\right]\left(\mathrm{Tr}\left[\rho\sigma_\mathrm{x}\right]\sigma_\mathrm{x} + \mathrm{Tr}\left[\rho\sigma_\mathrm{y}\right]\sigma_\mathrm{y}\right) \\ &+ \left(\frac{\gamma_1^2 - \gamma_2^2}{\gamma_1^2 + \gamma_2^2} + \exp\left[-2\left(\gamma_1^2 + \gamma_2^2\right)t\right]\left(\mathrm{Tr}\left[\rho\sigma_\mathrm{z}\right] - \frac{\gamma_1^2 - \gamma_2^2}{\gamma_1^2 + \gamma_2^2}\right)\right)\sigma_\mathrm{z}\Bigg). \end{aligned} \tag{103}$$

Comparing the expression with the expressions for noise in subsection 5.2 shows that the action of Lindblad operators $\{\gamma_1\sigma_+, \gamma_2\sigma_-\}$ has the same effect as application of generalized amplitude damping Kraus operators, see subsubsection 5.2.3, with $\gamma = 1 - \exp(2(\gamma_1^2 + \gamma_2^2)t)$ and $p = \frac{1}{2}((\gamma_1^2 - \gamma_2^2)/(\gamma_1^2 + \gamma_2^2) + 1)$.

We thus conclude, as above, that by using these Lindblad operators we can model the effect of general amplitude damping also for systems undergoing nontrivial evolution.

5.4.3. *Longitudinal and Transversal Coupling.* In general, both effects above will come into play and we use the T_1 and T_2 or *relaxation* and *decoherence* time defined as

$$T_1^{-1} = \gamma_1^2 + \gamma_2^2 \tag{104}$$

$$T_2^{-1} = \phi + \tfrac{1}{2}T_1^{-1} \tag{105}$$

to quantify them. T_1 describes the timescale for the decay of $|1\rangle$ to $|0\rangle$ while T_2 describes the timescale for the decay of superpositions $|0\rangle + |1\rangle$ into mixtures $|0\rangle\langle 0| + |1\rangle\langle 1|$ [20].

In terms of these times, $\gamma = 1 - \exp(t/(2T_1))$ for generalized amplitude damping and $\alpha = (1 + \exp(-t/(2T_2))/2$ for dephasing.

The remaining choice we can make is the stationary value of the σ_z coefficient, given by

$$\lim_{t\to\infty} \langle \sigma_z \rangle = \frac{\gamma_1^2 - \gamma_2^2}{\gamma_1^2 + \gamma_2^2}. \tag{106}$$

This way we can model quite general noise on qubits using the Lindblad operators $\{\sqrt{\phi}\sigma_z, \gamma_1\sigma_+, \gamma_2\sigma_-\}$.

For multi-qubit systems, we assume that the Lindblad operators are all single qubit operators and take their tensor products with identities to get multi-qubit operators.

5.5. **Gate Fidelity.** We outlined different errors affecting our state to think about how a state might look like in reality, affected by noise, as compared to the perfect pure states we use to think about algorithms.

We might wonder how far two states, say the pure state we imagine for our quantum computation and the mixed state of the real implementation, are apart.

For this purpose, we define a distance measure for states: The *fidelity*[25]. For two states ρ and σ, the fidelity F is defined as

$$F(\rho, \sigma) = \mathrm{Tr}\left(\sqrt{\sqrt{\rho}\sigma\sqrt{\rho}}\right). \tag{107}$$

While not obvious from Equation 107, the fidelity is symmetric in its inputs.

Additionally, the fidelity is invariant under unitary transformations and it satisfies

$$0 \leq F(\rho, \sigma) \leq 1, \tag{108}$$

where equality with 0 iff ρ and σ have orthogonal support and equality with 1 iff $\rho = \sigma$.

These properties make the fidelity a useful distance measure for *states*.

For gates and channels, we define the *gate* or *channel* fidelity as the minimum state fidelity over all density operators acted upon by the two gates or channels that we wish to compare:

$$F(\mathcal{E}, \mathcal{C}) = \min_{|\psi\rangle} F(\mathcal{E}(|\psi\rangle\langle\psi|), \mathcal{C}(|\psi\rangle\langle\psi|)). \tag{109}$$

That we only need to minimize over all possible pure states can be derived using the property of *strong concavity* of the fidelity.

The gate fidelity tells us how close the output of two channels with identical input is by giving a lower bound on the overlap.

6. Quantum Error Correction

6.1. **Introduction.** Having seen the effects of noise in section 5, we must now think about how to get noise under control in order for quantum computers to work. Without any countermeasures, the state of our computation would irreversibly deteriorate over time and any useful computation would be limited to way below T_1 and T_2 times of a system.

In the following we concentrate on the challenge of just protecting quantum information from environmental effects without considering operations on the state we try to protect, i.e. without actual computation.

To protect information, we have to encode it into bigger systems such that we have certain redundancies. These redundancies allow us to notice when something

goes wrong, i.e. when an error happens on the system, without necessarily affecting the state we wish to protect.

The idea of quantum error correction is thus to encode information in terms of qubits into a larger system of qubits. The encoded qubits carrying the information are called *logical* qubits, while the qubits in which the logical qubits are encoded are the *physical* qubits, alluding to their actually being physical realizations of qubits in quantum computers.

6.2. **Correctable Errors.** Let us assume that k *logical* qubits are encoded in a system of n *physical* qubits. The whole Hilbert space of the system has dimension 2^n while the k logical qubits give rise to a subspace of dimension 2^k. We call the subspace spanned by the logical qubits the *coding space*, i.e. the space of valid codewords.

Consider a general error channel $\mathcal{E}$ specified by Kraus operators E_j and let $|\psi_i\rangle$ be the codewords of our encoding with $i = 1, \ldots, k$.

In order to be able to correct the effect of the error channel $\mathcal{E}$, we need to be able to distinguish codewords it has been applied on, i.e.

$$\begin{aligned} \operatorname{Tr}\left[\mathcal{E}\left(|\psi_i\rangle\langle\psi_i|\right)\mathcal{E}\left(|\psi_j\rangle\langle\psi_j|\right)\right] &= \operatorname{Tr}\left[\sum_{k,l} E_k|\psi_i\rangle\langle\psi_i|E_k^\dagger E_l|\psi_j\rangle\langle\psi_j|E_l^\dagger\right] \\ &= \sum_{k,l}|\langle\psi_i|E_k^\dagger E_l\psi_j\rangle|^2 \\ &\stackrel{!}{=} 0 \end{aligned} \tag{110}$$

for $i \neq j$.

Since all summands are positive, this means that $\langle\psi_i|E_k^\dagger E_l|\psi_j\rangle = 0$ for all $i \neq j$. Additionally, we do not want the measurement of the above overlap to tell us anything about the codeword $|\psi_i\rangle$ but only about the errors $E_k^\dagger E_l$, which means that

$$\sum_{k,l}|\langle\psi_i|E_k^\dagger E_l|\psi_i\rangle|^2 = C_{kl}, \tag{111}$$

where C_{kl} does not depend on i. If the measurement revealed information about the $|\psi_i\rangle$, it would change superpositions of codewords.

Together, the above criteria — correctable errors leave codewords distinguishable and their identification does not reveal information about the codewords — can be summarized as a condition on the Kraus operators of the channels involved:

$$\langle\psi_i|E_k^\dagger E_l|\psi_j\rangle = \delta_{ij}C_{kl}. \tag{112}$$

The equation above is a *necessary condition* for the correctability of quantum errors. To see that it is also *sufficient*, note that C_{kl} is Hermitian and thus diagonalizable. Diagonalizing C_{kl} and rescaling the transformed Kraus operators appropriately, we get

$$\langle\psi_i|E_k'^\dagger E_l'|\psi_j\rangle = \delta_{ij}\delta_{kl} \tag{113}$$

or

$$\langle\psi_i|E_k'^\dagger E_l'|\psi_j\rangle = 0, \tag{114}$$

depending on k.

Equation 113 implies that all errors $E_k'^\dagger E_l'$ are uniquely *identifiable* and correctable except some that obey Equation 114 but whose likelihood is zero.

If there are errors that obey Equation 114, it implies that not all errors are uniquely *identifiable*. We call codes with that property *degenerate*. Codes where all correctable errors can be distinguished are called *nondegenerate*.

So by the argument above a code can identify — and thus correct — all errors due to Kraus operators E_i *iff* it satisfies equation Equation 112[14].

6.3. **Stabilizer Codes.** Equation 112 gives us a general condition on codewords and errors, such that we are guaranteed to be able to correct those errors on the codewords. Stabilizer codes are a framework to construct codewords in a way that allows us to protect our information against specific errors.

From Equation 112 it follows that if we can correct errors due to Kraus operators E_1 and E_2 we can also correct linear superpositions $a_1E_1 + a_2E_2$. Thus, we need only consider a basis for the correctable errors and for that we choose the *Pauli group*.

The Pauli group $\mathcal{G}_n$ is the set of all n-fold tensor products of operators from the set $\{\sigma_\mathrm{x}, \sigma_\mathrm{y}, \sigma_\mathrm{z}, \mathbb{1}\}$ with additional overall factors ± 1 and $\pm \mathrm{i}$ [25]. Note that with these additional factors $\mathcal{G}_n$ forms a *group* under multiplication.

For every $g \in \mathcal{G}_n$ we define the *weight* of g to be the number of sites, i.e. qubits, it acts nontrivially on. Thus e.g. $g = \mathbb{1}^{\otimes 2} \otimes \sigma_\mathrm{x} \otimes \mathbb{1} \otimes \sigma_\mathrm{z} \otimes \mathbb{1}^{\otimes 4}$ is a weight-2 operator in $\mathcal{G}_9$.

From now on we will often drop the n assuming that its value is clear from context or unimportant.

Since $\sigma_i^2 = \mathbb{1}$ for all $i \in \{\mathrm{x}, \mathrm{y}, \mathrm{z}\}$, $g^2 = \pm\mathbb{1}$ for all $g \in \mathcal{G}$ and since σ_x, σ_y and σ_z either commute or anticommute if they act on the same qubit, any two elements in $\mathcal{G}$ either commute or anticommute.

The Pauli operators are all Hermitian but with the additional factors of $\pm\mathrm{i}$ being allowed in $\mathcal{G}$, g is either Hermitian or anti-Hermitian with e.g. $-\mathrm{i}\mathbb{1}$ being evidently anti-Hermitian but it still holds that for all $g \in \mathcal{G} \Rightarrow g^\dagger \in \mathcal{G}$. The operators g also inherit the unitarity of the Paulis and $gg^\dagger = \mathbb{1}$.

With $\mathcal{G}$, we can properly introduce the *stabilizer* $\mathcal{S} \subset \mathcal{G}$. To encode k logical in n physical qubits, we need $\mathcal{S}$ to have 2^{n-k} elements. The code space T can then be defined as

$$T = \{\ |\ \psi\rangle \mid \forall M \in \mathcal{S} \colon M|\psi\rangle = |\psi\rangle\ \}\,. \tag{115}$$

Note that from this definition it follows that $\mathcal{S}$ is Abelian, since only commuting operators can have common eigenvectors, and neither $\mathrm{i}\mathbb{1}$ nor $-\mathbb{1}$ are in $\mathcal{S}$ since they both do not have eigenvectors with eigenvalues $+1$ except the zero-vector.

Having defined the codespace as the mutual $+1$-eigenspace of all stabilizers, we can now go back to the error-correction criteria defined above.

We saw that two elements in $\mathcal{G}$ either commute or anticommute. Consider the case where $E = E_k^\dagger E_l \in \mathcal{G}$ anticommutes with an $M \in \mathcal{S}$ and use

$$\begin{aligned} \langle\psi_i|E|\psi_j\rangle &= \langle\psi_i|EM|\psi_j\rangle \\ &= \langle\psi_i|\{E, M\} - ME|\psi_j\rangle \\ &= -\langle\psi_i|ME|\psi_j\rangle \\ &= -\langle\psi_i|E|\psi_j\rangle \end{aligned} \tag{116}$$

from which it follows that $\langle\psi_i|E|\psi_j\rangle = 0$ which satisfies (112) with $C_{kl} = 0$.

So every set E_i generating $E = E_k^\dagger E_l \in \mathcal{S}$ that anticommutes with an $M \in \mathcal{S}$ is correctable in the stabilizer code.

What about the case where E commutes with all $M \in \mathcal{S}$? Then E must belong the *centralizer* $C(\mathcal{S})$ of $\mathcal{S}$ which is defined as

$$C(\mathcal{S}) = \{\ g \in \mathcal{G} \mid \forall M \in \mathcal{S} \colon [\,M, g\,] = 0\ \}\,. \tag{117}$$

$C(\mathcal{S})$ obviously contains $\mathcal{S}$ but an E that commutes with all of the stabilizer operators might either belong to $C(\mathcal{S})/\mathcal{S}$ or $\mathcal{S}$.

If $E \in \mathcal{S}$, we have

$$\langle\psi_i|E|\psi_j\rangle = \langle\psi_i|\psi_j\rangle = \delta_{ij}. \tag{118}$$

but if $E \in C(\mathcal{S})/\mathcal{S}$, we get

$$ME|\psi\rangle = EM|\psi\rangle = E|\psi\rangle, \tag{119}$$

for some $|\psi\rangle \in T$ which means that $E|\psi\rangle \in T$ is a valid codeword but since $E \notin \mathcal{S}$ there is some $|\psi'\rangle \in T$ that is not fixed by E. Thus E will be an undetectable and uncorrectable error.

The elements in $C(\mathcal{S})/\mathcal{S}$ are operators within the codespace and thus the logical operators on the codewords are contained in it.

In summary, a stabilizer code defined by a stabilizer $\mathcal{S}$ can detect errors E that are either in $\mathcal{S}$ or in $\mathcal{G}/(C(\mathcal{S})/\mathcal{S})$, i.e. belong to the stabilizer or the Pauli group except for the centralizer of $\mathcal{S}$ without $\mathcal{S}$.

If errors are in $C(\mathcal{S})/\mathcal{S}$, they are not detectable and correspond to logical operators that manipulate degrees of freedom in the codespace.

The *distance* of the code is defined in terms of the *weight* of the smallest error $E \in \mathcal{G}$ that is not correctable, i.e. the distance of a code defined by $\mathcal{S}$ is the weight of the operator with the smallest weight in $C(\mathcal{S})/\mathcal{S}$.

6.3.1. *Three-Qubit Repetition Code.* The *repetition code* is an example of a stabilizer code. For the repetition code, we define the stabilizer $\mathcal{S} \subset \mathcal{G}_3$ as

$$\mathcal{S} = \{\sigma_z \otimes \sigma_z \otimes \mathbb{1}, \mathbb{1} \otimes \sigma_z \otimes \sigma_z\}. \tag{120}$$

$\mathcal{S}$ has two elements and acts on three physical qubits, thus stabilizes one logical qubit. The +1-eigenspace of all $M \in \mathcal{S}$ is spanned by the states $|000\rangle$ and $|111\rangle$ which we can use as the logical states $|0\rangle$ and $|1\rangle$.

By inspection we see that the code can correct all errors from the set $E_i = \{\sigma_x^{(1)}, \sigma_x^{(2)}, \sigma_x^{(3)}, \mathbb{1}\}$, where we include $\mathbb{1}$ for *no error*. Note that if we included two-fold tensor products of Paulis, e.g. $\sigma_x^{(1)}\sigma_x^{(2)}$, we get $\sigma_x^{\otimes 3}$ belonging to the set of all possible combinations of E_i but $\sigma_x^{\otimes 3} \in C(\mathcal{S})/\mathcal{S}$, i.e. is an uncorrectable error.

Knowing what errors the three-qubit repetition code can correct, we can look at how the correction is actually achieved. To detect errors, we have to *measure* the operators $M \in \mathcal{S}$, getting as results eigenvalues ± 1. The measurement result for all M is called the *syndrome*. The syndrome is used to locate the errors from the correctable error set in the code.

Every state $E_i|\psi\rangle$, with E_i being a correctable error and $|\psi\rangle$ belonging to the code-space, is associated with one of the 4 possible syndromes (s_1, s_2):

E_i	$\mathbb{1}$	$\sigma_x^{(1)}$	$\sigma_x^{(2)}$	$\sigma_x^{(3)}$
s_1	1	-1	-1	1
s_2	1	1	-1	-1

The sign of s_i is flipped if the nontrivial parts of M_i and E_j overlap since σ_z in M_i and σ_x in E_j anticommute. Since each correctable error has a unique syndrome, the repetition code is *non-degenerate*. As we will see later, this is not a general property of stabilizer codes.

Having identified the error, we can undo it by applying $E_j^\dagger$, using that $E_j \in \mathcal{G}$ and $\forall g \in \mathcal{G} \mid g^\dagger g = \mathbb{1}$. This is a technique that is generally applicable since all elements of $\mathcal{G}$ are invertible by their unitarity.

Generalizing the repetition code to more than three qubits is straightforward, we simply add more elements $\sigma_z^{(i)}\sigma_z^{(i+1)}$ to $\mathcal{S}$ with $i = 1, ..., N-1$ for an N-qubit repetition code. With increasing N the code can correct more simultaneous σ_x- or

bitflip-errors but it cannot detect *phaseflip*-errors at all because σ_z on any qubit commute with all elements of $\mathcal{S}$.

Thus, strictly speaking, the repetition code is a distance-1 code but we can also think of it as a distance-3 code that only corrects bitflip-errors.

An issue we have ignored so far is the requirement to measure the stabilizer which means measuring a two-qubit operator. We just assumed that it is possible to measure two qubits at the same time but currently no implementation for scalable quantum computers considers multi-qubit measurements.

In order to be able to measure two-qubit operators without actually implementing two-qubit measurement in hardware, we need to introduce additional *physical* qubits: Ancillary qubits or *ancillas*. After entangling the ancillas with the data qubits, we can extract information about the data qubits by measuring the ancillas. The circuits for such measurements are shown in Figure 8 and it is easy to show by explicit calculation that they yield the same result as two-qubit measurements.

(A) $Z \otimes Z$ (B) $X \otimes X$

FIGURE 8. Stabilizers are measured indirectly via ancilla measurements after the ancillas are entangled with the data-qubits. In the right circuit the control qubits' bases are changed before and after the CNOTs to get them conditional on σ_x eigenstates.

Adding ancillas — usually one for each stabilizer although we could reuse just one and measure all stabilizers consecutively on it — increases the size of the repetition code from n to $2n-1$. But not only that, it also introduces an additional *source of errors*: Ancilla qubits, being physical qubits, may suffer the same errors as the data qubits but unlike the data qubits they are not embedded in an error correcting code.

Errors on ancillas will change the syndrome and might indicate errors that did not happen — false positives — or indicate errors where none happened — false negatives. Both situations might lead to wrong error correction.

To fight errors on the ancillas, we can *measure repeatedly*. If we initialize the ancilla properly, we can repeat the measurement of the stabilizers multiple times which gives us a higher-dimensional syndrome, where the additional dimension is time. Analyzing this syndrome, and e.g. using a majority vote on the ancillas, enables us to also correct ancilla errors, assuming that we can measure fast enough that additional code qubit errors are not happening during the multiple measurements.

But the code as presented thus far still only corrects bitflip-errors. To correct phaseflip-errors, we can change the basis of our code by conjugating $\mathcal{S}$ with three-fold tensor product Hadamard operators. This transforms $\mathcal{S}$ to

$$\mathcal{S}' = \{\sigma_x \otimes \sigma_x \otimes \mathbb{1}, \mathbb{1} \otimes \sigma_x \otimes \sigma_x\} \tag{121}$$

and the E_i to

$$E_i' = \{\sigma_z^{(1)}, \sigma_z^{(2)}, \sigma_z^{(3)}, \mathbb{1}\}, \tag{122}$$

which can either be seen by inspection as above or noting that a basis change in $\mathcal{S}$ induces the same basis change in E_i.

But this new code cannot correct bitflip errors. We can correct both bitflip and phaseflip errors by *concatenating* codes: A logical qubit is encoded in three logical qubits that are encoded in three physical qubits each for a total of 9 physical qubits.

The first encoding is done using the bitflip repetition code, the second is done using three phaseflip-codes.

The resulting code on 9 physical qubits with two levels of abstraction from physical qubits to three phaseflip-code encoded qubits to one logical bitflip-code encoded qubit is called the *Shor code* [33].

The problem with this approach is that the measurement is highly *nonlocal*. In physical realizations it is preferable to measure only nearest neighbours because long-range entanglement is difficult to keep up during measurement.

But in the Shor code $\mathcal{S}$ contains elements such as $\sigma_z \otimes \mathbb{1}^{\otimes 2} \otimes \sigma_z \otimes \mathbb{1}^{\otimes 2} \otimes \sigma_z \otimes \mathbb{1}^{\otimes 2}$, which require measurement of qubits stretching almost the whole code which would require connections between qubits that are far apart.

This is a problem that renders repetition codes not viable for quantum error correction in scalable systems, especially as the distances grow when using N-qubit repetition codes.

6.3.2. *Surface Codes.* In the preceding section it was shown that to implement repetition codes in physical hardware, one would have to operate on or measure qubits that might be far away from each other. Since in a physical system, due to space constraints, nearest neighbor-interactions are preferred, other codes are considered more useful.

A class of such codes are the *surface codes*. Surface codes evolved from *toric codes* which were presented by Kitaev [19] in the context of simple models for topological systems. Toric codes were then further developed to surface codes in [1] and [4].

Surface codes can be understood as stabilizer codes on a lattice where a qubit sits on each vertex. Consider an m by m lattice with m odd as shown in Figure 9. Note that this is not the standard representation where stabilizers reside on both faces and vertices but a representation as in e.g. [36] instead.

We split the faces of the lattice into two distinct sets, called X-plaquettes and Z-plaquettes with elements X_i and Z_i respectively. In Figure 9, X-plaquettes are colored in a darker shade than Z-plaquettes.

The sets are chosen such that no two elements of the same set overlap at more than one qubit which leads to a chessboard-like grid. Special two-qubit plaquettes are added on the boundaries. Left and right boundaries have two-site X-plaquettes while top and bottom boundaries have two-site Z-plaquettes.

We can now define the stabilizer $\mathcal{S}$ as

$$\mathcal{S} = \prod_{i \in X_i} \sigma_x^{(i)} \cup \prod_{i \in Z_i} \sigma_z^{(i)}. \tag{123}$$

Evidently $\mathcal{S} \subset \mathcal{G}$ and by construction, X_i and Z_j act nontrivially either on distinct or exactly two shared qubits. In the former case, they commute trivially, in the latter case they commute since σ_x and σ_z anticommute and products of an even number of anticommuting operators commute. Also due to Paulis commuting with themselves and the identity, all X_i, Z_j commute among themselves. So we have that all elements of $\mathcal{S}$ commute and thus have common eigenvectors.

The code-space defined by $\mathcal{S}$ is the mutual $+1$-eigenspace of all elements of $\mathcal{S}$. For an m by m lattice, we have m^2 physical qubits. There are $(m-1)^2$ four-qubit plaquettes and $2(m-1)$ two-qubit plaquettes, thus the code has 2 degrees of freedom, corresponding to *one* logical qubit.

The centralizer $C(\mathcal{S})$ of the surface code consists of operators in $\mathcal{G}$ that commute with all elements of $\mathcal{S}$. In our reasoning for why X_i and Z_j commute it was important that they always act nontrivially on either zero or an even number of shared qubits. If we consider errors that consist only of σ_xs or σ_zs, in order to belong to $C(\mathcal{S})/\mathcal{S}$ they have to share an even number of qubits with any plaquette of the other type.

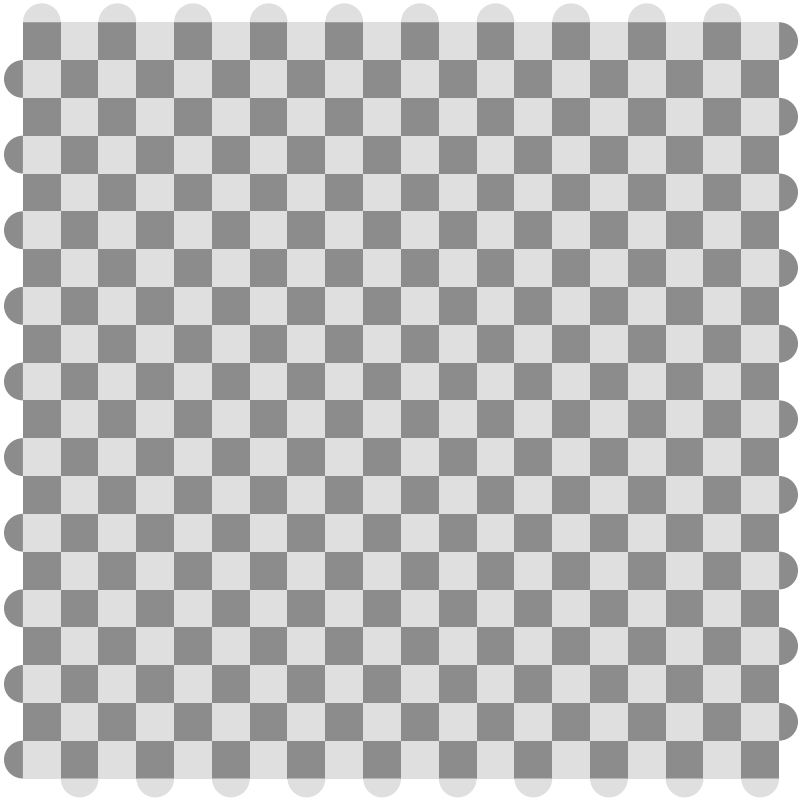


FIGURE 9. A lattice showing the underlying geometry of a surface code. Qubits lie on the vertices of the lattice, dark faces correspond to X-plaquettes, light faces to Z-plaquettes. Notice the boundary plaquettes which border two qubits: The upper and lower, and the left and right boundary have the same type respectively.

This leaves only closed loops of operators or chains of operators of the same kind that end on the borders as possible candidates for different uncorrectable E. While closed loops of Paulis can be constructed as products of stabilizer elements, chains of σ_{x} or σ_{z} that span the whole code are elements of the centralizer but not the stabilizer. Chains of σ_{x} must start and end on the sides while chains of σ_{z} must start and end on top and bottom of the code, since otherwise they do not commute with the two-qubit-boundary plaquettes.

So the only elements $E \in C(\mathcal{S})/\mathcal{S}$ are chains of operators of the same kind that stretch from boundary to boundary.

These are also the logical operators of the coding space T generated by $\mathcal{S}$. We can identify chains of σ_{x} from left to right with the logical σ_{x} and chains of σ_{z} from top to bottom with the logical σ_{z}. Defining the logical $|0\rangle$ as any mutual $+1$ eigenstate of all stabilizer elements, we can construct the logical $|1\rangle$ by applying the logical σ_{x} operator etc.

Logical operators and states are unique up to stabilizer elements. Thus any σ_{x} chain stretching the width of the code is equivalent to a logical σ_{x} operator application.

Since the shortest logical operators or elements in $C(\mathcal{S})/\mathcal{S}$ span the whole width or height of the code, $\mathcal{S}$ on an m by m grid defines a distance-m code.

With surface codes we share the same challenge as with repetition codes, in that the measurements are multi-qubit measurements of either two or four qubits at a time corresponding to the bulk and boundary stabilizer-measurements respectively. Again, this necessitates the introduction of ancillas where each stabilizer might be associated with an ancilla for a total of $m^2 - 1$ ancillas and the code growing to $2m^2 - 1$ physical qubits for a distance m code.

Associating each stabilizer to an ancilla has the advantage that the operations are local: only nearest neighbours have to be entangled before the actual measurement of an ancilla. Compared to the Shor-code from subsubsection 6.3.1 this advantage is achieved at the price of a two-dimensional layout of the qubits.

Correction can again be achieved by measuring all stabilizers to get a syndrome. In the case of surface codes, this syndrome does *not* allow us to uniquely identify the physical error that occurred, yet still allows us to correct its effect if it belongs to the set of correctable errors.

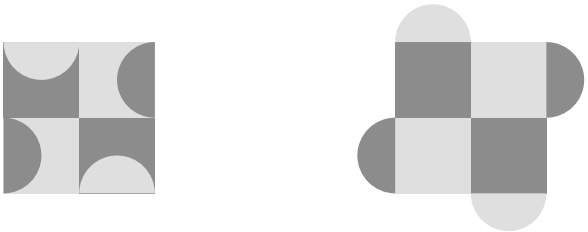
This is shown explicitly in subsection 6.4 but can already be seen by noting that at the corners of Figure 9, one four-qubit stabilizer has to detect two errors itself and since it can only carry one bit of information, this detection cannot allow unique identification. But it is also clear that either of those errors is corrected by correcting the other error because the product of two errors at the edges is a two-qubit stabilizer.

The surface codes are thus *degenerate codes*.

6.4. $\mathcal{S}_{17}$ **Surface Code.** Having defined the general framework for surface codes, we can investigate the *smallest useful* surface code. Small refers to the number of physical qubits and useful implies that the code can both detect *and* correct a nontrivial set of errors. On an m by m grid, these conditions imply that we need a distance-3 code with 9 physical qubits to encode one logical qubit.

But as we have seen above, actual implementation in a quantum computer, due to our limiting ourselves to two-qubit operations and single qubit measurements, necessitates the use of more physical qubits than the nine above: We need *ancillary* qubits for the measurement.

The number of the ancillary qubits is not fixed by the stabilizers of a distance-3 stabilizer code; different configurations are possible.



(A) $\mathcal{S}_{13}$ (B) $\mathcal{S}_{17}$

FIGURE 10. Possible 3 by 3 surface codes: (A) $\mathcal{S}_{13}$: four ancillary qubits are used and each ancilla measures both a four-qubit and a 2-qubit stabilizer one after another. (B) $\mathcal{S}_{17}$: 8 ancillary qubits are used and each ancilla measures exactly one stabilizer.

Reusing ancillary qubits within a syndrome measurement while only allowing local interactions allows a setup as in Figure 10a, where $\mathcal{S}_{13}$ is shown to require 13 qubits.

Using all ancillary qubits once per syndrome measurement, the smallest possible code is $\mathcal{S}_{17}$ shown in Figure 10b, where each ancillary qubit is used for a different stabilizer operator.

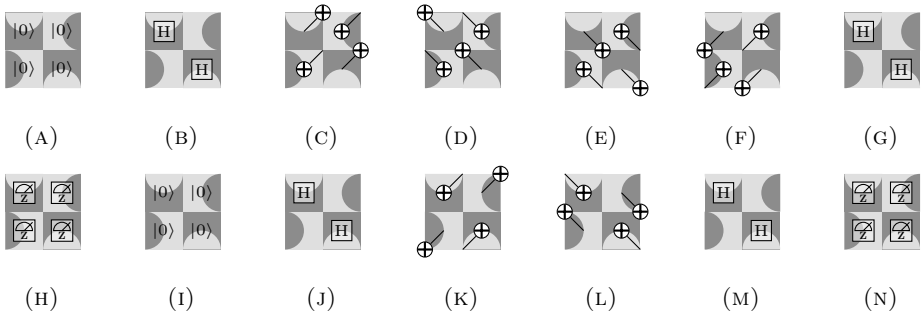


FIGURE 11. The 14 steps for a complete syndrome measurement for $\mathcal{S}_{13}$ are shown in order. From (A) to (H), the four-qubit stabilizers are measured. From (I) to (N), the two-qubit stabilizers are measured. Hadamard operators and measurements in the σ_z eigenbasis are shown as boxes. The CNOT are shown as lines where the endpoint with a cross in a circle signifies the target.

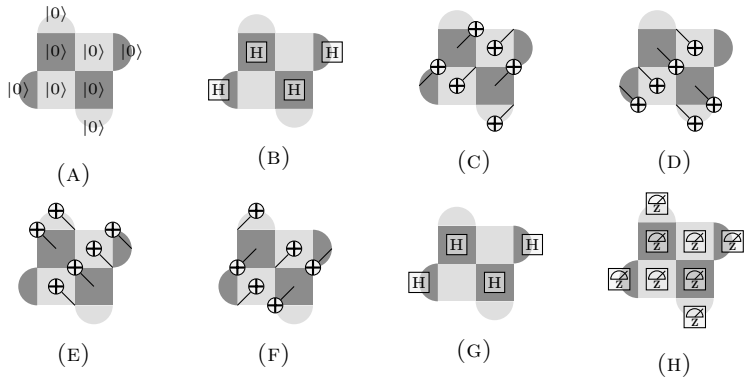


FIGURE 12. The eight steps for a complete syndrome measurement for $\mathcal{S}_{17}$ are shown in sequence. Hadamard operators and measurements in the σ_z eigenbasis are shown.

The algorithms as successions of time steps are shown for both $\mathcal{S}_{13}$ and $\mathcal{S}_{17}$ in Figure 11 and Figure 12 respectively. We see that for one syndrome measurement $\mathcal{S}_{13}$ requires 14 steps due to the reusing of ancillas. $\mathcal{S}_{17}$ only requires 8 steps and thus has a lower *algorithmic depth* than $\mathcal{S}_{13}$.

Coming back to our original idea to use the *smallest useful* surface code, taking algorithmic depth into account, we decide to use $\mathcal{S}_{17}$, preferring the 30% increase in qubits to a 75% increase in algorithmic depth.

Using the labeling in Figure 13, we can explicitly define the stabilizer with elements X_i and Z_i being associated with ancilla i:

$$\begin{aligned}(124)\quad \mathcal{S}_{17} = \{&X_2 = \sigma_x^{(9)}\sigma_x^{(10)}\sigma_x^{(12)}\sigma_x^{(13)}, X_7 = \sigma_x^{(13)}\sigma_x^{(14)}\sigma_x^{(16)}\sigma_x^{(17)},\\ &X_4 = \sigma_x^{(11)}\sigma_x^{(14)}, X_5 = \sigma_x^{(12)}\sigma_x^{(15)}, Z_3 = \sigma_z^{(10)}\sigma_z^{(11)}\sigma_z^{(13)}\sigma_z^{(14)},\end{aligned}$$

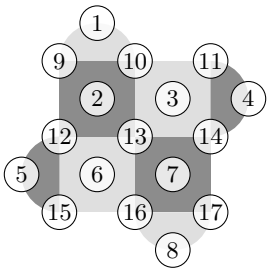


FIGURE 13. All 17 qubits in $\mathcal{S}_{17}$ enumerated.

$$Z_6 = \sigma_{\mathrm{z}}^{(12)}\sigma_{\mathrm{z}}^{(13)}\sigma_{\mathrm{z}}^{(15)}\sigma_{\mathrm{z}}^{(16)}, Z_1 = \sigma_{\mathrm{z}}^{(9)}\sigma_{\mathrm{z}}^{(10)}, Z_8 = \sigma_{\mathrm{z}}^{(16)}\sigma_{\mathrm{z}}^{(17)}\}$$

Here all X_i and Z_i trivially commute among themselves and overlapping elements of different type commute as e.g.

$$\begin{aligned} X_2Z_3 &= \sigma_{\mathrm{x}}^{(9)}\sigma_{\mathrm{x}}^{(10)}\sigma_{\mathrm{x}}^{(12)}\sigma_{\mathrm{x}}^{(13)}\sigma_{\mathrm{z}}^{(10)}\sigma_{\mathrm{z}}^{(11)}\sigma_{\mathrm{z}}^{(13)}\sigma_{\mathrm{z}}^{(14)} \\ &= \sigma_{\mathrm{z}}^{(11)}\sigma_{\mathrm{z}}^{(14)}(\sigma_{\mathrm{x}}^{(10)}\sigma_{\mathrm{z}}^{(10)})(\sigma_{\mathrm{x}}^{(13)}\sigma_{\mathrm{z}}^{(13)})\sigma_{\mathrm{x}}^{(9)}\sigma_{\mathrm{x}}^{(12)} \\ &= \sigma_{\mathrm{z}}^{(11)}\sigma_{\mathrm{z}}^{(14)}(-\sigma_{\mathrm{z}}^{(10)}\sigma_{\mathrm{x}}^{(10)})(-\sigma_{\mathrm{z}}^{(13)}\sigma_{\mathrm{x}}^{(13)})\sigma_{\mathrm{x}}^{(9)}\sigma_{\mathrm{x}}^{(12)} \\ &= (-1)^2\sigma_{\mathrm{z}}^{(10)}\sigma_{\mathrm{z}}^{(11)}\sigma_{\mathrm{z}}^{(13)}\sigma_{\mathrm{z}}^{(14)}\sigma_{\mathrm{x}}^{(9)}\sigma_{\mathrm{x}}^{(10)}\sigma_{\mathrm{x}}^{(12)}\sigma_{\mathrm{x}}^{(13)} \\ &= Z_3X_2. \end{aligned} \tag{125}$$

The set of correctable errors E is

$$E = \{\sigma_{\mathrm{x/z/y}}^{(9)}, \sigma_{\mathrm{x/z/y}}^{(10)}, \sigma_{\mathrm{x/z/y}}^{(11)}, \sigma_{\mathrm{x/z/y}}^{(12)}, \sigma_{\mathrm{x/z/y}}^{(13)}, \sigma_{\mathrm{x/z/y}}^{(14)}, \sigma_{\mathrm{x/z/y}}^{(15)}, \sigma_{\mathrm{x/z/y}}^{(16)}, \sigma_{\mathrm{x/z/y}}^{(17)}\}. \tag{126}$$

Thus $\mathcal{S}_{17}$ can correct the effect of any weight-1 element of the Pauli group.

But if we include a two-qubit error as e.g. $\sigma_{\mathrm{x}}^{(9)}\sigma_{\mathrm{x}}^{(10)}$, we get a chain spanning the whole code $\sigma_{\mathrm{x}}^{(9)}\sigma_{\mathrm{x}}^{(10)}\sigma_{\mathrm{x}}^{(11)}$ in the set of all products of errors which belongs to $C(\mathcal{S}_{17})/\mathcal{S}_{17}$ and is thus an uncorrectable error.

The codespace of $\mathcal{S}_{17}$ is spanned by the states

$$|0\rangle_{\mathrm{L}} = |\psi\rangle \tag{127}$$

$$|1\rangle_{\mathrm{L}} = \sigma_{\mathrm{z}}^{(10)}\sigma_{\mathrm{z}}^{(13)}\sigma_{\mathrm{z}}^{(16)}|\psi\rangle \tag{128}$$

up to stabilizers, where $|\psi\rangle$ is defined as

$$|\psi\rangle = N(\mathbb{1} + X_2)(\mathbb{1} + X_4)(\mathbb{1} + X_5)(\mathbb{1} + X_7)|0\rangle_{\mathrm{data}}^{\otimes 9}, \tag{129}$$

i.e. the projection of the $|0\rangle_{\mathrm{data}}^{\otimes 9}$ state, which belongs to the $+1$ eigenspace of all Z_i, onto the $+1$ eigenspace of all X_j normalized by N.

The states $|0\rangle_{\mathrm{L}}$ and $|1\rangle_{\mathrm{L}}$ are highly entangled states and, as explained in subsubsection 6.3.2, superpositions of all states with either an even or odd numbers of crossings from top to bottom.

For $\mathcal{S}_{17}$ we can explicitly show some of the states spanning the subspace of $|0\rangle_{\mathrm{L}}$ in Figure 14.

Now we can define the logical σ_{x}-operator as

$$\sigma_{\mathrm{xL}} = \sigma_{\mathrm{z}}^{(10)}\sigma_{\mathrm{z}}^{(13)}\sigma_{\mathrm{z}}^{(16)}. \tag{130}$$

or any chain of σ_{z} spanning the code from top to bottom since they are equivalent up to stabilizers, and see that

$$\sigma_{\mathrm{xL}}|0\rangle_{\mathrm{L}} = |1\rangle_{\mathrm{L}} \tag{131}$$

$$\sigma_{\mathrm{xL}}|1\rangle_{\mathrm{L}} = |0\rangle_{\mathrm{L}}. \tag{132}$$

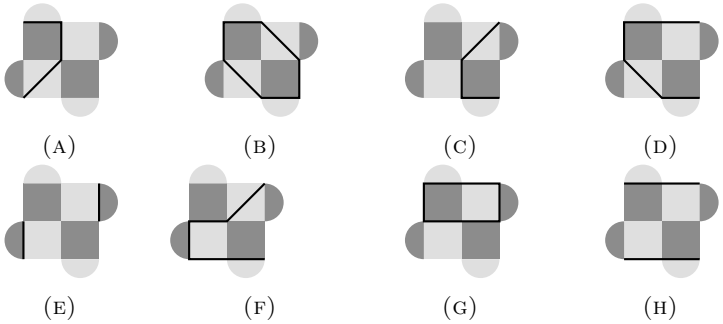


FIGURE 14. Some of the states that are superimposed for the logical 0 state. The black lines imply application of σ_{x} on the qubits they cross.

Defining σ_{zL} as

$$\sigma_{\mathrm{zL}} = \sigma_{\mathrm{x}}^{(12)}\sigma_{\mathrm{x}}^{(13)}\sigma_{\mathrm{x}}^{(14)}, \tag{133}$$

we can confirm that the new logical operators are in fact logical Pauli operators by showing that they obey the same commutation relations as the Pauli operators, e.g. they anticommute as

$$\sigma_{\mathrm{zL}}\sigma_{\mathrm{xL}} = -\sigma_{\mathrm{xL}}\sigma_{\mathrm{zL}}. \tag{134}$$

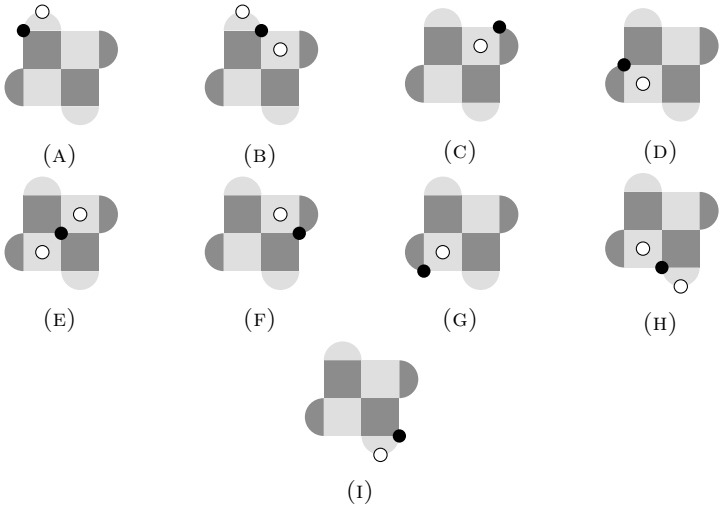


FIGURE 15. The syndrome for all X-errors that $\mathcal{S}_{17}$ can detect by Z-plaquette measurement. Errors are shown as black dots, -1 syndromes as white dots with black boundaries.

As in subsubsection 6.3.1, we detect errors by measuring all stabilizer elements to get a syndrome. The syndrome for all correctable X-errors is shown in Figure 15, where errors are indicated by black dots on data qubits and -1 measurement results by white dots on the ancillas.

The graphic for Z-errors is the same but syndrome and errors are rotated by 90 degrees such that the ancillas are on X-plaquettes instead.

In Figure 15c and Figure 15f we have the same syndrome for different errors as in Figure 15d and Figure 15g, showing explicitly that $\mathcal{S}_{17}$ is a degenerate code.

Errors on ancillas are, as for the repetition code, not correctable if the syndrome is measured just once. Multiple rounds of syndrome measurement are necessary to identify errors on ancillas and correct them without disturbing the code. Actually using the syndrome data then to recover the original computational state requires *decoding*.

6.4.1. *Wen Stabilizers.* For the stabilizer elements introduced so far, we treat errors that manifest as σ_x errors and those that show themselves as σ_z errors separately, each being detected by a different subset of the stabilizer.

In practice, this is a disadvantage if the error probabilities for both bit- and phaseflip errors are not the same: We are limited by the worse of the two. In the limit that one type of error does not occur, half of our syndrome is completely useless.

It thus makes sense to change our stabilizers such that each stabilizer detects both kinds of error. To achieve that, we do the same thing as we have done in subsubsection 6.3.1 to get from a bitflip to a phaseflip code: We change the basis of the stabilizer elements.

Instead of flipping all bases of an $M \in \mathcal{S}$, we flip the basis of all qubits that sit on a even row or column of the grid by conjugating them with Hadamard operators. This way, all our 4 site stabilizers look the same and

$$\begin{aligned}(135)\quad \mathcal{S}_{17} = \{&X_2 = \sigma_x^{(9)}\sigma_z^{(10)}\sigma_z^{(12)}\sigma_x^{(13)}, X_7 = \sigma_x^{(13)}\sigma_z^{(14)}\sigma_z^{(16)}\sigma_x^{(17)},\\ &X_4 = \sigma_x^{(11)}\sigma_z^{(14)}, X_5 = \sigma_z^{(12)}\sigma_x^{(15)}, Z_3 = \sigma_x^{(10)}\sigma_z^{(11)}\sigma_z^{(13)}\sigma_x^{(14)},\\ &Z_6 = \sigma_x^{(12)}\sigma_z^{(13)}\sigma_z^{(15)}\sigma_x^{(16)}, Z_1 = \sigma_z^{(9)}\sigma_x^{(10)}, Z_8 = \sigma_x^{(16)}\sigma_z^{(17)}\}.\end{aligned}$$

Note that we keep the distinction between two kinds of plaquettes since X and Z plaquettes will still detect different errors on different qubits.

These plaquettes were established by Wen [42] and we will use them in $\mathcal{S}_{17}$ for the calculations in subsection 8.2.

The stabilized states transform the same way, as do the logical operators and the set of errors. The set of errors is invariant under that transformation though, which is why we can do this change without drawbacks.

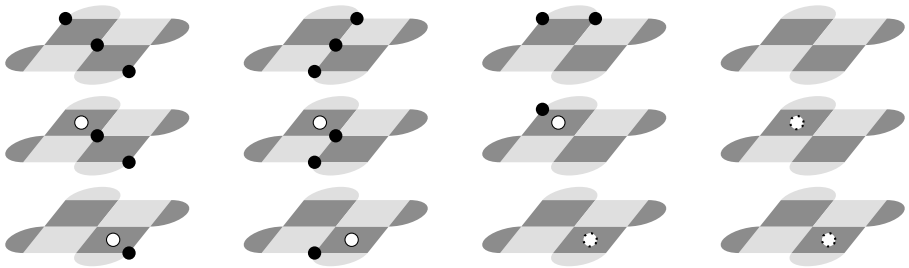
This way, we can treat both types of errors equally, and mitigate any asymmetries of the error probabilities.

6.5. **Decoding.** Decoding is the process of getting from the syndrome to an error correcting procedure. Above we sketched how, given a syndrome, we pick the error correcting procedure by seeing which error corresponds to the syndrome and undoing that error by applying its conjugated operator. This works assuming perfect measurement and no noise on the ancillas.

As mentioned, we can correct ancilla errors by repeating the measurements or comparing the measurements of different error correcting rounds which result in a three dimensional syndrome for $\mathcal{S}_{17}$.

In the three dimensional syndrome it is highly nontrivial to find the set of errors that is likeliest to have caused it or the recovery procedure that is likeliest to succeed. Consider the syndrome in Figure 16: Four different combinations of errors are provided that produce the same syndrome and in the decoding process a decision has to be made as to which set of errors is assumed to have happened.

Optimally, one would calculate the probability for each set of errors that could cause the syndrome and group all the errors that are corrected by the same procedure to finally get the correction procedure that has the highest probability to succeed. For multiple measurement rounds and/or larger surface codes, this quickly becomes computationally infeasible.



(A) (B) (C) (D)

FIGURE 16. A syndrome over three measurement rounds is shown with different errors configurations that could cause it. Black dots correspond to errors, white dots with black bound to -1 stabilizer measurement results and white dots with dotted bound to ancilla errors, i.e. wrong -1 measurement results.

There are multiple algorithms to help us with that, offering approximate solutions with a high probability of success. The most promising is *minimum weight perfect matching* which is based on algorithms which were developed long before the idea of quantum error correction even came to be.

The underlying problem can be formulated as finding a *maximum matching* of a *graph* with *minimal weight*. The first part was tackled by Edmonds in [9] where he outlined the problem and proposed an efficient algorithm to solve it. Given a graph G with vertices v and edges e, a matching is a set of edges such that each vertex is endpoint to just one edge in the matching. A maximum matching is then a matching with the maximum number of edges possible.

Giving each edge a weight, we can ask what the maximum matching is that maximizes the sum of the weights of the edges in the matching. An algorithm to solve this problem was proposed by the same Edmonds in [8].

For decoding in the surface code, we are interested in finding chains of errors whose endpoints are -1 syndrome measurements. The challenge in a three dimensional syndrome is to match up all error-chain ends to deduce the recovery operation necessary.

This problem easily translates into one of finding a matching in a graph where the endpoints of errorchains are the vertices and the edges are all possible connections between the vertices. Weighing an edge by the probability of it occurring, e.g. by calculating the probability of the shortest path causing it, we are interested in finding a matching that maximizes the probability of the edge set having caused those syndrome measurements.

This whole procedure is then called minimum weight perfect matching and it works well as long as the errors are sufficiently *sparse* [12].

For $\mathcal{S}_{17}$ the minimum weight perfect matching algorithm can be translated into a simple lookup table since the number of possible syndromes is limited [36].

If memory is abundant and/or the code is very small, a lookup table that is constructed experimentally or calculated using a theoretical model will be optimal.

Consider the probability of a syndrome $\boldsymbol{s}$ given an initial state $|\phi\rangle$: $p_{|\phi\rangle}(\boldsymbol{s})$, where the syndrome includes the final logical state measurement. If we are interested in a measurement in the computational basis and calculate all $p_{|0\rangle}(\boldsymbol{s})$ and $p_{|1\rangle}(\boldsymbol{s})$, we can correct a state by measuring the syndrome which yields a syndrome $\boldsymbol{s'}$ and then looking at the probabilities $p_{|0\rangle}(\boldsymbol{s'})$ and $p_{|1\rangle}(\boldsymbol{s'})$ and pick whichever conditional probability is greater.

The associated computational basis state of the probability you choose is then the *best possible guess* as to which state the system was in initially, which could be after a computation or a step of a computation. Correction can then be applied to match the measured logical state with the probable logical state [3].

The probability of *failure* is the lesser of the two conditional probabilities, i.e. if one chooses to interpret the state as $|0\rangle$ given a syndrome $\boldsymbol{s'}$, the probability of failure is $p_{|1\rangle}(\boldsymbol{s'})$.

Getting the conditional probabilities can either be achieved by running experiments with known preparations of the initial state or explicit calculations as in subsection 8.2.

7. Quantum Dots as Physical Platform

7.1. **Quantum Dots.** Quantum dots are *artificial atoms* whose properties can be controlled very precisely in experimental setups. At the most basic level, a quantum dot is a structure that *confines* electrons to volumes of space whose lengths are comparable to the wavelengths of the electrons.

For this thesis, we only consider *lateral* quantum dots [21]. To create them, we start with a two-dimensional electron gas (2DEG).

The 2DEG is an electron gas that is strongly confined in one dimension. One possibility for a 2DEG to appear is at the boundary of two semiconductors with different band gaps and suitable doping [10].

Consider for a toy model Figure 17a. We have two semiconductors, gray and white, stacked on top of each other. Due to the higher conduction band of the gray material, it is favorable for electrons to be in the lower semiconductor.

If we introduce a dopant in the upper semiconductor, i.e. an impurity that easily ionizes by giving off electrons, the ions will change the conduction band of the heterostructure, making it more favorable for electrons to be close to the positively charged ions.

If as a consequence the conduction band is distorted as in Figure 17b, we get a small part of the conduction band at the interface of the two materials to dip below the Fermi energy.

This leads to an electron gas at the interface that is strongly confined in one spatial direction: the 2DEG.

Metallic gates can then be placed on the 2DEG as in Figure 17c and by applying a negative potential, we can confine the electrons to specific islands: The quantum dot.

By connecting the quantum dot with electron sources and sinks and adjusting the gate voltages, we can load the quantum dot with single electron precision which allows us to place exactly one electron in the quantum dot.

The electron, as a spin-$\frac{1}{2}$ particle, can be used to encode a qubit in its spin. Using the spin of an electron as opposed to the lowest energy levels of a multi-level system decreases the probability of *leakage*, see subsubsection 5.2.5.

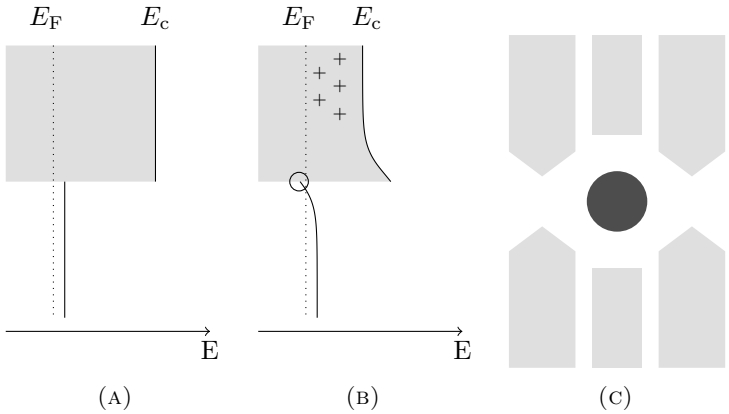
The physics of the electron in a quantum dot are dominated by two effects that are the main sources of noise: *Spin-orbit interaction* (SOI) and *hyperfine interaction* [20].

We first look at SOI. SOI is an effect that can be derived as a first order correction to the naive quantum description of an electron by starting with the *relativistic Dirac equation* in the presence of an electric field $\vec{E}$ [32].

The general SOI term is of the form [44]:

$$H_{\mathrm{SO}} = -\alpha\vec{\sigma}(\vec{E} \wedge \vec{p}), \tag{136}$$

where α is a coupling constant, $\wedge$ is the cross product and $\vec{p}$ is the momentum vector of an electron.



(A) (B) (C)

FIGURE 17. Model for 2DEG. In (A), the different band gaps of the materials are shown with an abrupt change in the conduction band E_c. In (B), the material with the higher conduction and energy was dotted such that positively charged ions remain but the more mobile electrons collect at the interface, distorting the conduction band to just below the Fermi energy E_F (black circle). This is where the 2DEG will form, strongly constrained in the z-direction. (C) Within the 2DEG, a quantum dot (black) can be created by applying negative voltages to metallic gates on the interface (light).

The effect of the electric field $\vec{E}$ can be split into two contributions which we can analyse individually: The *Rashba* term and the *Dresselhaus* term.

The Rashba term is due to the heterostructure-interface, where an electric field confines the electrons into the 2DEG. The magnetic field is thus always in the plane of the 2DEG and leads to an SOI term of the form

$$H_R = \alpha_R(\sigma_x k_y - \sigma_y k_z) \tag{137}$$

with α_R as a coupling constant and $k_{x/y}$ momentum vector components.

The Hamiltonian H_R in Equation 137 causes the spin of an electron to precess around an axis within the 2DEG with the axis depending on the direction of motion of an electron while the frequency is independent of the motion.

The Dresselhaus appears due to a lack of inversion symmetry in the underlying material. The derivation of this term is more involved but it leads to [20]

$$H_D = \alpha_D \left(p_{x'}(p_{y'}^2 - p_{z'}^2)\sigma_{x'} + p_{y'}(p_{z'}^2 - p_{x'}^2)\sigma_{y'} + p_{z'}(p_{x'}^2 - p_{y'}^2)\sigma_{z'} \right), \tag{138}$$

where x′,y′ and z′ are the *crystallographic axes* [100], [010] and [001] respectively.

As a noise source, SOI couples the system to *phonons* in the material which leads to relaxation. We describe that relaxation in terms of the T_1 time for which it can be shown that

$$T_1 \propto \frac{\omega_0^4}{(g\mu_b B)^5}, \tag{139}$$

where on the right side we have the ratio of *orbital level spacing* and the *Zeeman splitting*.

The second effect that dominates noise for the lateral quantum dot is *hyperfine interaction*. The quantum dot itself typically consists of $10^4 - 10^6$ atoms so the electron in the dot might feel the presence of a macroscopic number of nuclear spins at all times [20].

Assuming that a magnetic field B_z is applied perpendicular to the 2DEG, we can describe the interaction with nuclear spins by a Hamiltonian

$$H_{\text{hf}} = \tfrac{1}{2} g\mu_B B_z + \tfrac{1}{2} \sum_k A_k \vec{I}_k \vec{\sigma} \tag{140}$$
$$= \tfrac{1}{2} \left(g\mu_B B_z + \sum_k A_k I_{k,z} \right) + \tfrac{1}{4} \sum_k A_k (I_{k+}\sigma_- + I_{k-}\sigma_+),$$

where $\vec{I}_k$ is the nuclear spin of the k^{th} nucleus with *hyperfine* coupling constant A_k.

From Equation 140 we see that the nuclear spins cause an effective magnetic field, called the *Overhauser field*, acting on the electron.

Unless all nuclear spins are perfectly polarized, there will be fluctuations around the mean polarization for the Overhauser field. This fluctuation, which can be interpreted as an environment that couples via σ_z to the electron-spin, induces dephasing, see subsubsection 5.2.4.

While the interaction of the electron with the nuclei is a complex and non-markovian process, it is well described in the Markov approximation for large magnetic fields B_z where $T_2 \propto B_z^2$.

Qubits in lateral quantum dots can be encoded into the spin state of the single electron occupying the dot as $|0\rangle = |\uparrow\rangle$ and $|1\rangle = |\downarrow\rangle$.

Rotations around the z-axis can be applied by splitting the energy levels of the spin states by applying a magnetic field. The time-evolution will introduce a relative phase between the $|0\rangle$ and $|1\rangle$ state.

Rotations around another axis, e.g. the x-axis, can be executed by *electron spin resonance* [20]: By applying both a constant magnetic field B_z in the z-direction and an oscillating magnetic field B_x in the x-direction, the spin can be rotated from $|0\rangle$ to $|1\rangle$ and vice versa. Thus the whole Hilbert space of the spin can be reached by applying magnetic fields.

Due to the SOI it is even possible to rotate the spin by applying an oscillating electric field and a static magnetic field, using *electric-dipole-induced spin resonance* [13].

For single-qubit gates the gate time is related to the (static) magnetic field since both the Zeeman-splitting and the effect of electron spin resonance are stronger for stronger magnetic fields and can thus be applied in less time.

For initialization, say in a state $|0\rangle^{\otimes N}$, an array of quantum dots can be cooled in a uniform magnetic field in the z-direction. For measurement, the quantum dot can be connected to another quantum dot via a spin-dependent *spin-valve* barrier. Then, by measuring the connected quantum dot, the presence of an electron implies the spin of the kind the spin-valve lets pass [22].

Finally, we need two-qubit gates for universal quantum computing. For two-qubit gates we consider two mechanisms: Overlap of neighboring quantum dots leading to an effective Heisenberg coupling [22] and capacitive coupling of quantum dots via floating gates [38].

7.2. **Overlapping Wavefunctions.** Consider two neighboring quantum dots that are separated by a controllable gate. By lowering the gate potential as shown in Figure 18, the wavefunctions of the two electrons contained in the quantum dots can overlap which leads to a *transient Heisenberg coupling* of the form

$$H_s(t) = J(t) \vec{\sigma}^{(1)} \vec{\sigma}^{(2)}, \tag{141}$$

where $J(t)$ is a time-dependent exchange constant due to the gate potential between the quantum dots.

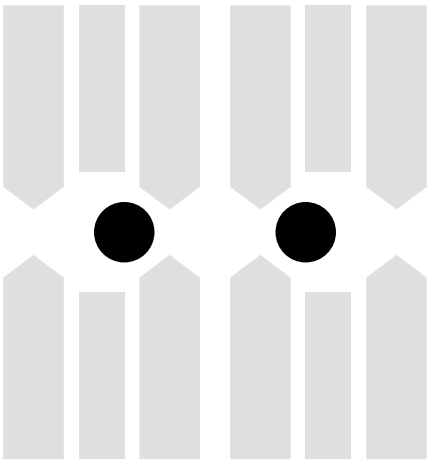


FIGURE 18. Two quantum dots next to each other can interact in a controlled manner by lowering the gate potential between them and effectively moving them closer together.

In [22], the *channel* for an application of the Hamiltonian in Equation 141 including interactions with an environment is derived using both Markov and Born approximation to arrive at

$$\mathcal{V}(t) = \exp[-(t-\tau_{\mathrm{s}})\mathcal{K}_3]\mathcal{U}_{\mathrm{s}}(\tau_{\mathrm{s}})(\mathbb{1}-\mathcal{K}_2), \tag{142}$$

where τ_s is the duration of the spin-spin coupling due to Equation 141 and $\mathcal{U}_{\mathrm{s}}$ is the channel for the Hamiltonian H_{s} acting on the system. The channel

$$\mathcal{K}_2 = \mathcal{U}_{\mathrm{s}}^{\dagger}(\tau_{\mathrm{s}}) \int_0^{\tau_{\mathrm{s}}} d\tau \int_0^{\infty} dt \, \mathrm{Tr}_{\mathrm{E}} \, \mathcal{L}_{\mathrm{int}}\mathcal{U}_{\mathrm{s}}(\tau)\mathcal{U}_{\mathrm{E}}(t)\mathcal{L}_{\mathrm{int}}\rho_{\mathrm{E}}\mathcal{U}_{\mathrm{s}}(\tau_{\mathrm{s}}-\tau) \tag{143}$$

and

$$\mathcal{K}_3 = \int_0^{\infty} dt \, \mathrm{Tr}_{\mathrm{E}} \, \mathcal{L}_{\mathrm{int}}\mathcal{U}_{\mathrm{E}}(t)\mathcal{L}_{\mathrm{int}}\rho_{\mathrm{E}}. \tag{144}$$

The channel $\mathcal{K}_2$ is a (small) correction to the state the system Hamiltonian H_{s} acts on. It takes into account that the system interacts with the environment, initially in state ρ_{E} during the application of $\mathcal{U}_{\mathrm{s}}$, and how the evolution of the environment influences the evolution of the system and vice versa.

The channel $\mathcal{K}_3$ describes the effect of the environment on the system *after* the channel $\mathcal{U}_{\mathrm{s}}$ was applied.

A matrix expression for Equation 142 is derived in [22]. Introducing an inner product $\langle , \rangle$ on operators defined as

$$\langle A, B\rangle = \mathrm{Tr}_{\mathrm{s}}(A^{\dagger}, B), \tag{145}$$

we can write the channel $\mathcal{V}$ as a matrix

$$\begin{aligned} \mathcal{V}_{\alpha\beta|\gamma\delta} &= \langle e_{\alpha\beta}, \mathcal{V}e_{\gamma\delta}\rangle \\ &= \sum_{a,b,\alpha',\beta'} C^{*}_{ab|\alpha\beta}(\exp[-(t-\tau_s)\mathcal{K}_3])_{ab|ab}C_{ab|\alpha'\beta'} \\ &\qquad \exp[-\mathrm{i}\tau_s(E_{\alpha'}-E_{\beta'})]\,(\mathbb{1}-\mathcal{K}_2)_{\alpha'\beta'|\gamma\delta}\,, \end{aligned} \tag{146}$$

where $C_{ab|\alpha\beta} = \langle e^p_{ab}, e^m_{\alpha\beta}\rangle$ is a basis change between the *polarization* and the *multiplet* basis, which are respectively:

$$e_{\alpha\beta} = |\alpha\rangle\langle\beta| \tag{147}$$

$$e_{ab} = e_a^{(1)} e_b^{(2)} \tag{148}$$

with $e^{(i)}_{1,2,3,4} = \frac{1}{\sqrt{2}}(\mathbb{1}, \sigma_{\mathrm{x}}^{(i)}, \sigma_{\mathrm{y}}^{(i)}, \sigma_{\mathrm{z}}^{(i)})$ and $|1\rangle = 2^{-\frac{1}{2}}(|01\rangle - |10\rangle)$, $|2\rangle = 2^{-\frac{1}{2}}(|01\rangle + |10\rangle)$ and $|3\rangle = |00\rangle$, and $|4\rangle = |11\rangle$.

We use *Greek* and *Latin* letters to denote an operator being expanded in the *polarization* and *multiplet* basis respectively.

In Equation 146, channels $\mathcal{K}_2$ and $\mathcal{K}_3$ are

$$\mathcal{K}_{3,ab|cd} = \langle e_{ab}, \Gamma \left(3e_{cd} - \tfrac{1}{2} \sum_i \vec{\sigma} e_{cd} \vec{\sigma} \right) \rangle \tag{149}$$

and

$$\begin{aligned} \mathcal{K}_{2,\alpha\beta|\gamma\delta} = \sum_{i,\alpha'} \big(& \tfrac{1}{4} \delta_{\alpha\gamma} \langle \delta | \vec{\sigma}^{(i)} | \alpha' \rangle \langle \alpha' | \vec{\sigma}^{(i)} | \beta \rangle k^*_{\alpha'\alpha'|\delta\beta} \\ + \quad & \tfrac{1}{4} \delta_{\beta\delta} \langle \alpha | \vec{\sigma}^{(i)} | \alpha' \rangle \langle \alpha' | \vec{\sigma}^{(i)} | \gamma \rangle k^*_{\alpha'\alpha'|\gamma\alpha} \big) \\ & - \sum_i \tfrac{1}{4} \langle \alpha | \vec{\sigma}^{(i)} | \gamma \rangle \langle \delta | \vec{\sigma}^{(i)} | \beta \rangle [k_{\alpha\beta|\gamma\delta} - (k_{\beta\alpha|\delta\gamma})^*], \end{aligned} \tag{150}$$

with

$$k_{\alpha\beta|\gamma\delta} = \tfrac{1}{2\omega_{\alpha\gamma}} \left(\Gamma c_{\delta\beta} - \Delta s_{\delta\beta} + \mathrm{i}(\Gamma s_{\delta\beta} + \Delta c_{\delta\beta}) \right) \left(s_{\alpha\gamma} + \mathrm{i}(1 - c_{\alpha\gamma}) \right), \tag{151}$$

$c_{ij} = \cos(\tau_s \omega_{ij})$, $s_{ij} = \sin(\tau_s \omega_{ij})$ and $\omega_{ij} = E_i - E_j$.

Further $E_1 = -\frac{3}{4} J_0$, $E_{2,3,4} = \frac{1}{4} J_0$, $J_0 = \pi / \tau_s$ and Δ and Γ are parameters expressing the coupling to and the spectral density of the environment.

Choosing τ_{s} such that $\tau_{\mathrm{s}} J_0 = \pi/2$ results in the application of H_{s} being equivalent to a $\sqrt{\mathrm{swap}}$ gate, i.e. halfway a gate that swaps two states.

The $\sqrt{\mathrm{swap}}$ gate can be used to construct a CNOT, using the sequence

$$\mathrm{CNOT}_{\mathrm{loss}} = \sqrt{\sigma_{\mathrm{z}}}^{(1)} \sqrt{-\sigma_{\mathrm{z}}}^{(2)} \sqrt{\mathrm{swap}}\ \sigma_{\mathrm{z}}^{(1)} \sqrt{\mathrm{swap}} \tag{152}$$

and is thus sufficient, together with the single qubit rotations, for universal quantum computation.

7.3. **Floating Gates.** While the suggestion from above needs the quantum dots to be close enough such that we can mediate interactions by gates between them, another suggestion due to [38] allows *long-distance spin-spin coupling*.

Greater distances between quantum dots enable us to cram more control hardware on a quantum chip – an important factor when considering scalability. In the aforementioned paper, *floating gates* are suggested to couple quantum dots *electrostatically* as opposed to via overlap of the wavefunctions, i.e. tunneling, as was presented in [22].

Electron-spins in neighboring quantum dots can couple electrostatically since the electron-spins feel each others charges, i.e. the charge degree of freedom is coupled to the motion, and the motion is coupled to the spin via the SOI.

A detailed analysis of this coupling can be found in [37]. Since the charges are screened by gates and the 2DEG between quantum dots, direct electrostatic coupling is not a useful choice for long-distance coupling.

This is why floating gates were suggested. Floating gates are metallic gates on the 2DEG. Their optimal design, as sketched in Figure 19, consists of two flat metallic disks connected by a thin metallic wire. The two disks are then placed near one quantum dot each and the charge of an electron in the quantum dots induces a mirror charge on the disk above it which in turn charges the far disk with opposite sign.

FIGURE 19. Sketch of a floating gate. The floating gate is suspended above the 2DEG in which two quantum dots (black) are confined by metallic gates (light). The quantum dots can be moved by changing the voltages of the metallic gates and thus change the interaction via the floating gate.

This is the basic mechanism by which the electrostatic force can work long range and as found in [38], the coupling only depends weakly on the wire distance.

A complete analysis of the electrostatic forces yields an effective Hamiltonian of the form

$$H = E_z(\sigma_{\mathrm{z}}^{(1)} + \sigma_{\mathrm{z}}^{(2)}) + J_{12}(\vec{\sigma}^{(1)}\vec{\gamma})(\vec{\sigma}^{(2)}\vec{\gamma}), \tag{153}$$

where J_{12} is a coupling constant that depends on both magnetic field and the details of the metallic gate and

$$\vec{\gamma} = (\alpha_{\mathrm{D}}\cos(2\gamma), -\alpha_{\mathrm{R}} - \alpha_{\mathrm{D}}\sin(2\gamma), 0) \tag{154}$$

with α_{R} and α_{D} being the Rashba and Dresselhaus interaction-strength respectively and γ is the angle between the crystallographic axis along the [100] direction and the axis along the wire.

The Hamiltonian in Equation 153 can be approximated by

$$H' = \tfrac{J_{12}}{2}|\vec{\gamma}_x|^2\left(\sigma_{\mathrm{x}}^{(1)}\sigma_{\mathrm{x}}^{(2)} + \sigma_{\mathrm{y}}^{(1)}\sigma_{\mathrm{y}}^{(2)}\right) + E_z(\sigma_{\mathrm{z}}^{(1)} + \sigma_{\mathrm{z}}^{(2)}) \tag{155}$$

if $E_z \gg J_{12}|\vec{\gamma}_x|^2$ and assuming that the magnetic field is perpendicular to the 2DEG.

We can use H' to implement a $\sqrt{\sigma_{\mathrm{x}}\sigma_{\mathrm{x}}}$ gate in two ways, with two and four applications of H' respectively:

$$\begin{aligned}\sqrt{\sigma_{\mathrm{x}}\sigma_{\mathrm{x}}}_{\mathrm{v1}} = {} & \exp[\mathrm{i}(\sigma_{\mathrm{z}}^{(1)} + \sigma_{\mathrm{z}}^{(2)})E_z t]\exp[-\mathrm{i}H't]\sigma_{\mathrm{x}}^{(1)} \\ & \exp[\mathrm{i}(\sigma_{\mathrm{z}}^{(1)} + \sigma_{\mathrm{z}}^{(2)})E_z t]\exp[-\mathrm{i}H't]\sigma_{\mathrm{x}}^{(1)}\end{aligned} \tag{156}$$

$$\begin{aligned}\sqrt{\sigma_{\mathrm{x}}\sigma_{\mathrm{x}}}_{\mathrm{v2}} = {} & \sigma_{\mathrm{x}}^{(2)}\exp[-\mathrm{i}H'\tfrac{t}{2}]\sigma_{\mathrm{x}}^{(1)}\sigma_{\mathrm{x}}^{(2)}\exp[-\mathrm{i}H'\tfrac{t}{2}] \\ & \sigma_{\mathrm{x}}^{(2)}\exp[-\mathrm{i}H'\tfrac{t}{2}]\sigma_{\mathrm{x}}^{(1)}\sigma_{\mathrm{x}}^{(2)}\exp[-\mathrm{i}H'\tfrac{t}{2}]\end{aligned} \tag{157}$$

where the application time t is given by

$$t = \frac{\pi}{4J_{12}\sqrt{\gamma_x^2 + \gamma_y^2}}. \tag{158}$$

Given an implementation for $\sqrt{\sigma_{\mathrm{x}}\sigma_{\mathrm{x}}}$ it is straightforward to implement a CNOT as

$$\mathrm{CNOT} = \sqrt{\sigma_{\mathrm{z}}}^{(1)}\sqrt{\sigma_{\mathrm{x}}}^{(2)}\mathrm{H}^{(1)}\sqrt{\sigma_{\mathrm{x}}\sigma_{\mathrm{x}}}\,\mathrm{H}^{(1)}. \tag{159}$$

This implementation is only exact for H'. For the real Hamiltonian of Equation 153, the gate, while unitary, will not exactly be a CNOT but instead an approximate channel $\mathrm{CNOT}_{\mathrm{v1}}$ or $\mathrm{CNOT}_{\mathrm{v2}}$ for two and four Hamiltonian applications respectively.

8. Channel Fidelity as Predictor for Code Performance

8.1. **Introduction.** As a synthesis of the topics treated so far, we look at the performance of an $\mathcal{S}_{17}$ stabilizer code for different two-qubit channels. The code performance can be calculated using tensor networks, allowing us to look at the full 17-qubit density operator.

In numerical studies of $\mathcal{S}_{17}$ so far, the two-qubit entangling gates have often been assumed perfect CNOTs [3, 24, 36] or only affected by decoherent noise [26], so our analysis of realistic unitaries with decoherent noise offers novel results.

Since the fidelity is often used to assess the quality of a gate or channel, it is important to know whether it is actually a useful measure in the context of surface codes. In this section we investigate whether the gate fidelity predicts the code performance well or not.

8.2. **Calculation.**

8.2.1. *Channel Fidelity.* The channel fidelity as introduced in subsection 5.5 is a minimization over all states in a Hilbert space. Although we have seen that we only need to take pure states into account, calculating the channel fidelity is still a costly undertaking.

To limit the computational cost of the channel fidelity, we limit ourselves to a minimization over all tensor products of eigenvectors of the Pauli operators σ_x, σ_y and σ_z, among which some result in *maximally entangled* states after application of a CNOT.

The fidelity is then calculated as the minimum overlap of any of those states with the channel to be analysed and a perfect CNOT applied, i.e.

$$\mathcal{F}(\mathrm{CNOT}_\mathrm{test}) = \min_{|\psi\rangle}\{\langle \mathrm{CNOT}\psi | \mathrm{CNOT}_\mathrm{test}\psi\rangle \mid \forall i,j\colon\ |\psi\rangle = \phi_i \otimes \phi_j\}, \tag{160}$$

where ϕ_i are the eigenstates of the Paulis.

As we limit our search space, strictly speaking the channel fidelity we calculate only provides an upper bound on the true channel fidelity but calculations with multiple thousand random pure states suggest that the true fidelity is very well approximated.

8.2.2. *Code Performance.* The code performance of $\mathcal{S}_{17}$ has to be measured by a number that provides *useful information* about the code. We define it as the *probability of a wrong decoding* with an optimal decoder: The lookup table (see subsection 6.5).

For the lookup table we have to calculate probabilities of different syndrome outcomes conditional on the logical state that is encoded before the errors and error correction is applied.

Since for the lookup table decoder, we always choose the logical state whose conditional probability, given the syndrome, is maximal, we simply need to sum all conditional probabilities of the opposite logical state. The result is the probability that after decoding we are in the wrong logical state and thus the decoding failed. So the code performance can be calculated as:

$$p_\mathrm{code} = \sum_{\boldsymbol{s}} \min\{p_{|0\rangle}(\boldsymbol{s}), p_{|1\rangle}(\boldsymbol{s})\} \tag{161}$$

We modify the calculation slightly and only ask what the probability is that a state initialized in the logical $|0\rangle$ is decoded wrong. Then

$$p_\mathrm{code} = \sum_{\boldsymbol{s}} \min\{p_{|0\rangle}(\boldsymbol{s}|0\rangle), p_{|0\rangle}(\boldsymbol{s}|1\rangle)\} \tag{162}$$

where $(\boldsymbol{s}|i\rangle)$ implies that the syndrome includes a logical measurement of $|i\rangle$. We can thus cut the number of calculations in half, getting an upper bound on p_{code} since decoding for initial logical state $|1\rangle$ could be worse but some test calculations suggest that this is not the case.

We consider two scenarios for the code performance:
scenario I. Let $\mathcal{S}_{17}$ be initialized in an eigenstate of the X-plaquettes (Wen style), which we can get by starting in a product state of $|0\rangle$ and applying Hadamards to qubits $10, 12, 14$ and 16 according to the labeling of Figure 13.

Next, we apply preparation noise to *all* qubits in the form of *depolarizing noise* with strength p_{init} where p_{init} is the probability of a Pauli being applied in the Kraus representation of depolarizing noise. One round of error correction is applied where the imperfect CNOTs from section 7 are used as entangling gates. For CNOTs conditional on σ_{x} eigenstates, Hadamards are applied to the control qubits before and after the CNOTs.

The X-plaquettes are measured first indirectly by their ancillas and then directly by measuring each involved data qubit in the basis given by the Pauli that acts on them in the X-plaquettes. The direct measurement corresponds to a *final measurement* in an experiment, since it destroys the entanglement. The measurements yield four results each, which, together with a measurement of the logical Z state, constitute a nine bit syndrome.

From this syndrome we can calculate the conditional probabilities we need to get the code performance for scenario I
scenario II. The second scenario we look at starts with a stabilized state, i.e. a $+1$-eigenstate of all stabilizers of $\mathcal{S}_{17}$. This state is constructed by starting in the same X-plaquette eigenstate as for scenario I and then applying the projectors for the Z-plaquette eigenstates and normalizing the state correctly.

Next, we apply preparation noise on the *data* qubits as depolarizing noise with strength p_{init} and after that apply one round of error correction, again with the quantum dot CNOTs as entangling gates, and measure *all* ancillas to get an eight bit syndrome. Finally we measure the logical operator to get a nine bit syndrome again.

Since we only measure one round of ancillas, we can not correct ancilla errors. This is the reason preparation noise is only applied to the data qubits. All ancilla errors will then be due to the imperfect channels that act as noisy CNOTs.

Additionally all information we get, except the logical measurement, is extracted via the entangling gates.

8.2.3. CNOT *Reference.* As a reference and easily analyzable system, we introduce two simple CNOT gates. Both are based on a perfect CNOT but one has depolarizing noise on both target and control applied beforehand ($\text{CNOT}_{\text{toy1}}$) while the other has depolarizing noise on both qubits applied throughout its application ($\text{CNOT}_{\text{toy2}}$). The latter is calculated by numerically solving the Lindblad equation, Equation 99, with Lindblad operators $\{\sqrt{\gamma/3}\sigma_{\text{x}}, \sqrt{\gamma/3}\sigma_{\text{y}}, \sqrt{\gamma/3}\sigma_{\text{z}}\}$ where $\gamma = -\frac{3}{8}\log[1-\frac{1}{3}p_{\text{CNOT}}]$. Here p_{CNOT} is the probability of the application of a Pauli as in Equation 84, thus $4p_{\text{CNOT}} = p_{\text{depol}}$ where p_{depol} is the probability of depolarization as in Equation 82.

These models also allow for estimates of the results. Consider $\text{CNOT}_{\text{toy1}}$ in scenario I and let the depolarizing noise in the CNOT act as a Pauli with a probability p_{CNOT}. The gate fidelity of $\text{CNOT}_{\text{toy1}}$ is then

$$\mathcal{F}(\text{CNOT}_{\text{toy1}}) \approx 1 - 2p_{\text{CNOT}} \tag{163}$$

while the code performance – remember that $\mathcal{S}_{17}$ can correct one error of each kind – is

$$p_{\text{code}} \approx 36(p_{\text{CNOT}} + p_{\text{init}})^2. \tag{164}$$

So we expect, to lowest order, quadratic scaling of the code performance in the gate *infidelity*, that is, $1 - \mathcal{F}(\text{CNOT}_{\text{toy1}})$, at least for small p_{init}. As the initial noise increases, the linear term $p_{\text{init}} p_{\text{CNOT}}$ is expected to dominate. The same holds for $\text{CNOT}_{\text{toy2}}$.

For scenario II, any error on an ancilla can lead to a failure of the code so p_{code} will have a dominant linear term. Additionally, two errors on the data qubits can lead to an error so superimposed there should be a quadratic dependency on the gate infidelity.

8.2.4. *Parameters.* For the calculations we cast every CNOT from section 7 into a function depending on two parameters each. Additionally, we have the initial noise p_{init} as a parameter for the code performance calculations. The reference CNOTs $\text{CNOT}_{\text{toy1}}$ and $\text{CNOT}_{\text{toy2}}$ are naturally parametrized in terms of the depolarizing noise being applied, thus only depend on one parameter. The depolarizing noise for those CNOT is $p_{\text{CNOT}} \in (0, 0.01)$ where as above we use the probability of the depolarizing noise acting as a Pauli, i.e. the squared amplitude of the Kraus prefactors in Equation 84.

For the floating gate CNOTs, we choose as parameters the ratio between Zeeman energy and coupling constant $R = E_z/J_{12}$ and $\gamma = \gamma_y$, holding $\gamma_x = 1$ fixed. We look at $R \in (30, 35)$ and $\gamma \in (0, 1)$ since for these parameters, the gate fidelity of both CNOT_{v1} and CNOT_{v2} shows interesting dependence on the parameters.

For the overlapping wavefunction CNOT, we use as parameters the time since the beginning of the interaction $t > 1$ in units of the interaction time τ_{s} and the decoherence parameter Γ. We look at $t \in (1, 1.1)$ and $\Gamma \in (0.007, 0.027)$ to stick close to the values used in [22].

An overview of the parameters plus the number of steps n_{steps} in the given range and their size is provided in Table 1. n_{steps} has been chosen such that the interesting structures in the gate fidelity are discernible.

CNOT	parameter	from	to	n_{steps}	Δ_{step}
$\text{CNOT}_{\text{toy1}}$	p_{CNOT}	0	0.01	20	5.26×10^{-4}
	p_{init}	0	0.01	20	5.26×10^{-4}
$\text{CNOT}_{\text{toy1}}$	p_{CNOT}	0	0.01	20	5.26×10^{-4}
	p_{init}	0	0.01	20	5.26×10^{-4}
CNOT_{v1}	R	30	35	21	0.25
	γ	0	1	6	0.2
	p_{init}	0	0.1	51	2×10^{-3}
CNOT_{v2}	R	30	35	10	0.556
	γ	0	1	40	0.0256
	p_{init}	0	0.1	15	7.1×10^{-3}
$\text{CNOT}_{\text{loss}}$	t	1	1.1	5	0.025
	Γ	0.007	0.027	10	0.0022
	p_{init}	0	0.1	30	0.00344

TABLE 1. Parameters used in the calculations for the different CNOTs.

8.3. **Computational Model for $\mathcal{S}_{17}$.**

8.3.1. *Representing $\mathcal{S}_{17}$ as a Tensor Network.* Tensor networks can be represented on a computer as a set of vertices corresponding to the qubits and a set of edges, corresponding to the contractions between the tensors as in the graphical language of subsection 3.4.

The tensor network for one expectation value for the two scenarios presented in subsubsection 8.2.2 is then built by separating the description of the calculations in distinct layers for

(1) Initial state preparation (one to four layers)
(2) Preparation noise
(3) 1st entangling round
(4) 2nd entangling round
(5) 3rd entangling round
(6) 4th entangling round
(7) Projectors to measurement results (two to five layers)

In each layer tensors are specified on all sites although some will just be identities. The layers can then be connected by adding edges between the same sites of consecutive layers, taking care of connecting the correct indices to build a tensor network that, upon contraction, yields the probability of the measurement results corresponding to the projectors.

The algorithms for the two scenarios we consider are shown in Figure 20a and Figure 20b where additional projectors are used in Figure 20b to construct an initial state stabilized by all stabilizer elements while in Figure 20a the stabilizer measurement requires additional layers.

Since preparation noise and the entangling rounds are channels, the initial state is entered as a density matrix.

8.3.2. *Building Tensors.* The tensors for the qubits – either density matrices, channels or projectors – are constructed in different ways.

For the initial product state, we can simply transform state vectors to density matrices. The preparation noise tensors $\mathcal{E}$ can be constructed by considering the Kraus operators of the depolarizing noise from Equation 84 and building a Choi representation with them as shown in subsection 4.5.

Since the measurements we consider are measurements in the Pauli basis and the stabilizers only consist of Paulis, we need to build tensors for projections onto Pauli eigenstates.

Noting that all $M \in \mathcal{G}$ square to the identity up to global phases, their projectors into the $\pm$-eigenspace P_M can be expressed as

$$P_M \propto (\mathbb{1} \pm M)\,. \tag{165}$$

We can implement this by defining tensors $\hat{M}^{(j)}_{ii'\boldsymbol{\alpha}}$ where $\boldsymbol{\alpha}$ is a vector of virtual indices [3]. Then we define the only non-zero components of $\hat{M}^{(j)}$ to be:

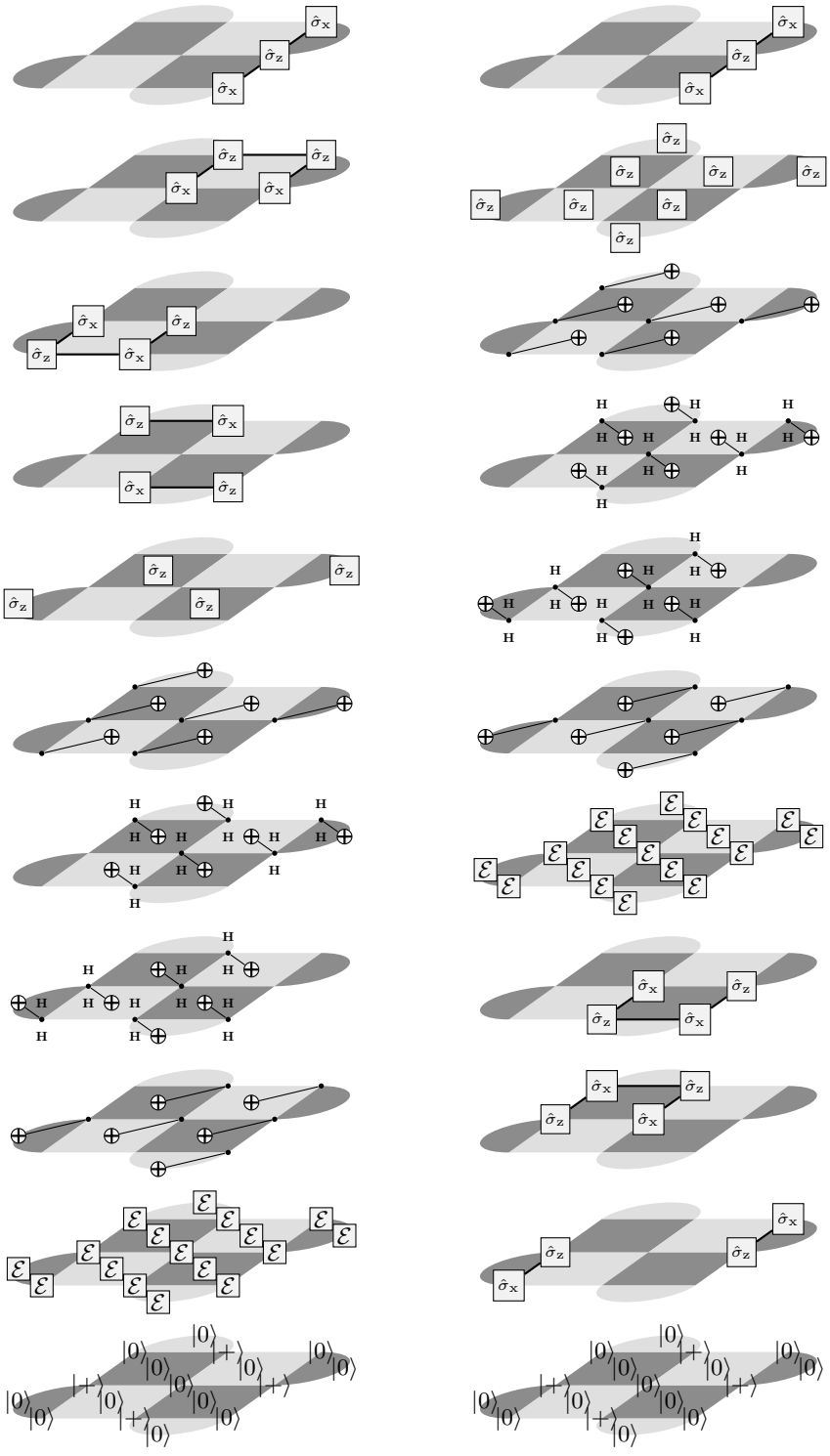
$$\hat{M}^{(j)}_{ii'\boldsymbol{\alpha}=\mathbf{0}} = \mathbb{1}_{ii'} \tag{166}$$

$$\hat{M}^{(j)}_{ii'\boldsymbol{\alpha}=\mathbf{1}} = M^{(j)}_{ii'} \tag{167}$$

where $M^{(j)}$ is is the stabilizer operator, i.e. a Pauli acting on site j. $\boldsymbol{\alpha}$ has either one or two components, corresponding to one or two virtual indices that are needed to contract the $\hat{M}^{(j)}$ representing a P_M.

It is easy to see that either all virtual indices are 1 or they are all 0 for the projector to be nonzero. In the first case, M is applied to all qubits, in the second, $\mathbb{1}$ is applied. Thus we get a projector as in Equation 165 if we take care of correct normalization and the overall sign of the components with virtual indices 1 for either the $+1$ or -1 eigenspace projector.

For ancilla measurements, the projectors are simple single-qubit tensors without virtual degrees of freedom of the form in Equation 165 where M is a σ_z matrix and the proportionality constant is $1/2$.



(A) scenario I (B) scenario II

FIGURE 20. Algorithms for the two scenarios considered. Each layer corresponds to one time step and the graphics are to be read from bottom to top. The dots with little H below and above denote that the CNOT is conditional on the $\pm$ states instead of the 0/1 states.

The two-qubit operators are transformed to Choi representations as in section 4. For the floating gate CNOTs — CNOT_{v1} and CNOT_{v2} — the unitary gates can be used as Kraus operators to get the Kraus representation from which it is easy to get the Choi representation. The channel $\text{CNOT}_{\text{loss}}$ can be directly converted to a Choi representation. The resulting two-qubit tensors are split into two local tensors, using the SVD in the same way as in subsubsection 3.2.2.

Thus we can construct all tensors shown in Figure 20.

8.3.3. *Contracting the Tensor Network.* The whole tensor network for a probability is contracted by successive *pairwise contraction*. In theory, we could attempt to find the optimal contraction sequence for the 101 and 105 nontrivial tensors in the two scenarios to contract them with the minimal computational resources.

This would mean finding the optimal sequence among the $> 101!$ possible contractions, which in practice is impossible to do exactly. We thus just contract the qubits of the same sites to reduce our layers to but one layer.

This final layer consists of 17 tensors, which makes it possible to find the optimal contraction order with reasonable resources.

For the implementation in this thesis, we used an algorithm presented in [29] which we sketch here.

The algorithm is based on a *breadth-first constructive approach* which works as follows for a tensor network with n tensors: We set up n sets S_i with $i \in 1, \ldots, n$. We fill S_1 with the tensors of the tensor network.

In each set $S_{i>1}$ we place all subsets of length i of tensors in S_1. We consider those elements in S_i to be the tensors resulting from the contraction of the elements of the sublists.

Then iteratively, starting with S_2, we ask for each element M in the set S_i: What is the cheapest contraction of two elements of sets $S_{j<i}$ such that the resulting tensor is M?

The cost is then calculated for each pair of tensors A, B that contract to M by calculating the cost of contracting A and B together and adding the cost of the creation of A and B themselves. Since if we deal with $M \in S_i$, we already found the lowest cost of all elements $A, B \in S_{j<i}$, we find at each step of the iteration the cheapest cost of constructing an element M by pairwise contraction of elements in $S_{j<i}$ and when we arrive at S_n, we only have one subset of S_1, the contraction of all tensors, and its associated lowest cost.

We can track the contractions leading to the lowest costs at each step and arrive at a — possibly not unique — sequence of optimal pairwise contractions at the end.

This algorithm can be improved by introducing a limit on the cost we consider and abort all contractions that cost more than that. Increasing the limit each time the algorithm finishes without constructing the contraction sequence for S_n, we can iteratively get the contraction sequence without considering all possible sequences.

With this optimal contraction sequence we can contract the tensor network to get its value, i.e. the probability of the measurement results enforced by the projectors.

8.3.4. *Simplifying the Tensor Network.* While we can apply arbitrary many single-qubit operators without increasing the size of our tensors, two-qubit operators increase the virtual dimension between the tensors they operate on.

To counteract this growing of the virtual dimension, we apply a simplification procedure before the final layer is contracted. The procedure uses the SVD to truncate virtual dimensions as shown in subsubsection 3.1.4: We contract tensors that share an edge, i.e. a contraction, and then split them again using a SVD and discard all vanishing singular values.

This allows us to decrease the computational cost of the final contraction by doing more work upfront in the simplification. But while the simplification costs mostly time, the final contraction can have great demands on memory. In the trade-off whether we invest more time or memory, we choose to invest more time since memory is limited on the hardware available to us.

8.3.5. *Limitations.* Tensor network methods are especially suited for low-entanglement systems. Quantum algorithms and quantum error correction, using entanglement as a resource, in contrast produce *highly entangled* states.

The success of our analysis is not due to low entanglement of the involved operations. Quite the opposite: The CNOTs used are strongly entangling with a maximal virtual dimension between control and target qubit of 16. The reason a tensor network analysis of the algorithm is possible where a full density operator expansion would fail is the limited *depth* of the algorithm, i.e. that we only apply one round of error correction.

With more rounds, the application of entangling gates leads to an explosion of the virtual dimensions between the ancilla and data qubits, using too much memory for consumer hardware.

8.4. **Results.**

8.4.1. *Channel Infidelities.* First we look at the channel infidelities for the different CNOTs we will use in the surface code. As a reference which can be understood somewhat analytically, we first look at $\text{CNOT}_{\text{toy1}}$ and $\text{CNOT}_{\text{toy2}}$, shown in Figure 21. Here we see the clear *linear* dependence of the infidelity on the underlying physical noise p_{CNOT} as predicted in subsubsection 8.2.3.

For noise applied throughout the CNOT operation — $\text{CNOT}_{\text{toy2}}$ — we find the channel infidelity being lower than for noise applied at the beginning as for $\text{CNOT}_{\text{toy1}}$.

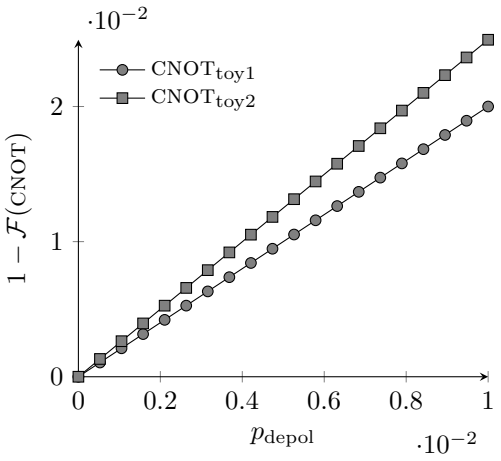
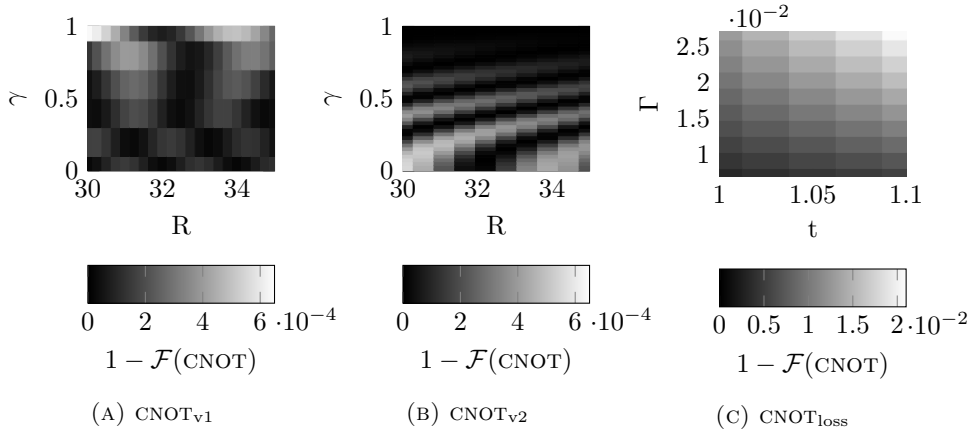


FIGURE 21. Channel infidelity as a function of the depolarizing noise parameter p_{CNOT} for the reference CNOTs.

For CNOT_{v1}, we look at the infidelity as a density plot over the two parameters we vary in Figure 22a. The parameter ranges have been chosen such that the infidelity depends on the parameters in a nontrivial way, i.e. reveals an interesting landscape.

The infidelity varies between 1.6×10^{-5} and 6.4×10^{-4} in the parameter range.

For the same parameters, which would correspond to the same physical setup, CNOT_{v2} shows a rather different dependence on the parameters as shown in Figure 22b



(A) CNOT_{v1} (B) CNOT_{v2} (C) $\text{CNOT}_{\text{loss}}$

FIGURE 22. Infidelities as a function of the parameters chosen for the different CNOTs we consider.

where we find values for the infidelity between 0 and 5.5×10^{-4}, overall clearly less than for CNOT_{v1}.

As a function of the parameters the infidelity for CNOT_{v2} is almost periodic with either fixed R or fixed γ.

Finally $\text{CNOT}_{\text{loss}}$ shows a very simple behaviour where the infidelity increases with both increased time t and increased decoherence Γ which is reasonable since both parameters increase the decoherence the system experiences.

8.4.2. *Code Performance for Initial Noise.* Next we look at the *code performance* which we measure by calculating the probability of wrong decoding as a function of the initial noise.

scenario I. We begin by considering scenario I.

For $\text{CNOT}_{\text{toy1}}$ the code performance as a function of the initial noise is shown in Figure 23. Since the plot for $\text{CNOT}_{\text{toy2}}$ is identical, we omit plotting it separately.

We see that the code performance responds quadratically to the initial noise, which matches our expectation that the terms linear in the initial noise, which correspond to just one error happening, can be corrected by the surface code.

The different curves correspond to different parameters p_{CNOT} of $\text{CNOT}_{\text{toy1}}$ and $\text{CNOT}_{\text{toy2}}$.

For CNOT_{v1}, CNOT_{v2} and $\text{CNOT}_{\text{loss}}$ we see the same quadratic dependence of the noise in Figure 24a, Figure 24b and Figure 24c where we plotted both the worst and best realizations of CNOTs considered.

The differences within a CNOT are minor except for $\text{CNOT}_{\text{loss}}$ which covers a comparatively greater range of channel fidelities. That seems to be reflected in a greater range of code performance.

scenario II. Next we can look at the same plots for scenario II where we start in a highly entangled stabilizer state instead of a product state.

The resulting plots for the code performance as a function of the initial noise are shown in Figure 25 for both $\text{CNOT}_{\text{toy1}}$ and $\text{CNOT}_{\text{toy2}}$.

The intersections with the y-axis are given by a term linear in the error probability of $\text{CNOT}_{\text{toy1}}$ derived in subsubsection 8.2.3 since no ancilla error is correctable. The dependence on the intial noise is linear first for $p_{\text{init}} < 4p_{\text{depol}}$ and then switches to quadratic.

This peculiar shape can be explained by considering the two plaquettes that border the logical operator and detect errors that change the logical state.

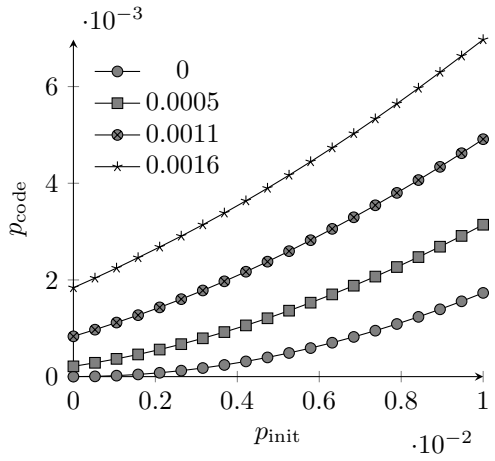
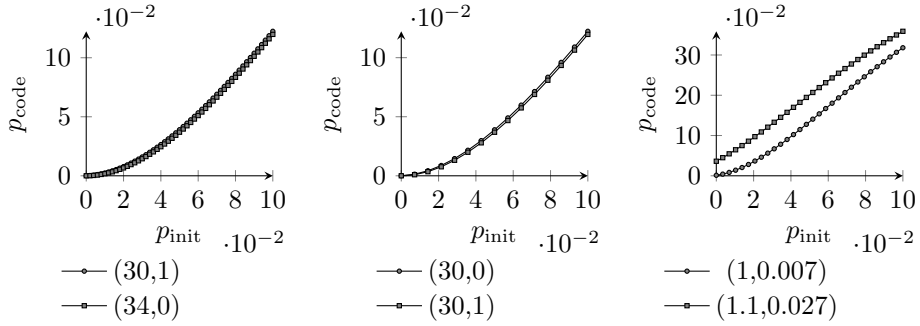


FIGURE 23. p_{code} is shown for $\text{CNOT}_{\text{toy1}}$ as a function of the initial noise p_{init} for different p_{CNOT}. As expected the code performance scales quadratically with the initial noise. The plots for $\text{CNOT}_{\text{toy2}}$ look identical and are thus not plotted separately.



(A) CNOT_{v1} of (R, γ) (B) CNOT_{v2} of (R, γ) (C) $\text{CNOT}_{\text{loss}}$ of (t, Γ)

FIGURE 24. For the different CNOT, both the lowest and highest value curves for p_{code} as a function of p_{init} are shown with their corresponding parameters for scenario I.

On these plaquettes, while the initial noise is low, a -1 syndrome is interpreted as a failure on the ancilla whose probability is held constant for a curve. But as the initial noise rises, an actual error on the data qubit becomes more likely and the interpretation is switched from ancilla to data error. As a consequence, the probability of the inverted choice, we have that for low p_{init} it is a data error, whose probability rises linearly, and as p_{init} increases it is interpreted as an ancilla error, whose probability is constant with an error probability quadratic in p_{init} superimposed.

For CNOT_{v1} in Figure 26a, these bumps are not visible, either due to them being less pronounced or because the plot is over a larger range of initial noise with less resolution. Either way we see a quadratic increase of the code error-probability as a function of initial noise although higher order corrections visibly start to contribute.

The same holds for CNOT_{v2} in Figure 26b while in Figure 26c the dependence on initial noise is similar as for the reference CNOTs, showing peculiar bumps likely due to interpretation switching.

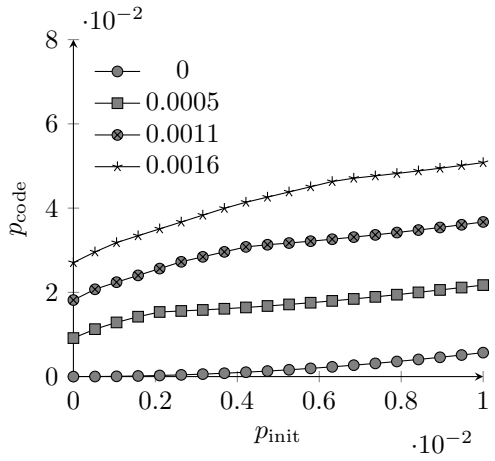
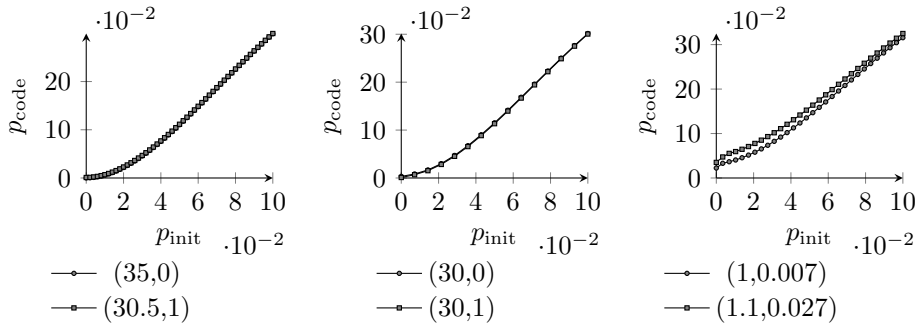


FIGURE 25. p_{code} as a function of p_{init} for different values of depolarization noise for scenario II.



(A) CNOT_{v1} of (R, γ) (B) CNOT_{v2} of (R, γ) (C) $\text{CNOT}_{\text{loss}}$ of (t, Γ)

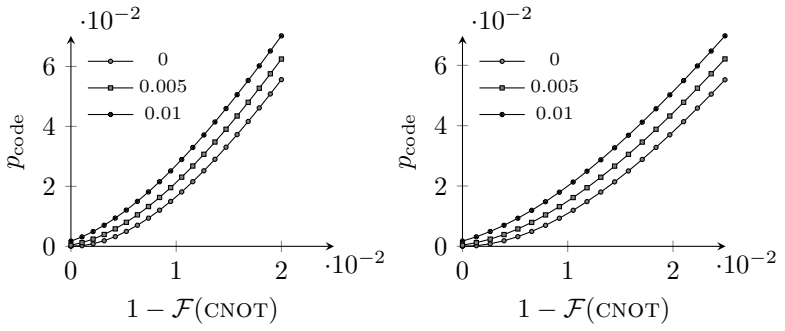
FIGURE 26. For the different CNOT, both the lowest and highest value curves for p_{code} as a function of p_{init} are shown with their corresponding parameters for scenario II.

8.4.3. *Code Performance for Channel Infidelities.* We have seen that the code performance as a function of the initial noise has the quadratic dependence we expect from $\mathcal{S}_{17}$. Next we look at the dependence of the code performance as a function of the channel infidelities.

For this purpose, we calculated the channel infidelities of all CNOTs that were used to calculate code performances and plot the code performance against the channel infidelity.

In Figure 27a, such a plot is shown for different p_{init} for $\text{CNOT}_{\text{toy1}}$ and $\text{CNOT}_{\text{toy2}}$. From the plots it is clear that $\text{CNOT}_{\text{toy1}}$ for a given noise level leads to the same code performance but has worse channel fidelity. This is reflected in the plot for $\text{CNOT}_{\text{toy2}}$ just being a stretched version of $\text{CNOT}_{\text{toy1}}$.

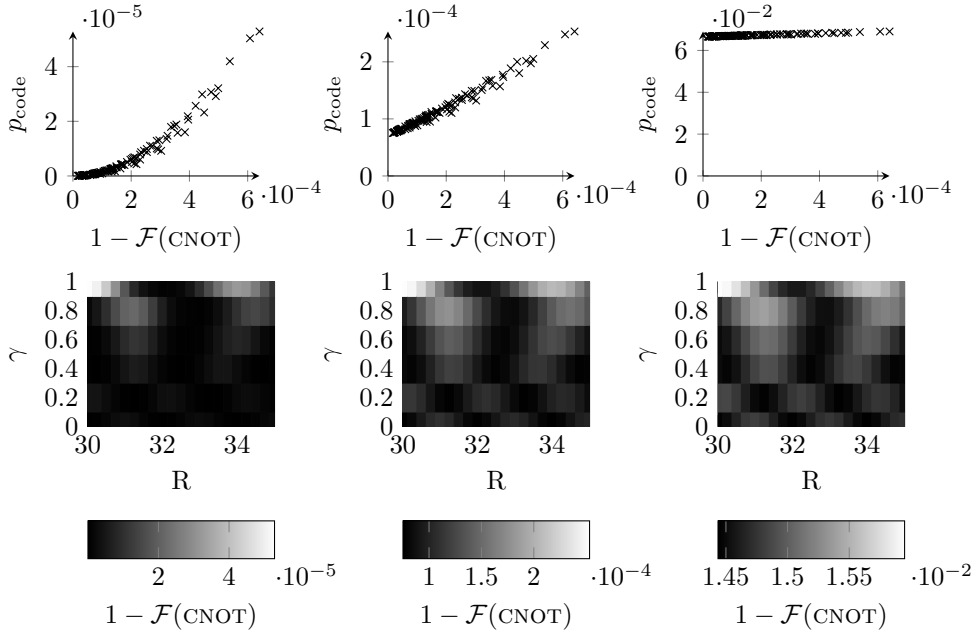
For CNOT_{v1}, we show results for different initial noise values in Figure 28. In the first row, scatterplots for the code performance as a function of the channel infidelity are shown. In the row below, the code performance as a function of the parameters R and γ are shown for the same initial noise as in the plot above. Comparing with Figure 22a, we see that the code performances show the same general features as the channel infidelity.



(A) $\text{CNOT}_{\text{toy1}}$ (B) $\text{CNOT}_{\text{toy2}}$

FIGURE 27. Code performance as a function of the channel infidelity for both $\text{CNOT}_{\text{toy1}}$ and $\text{CNOT}_{\text{toy2}}$ for scenario I. The plot for $\text{CNOT}_{\text{toy2}}$ is just a stretched version of the one for $\text{CNOT}_{\text{toy1}}$ since they both showed the same code performance before but different channel infidelities. The different curves correspond to different p_{init}.

And while the scatterplots imply a quadratic dependence for low initial noise going turning into no dependence for high noise — as the effect of the initial noise dominates the details of the CNOT — the plots show that different channel fidelities can lead to the same code performances.

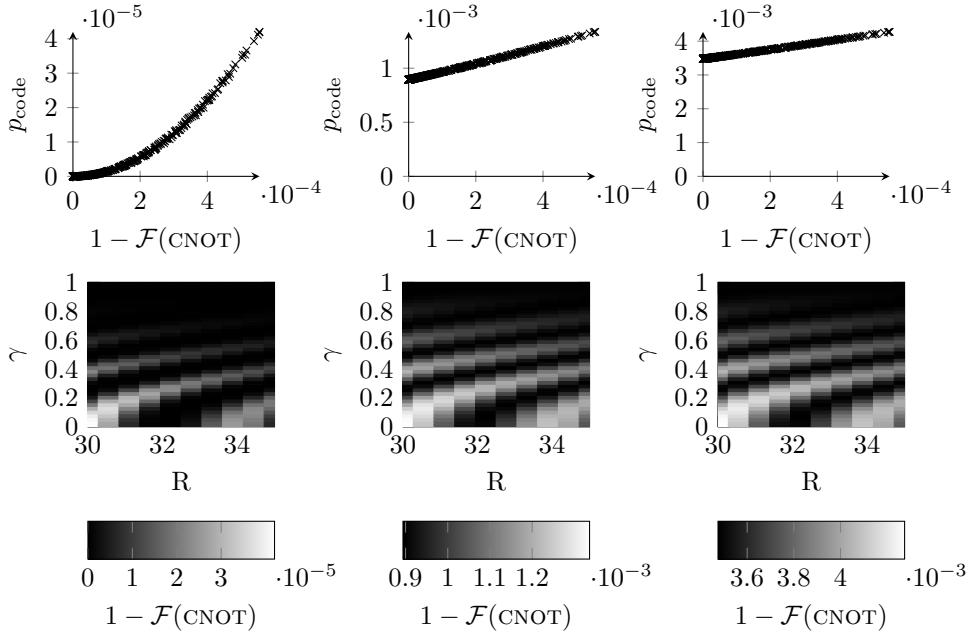


(A) $p_{\text{init}} = 0$ (B) $p_{\text{init}} = 0.002$ (C) $p_{\text{init}} = 0.07$

FIGURE 28. p_{code} as a function of the channel infidelity and the channel infidelity as a function of the parameters R and γ for CNOT_{v1} in scenario I.

The situation for CNOT_{v2}, shown in Figure 29, is similar in that the code performance shows the same features as the channel infidelity in Figure 22b. For the

same parameters, CNOT_{v2} shows much better performance than CNOT_{v1} and thus still has weak dependence on the channel infidelity even for $p_{\text{init}} = 0.07$. Comparing with CNOT_{v1} we also note that the spread of channel infidelity leading to the same code performance is negligible.



(A) $p_{\text{init}} = 0$ (B) $p_{\text{init}} = 0.007$ (C) $p_{\text{init}} = 0.014$

FIGURE 29. p_{code} as a function of the channel infidelity and the channel infidelity as a function of the parameters R and γ for CNOT_{v2} in scenario I.

For $\text{CNOT}_{\text{loss}}$we find almost no dependence on the initial noise in Figure 30. For all considered initial noise values, the plots show the same dependence, although for higher initial noise the code performance gets worse. As a function of the parameters, the general landscape of the code performance strongly mirrors the channel infidelity shown in Figure 22c.

Overall, the spread of channel infidelities leading to the same code performance seems quite small which would imply that the channel infidelity is indeed a good predictor for scenario I. Next we look at the same plots for scenario II.

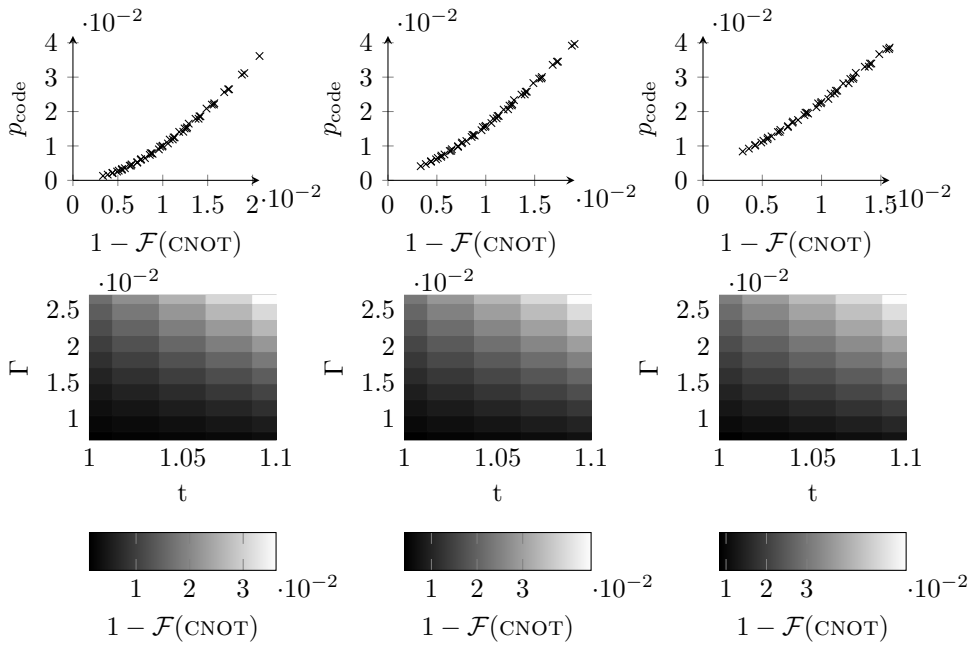
For scenario II, two plots for $\text{CNOT}_{\text{toy1}}$ and $\text{CNOT}_{\text{toy2}}$ are shown in Figure 31. Again, the code performance for a given parameter is the same whether we apply noise just before or throughout the CNOT but the channel fidelity is affected.

Thus for $\text{CNOT}_{\text{toy2}}$, the plot appears stretched in the x-direction compared to $\text{CNOT}_{\text{toy1}}$. The slope of the curve fits our expectation of linear dependence due to errors on ancillas not being correctable with another term superimposed taking into account that one data qubit error can be corrected but the initial noise also contributes to that, see subsubsection 8.2.3.

For CNOT_{v1} as in Figure 28, we see scatterplots for the code performance depending on the channel infidelity and the code performance depending on the CNOT parameters in Figure 32 for scenario II.

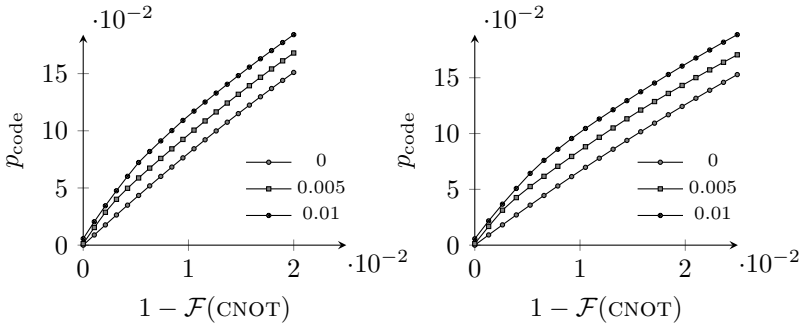
Compared to before, we see quite a different picture:

While the code performance as a function of parameters still mirrors the general dependence of the channel infidelity as a function of the parameters, we find a much



(A) $p_{\text{init}} = 0$ (B) $p_{\text{init}} = 0.003$ (C) $p_{\text{init}} = 0.007$

FIGURE 30. p_{code} as a function of the channel infidelity and the channel infidelity as a function of the parameters t and Γ for $\text{CNOT}_{\text{loss}}$ in scenario I.

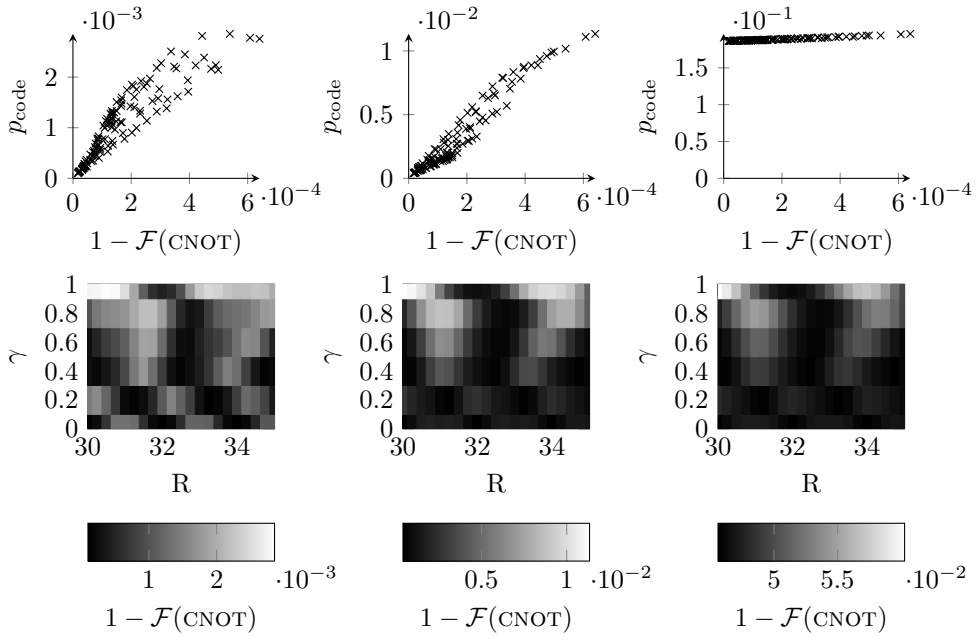


(A) $\text{CNOT}_{\text{toy1}}$ (B) $\text{CNOT}_{\text{toy2}}$

FIGURE 31. Code performance as a function of the channel infidelity for both $\text{CNOT}_{\text{toy1}}$ and $\text{CNOT}_{\text{toy2}}$ for scenario II. The plot for $\text{CNOT}_{\text{toy2}}$ is just a stretched version of the one for $\text{CNOT}_{\text{toy1}}$, since they both showed the same code performance before but different channel infidelities. The different curves correspond to different p_{init}.

greater spread of code performances for a small interval of channel infidelities and vice versa.

This effect is most pronounced for low initial noise. As the initial noise increases, the relative importance of the CNOT diminishes yet again and we recover a weak linear dependence on the channel fidelity for high initial noise.



(A) $p_{\text{init}} = 0$ (B) $p_{\text{init}} = 0.002$ (C) $p_{\text{init}} = 0.07$

FIGURE 32. p_{code} as a function of the channel infidelity and the channel infidelity as a function of the parameters R and γ for CNOT_{v1} in scenario II.

For CNOT_{v2} in Figure 33 we have a greater spread in of code performance for channel fidelities than in Figure 29 yet unlike CNOT_{v1} the greatest spread is not for vanishing initial noise but instead it reaches a maximum around $p_{\text{init}} \approx 0.007$.

Especially interesting is that in Figure 33b and Figure 33c it seems that the worst code performance is not achieved for the highest channel infidelities.

Looking at the code performance as a function of the parameters, we still get the same general structure as for the channel infidelity although we find that the regions of low error performance are slightly broader than the infidelity in the same parameter regions.

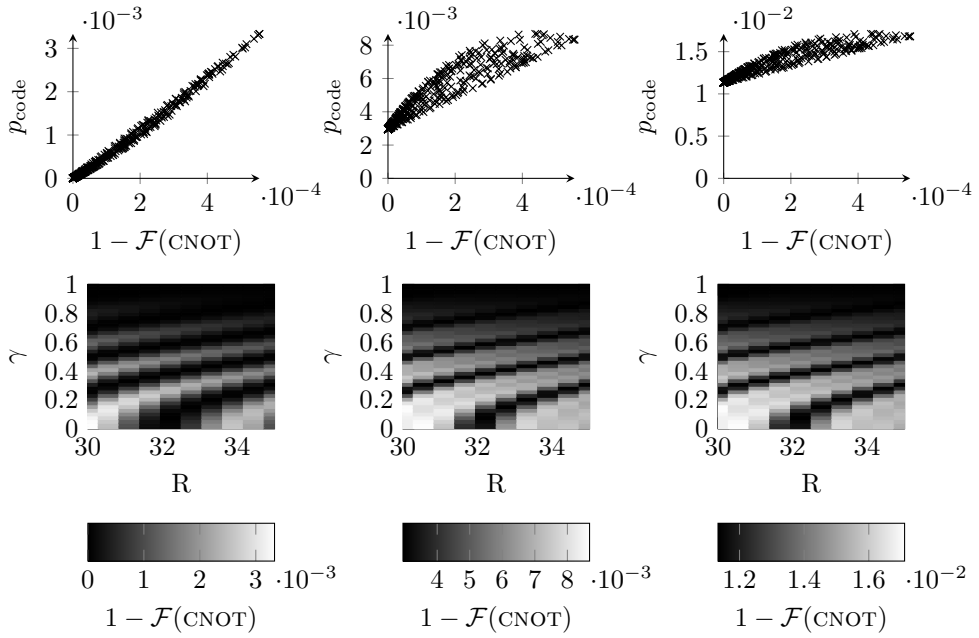
For $\text{CNOT}_{\text{loss}}$ we find, similar to the situation in Figure 30, that for scenario II in Figure 34the plots for different initial noises look pretty much the same. They show, as $\text{CNOT}_{\text{toy1}}$ and $\text{CNOT}_{\text{toy2}}$, mostly linear dependence on the channel fidelity with some nonlinear terms superimposed for non-vanishing initial noise.

Unlike CNOT_{v1} and CNOT_{v2}, $\text{CNOT}_{\text{loss}}$ does not show any spread of code performance for a given channel infidelity.

9. Conclusions

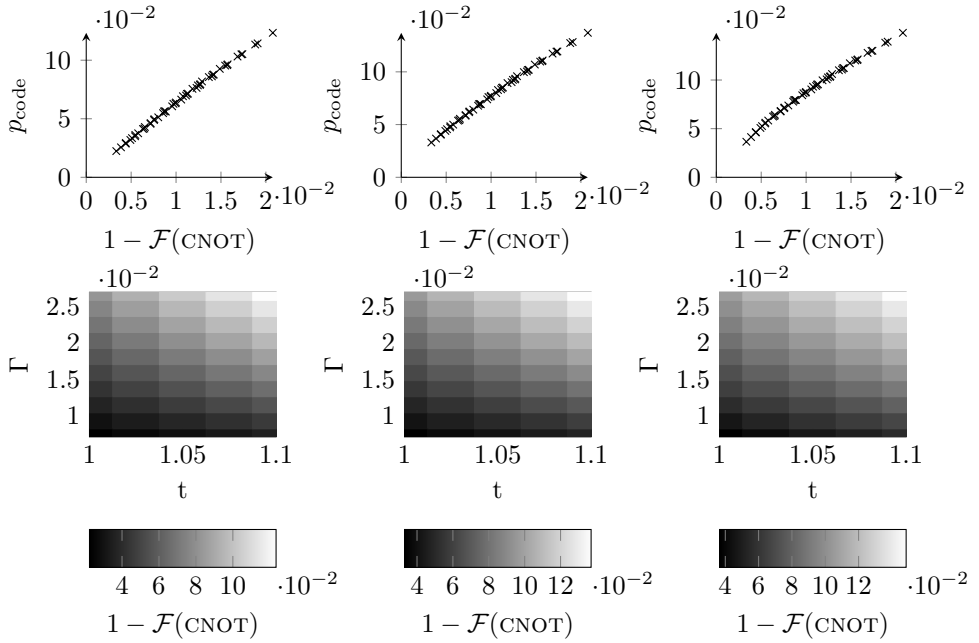
Using the tensor network approach, we were able to simulate a full 17-qubit surface code, namely $\mathcal{S}_{17}$, as an open quantum system with decoherent and coherent noise applied, on consumer computer hardware.

A drawback of the tensor network approach is that we are limited to one round of error correction since otherwise the entanglement becomes too strong and with the virtual dimensions of our tensors growing, the computational resources approach those of a normal description in terms of matrices and superoperators which require more resources than we have.



(A) $p_{\text{init}} = 0$ (B) $p_{\text{init}} = 0.007$ (C) $p_{\text{init}} = 0.014$

FIGURE 33. p_{code} as a function of the channel infidelity and the channel infidelity as a function of the parameters R and γ for CNOT_{v2} in scenario II.



(A) $p_{\text{init}} = 0$ (B) $p_{\text{init}} = 0.003$ (C) $p_{\text{init}} = 0.007$

FIGURE 34. p_{code} as a function of the channel infidelity and the channel infidelity as a function of the parameters t and Γ for $\text{CNOT}_{\text{loss}}$ in scenario II.

Nonetheless, the simulation offered important insights. We have seen that for both scenario I and scenario II the channel fidelity can be a good predictor for the code performance (for $\text{CNOT}_{\text{toy1}}$, $\text{CNOT}_{\text{toy2}}$ and $\text{CNOT}_{\text{loss}}$) but also a bad predictor (CNOT_{v1}, CNOT_{v2}). The CNOTs where the channel fidelity was a good predictor were based on perfect CNOTs with decoherent noise applied while the predictions were worse for the two floating gate CNOTs which were approximate unitary CNOTs. It thus seems that unitary effects might play an important role in surface codes which are not well captured by the channel fidelity.

Generally, the predictive power of the channel fidelity is worse for scenario II. There are two possibilities for why this is the case. The initial state being highly entangled might contribute to error configurations that are less likely for initial product states, which would imply a need for more analysis of error correction on entangled states. The other possibility is that accessing information about the syndrome without a final measurement, i.e. only by ancilla measurement which gets entangled via noisy CNOTs, increases the complexity of the effect that noisy CNOTs have on the code performance without depending directly on the channel fidelity we calculated.

Since many analyses use perfect CNOT or highly idealized versions, the somewhat more realistic analysis of the floating gate CNOTs shows that the idealized CNOTs can be misleading if one considers the channel fidelity as a measure of the usefulness of a CNOT.

Further investigations are required to set the connection between code performance and gate performance — measured in whatever way — onto a stronger, theoretical footing to pave the way for experimentalists to implement working surface codes and initiate a next phase in the building of a working quantum computer.

It seems though that for realistic systems an increase in the channel fidelity might not necessarily mean that a error correcting code performs better.

References

[1] Bravyi, S. B., and Kitaev, A. Y. Quantum codes on a lattice with boundary.

[2] Cormen, T. H. *Introduction to Algorithms*. MIT Press, 2009.

[3] Darmawan, A. S., and Poulin, D. Tensor-network simulations of the surface code under realistic noise. *arXiv preprint arXiv:1607.06460* (2016).

[4] Dennis, E., Kitaev, A., Landahl, A., and Preskill, J. Topological quantum memory. *Journal of Mathematical Physics 43*, 9 (sep 2001), 4452–4505.

[5] DiVincenzo, D. P. The physical implementation of quantum computation. *Fortschritte der Physik 48*, 9-11 (2000), 771–783.

[6] Dutta, O., Tagliacozzo, L., Lewenstein, M., and Zakrzewski, J. Toolbox for abelian lattice gauge theories with synthetic matter. *Phys. Rev. A 95* (May 2017), 053608.

[7] Eckart, C., and Young, G. The approximation of one matrix by another of lower rank. *Psychometrika 1*, 3 (1936), 211–218.

[8] Edmonds, J. Maximum matching and a polyhedron with 0, 1-vertices. *Journal of Research of the National Bureau of Standards 69B*, 1 and two (January-June 1965).

[9] Edmonds, J. Paths, trees, and flowers. *Journal canadien de mathématiques 17*, 0 (jan 1965), 449–467.

[10] Etienne, B., and Paris, E. Two-dimensional electron gas of very high mobility in planar doped heterostructures. *Journal de Physique 48*, 12 (1987), 2049–2052.

[11] Fowler, A. G. Coping with qubit leakage in topological codes. *Physical Review A 88*, 4 (oct 2013).

[12] Fowler, A. G., Mariantoni, M., Martinis, J. M., and Cleland, A. N. Surface codes: Towards practical large-scale quantum computation. *Phys. Rev. A 86* (Sep 2012), 032324.

[13] Golovach, V. N., Borhani, M., and Loss, D. Electric-dipole-induced spin resonance in quantum dots. *Physical Review B 74*, 16 (oct 2006).

[14] Gottesman, D. Stabilizer codes and quantum error correction. phdthesis, Caltech, May 1997.

[15] Grover, L. K. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC 96* (1996), ACM Press.

[16] Hastings, M. B. An area law for one-dimensional quantum systems. *Journal of Statistical Mechanics: Theory and Experiment 2007*, 08 (2007), P08024.
[17] Holmes, M., and Gray, A. Fast svd for large-scale matrices. In *Workshop on Efficient Machine Learning at NIPS* (2007).
[18] Jordan, J., Orús, R., and Vidal, G. Numerical study of the hard-core bose-hubbard model on an infinite square lattice. *Phys. Rev. B 79* (May 2009), 174515.
[19] Kitaev, A. Yu. Fault tolerant quantum computation by anyons. *Annals Phys. 303*, 1 (jan 2003), 2–30.
[20] Kloeffel, C., and Loss, D. Prospects for spin-based quantum computing in quantum dots. *Annual Review of Condensed Matter Physics 4*, 1 (apr 2013), 51–81.
[21] Kouwenhoven, L. P., Marcus, C. M., McEuen, P. L., Tarucha, S., Westervelt, R. M., and Wingreen, N. S. Electron transport in quantum dots. In *Mesoscopic electron transport*. Springer, 1997, pp. 105–214.
[22] Loss, D., and DiVincenzo, D. P. Quantum computation with quantum dots. *Physical Review A 57*, 1 (jan 1998), 120–126.
[23] Monthus, C. Dissipative random quantum spin chain with boundary-driving and bulk-dephasing: magnetization and current statistics in the non-equilibrium-steady-state. *Journal of Statistical Mechanics: Theory and Experiment 2017*, 4 (apr 2017), 043302.
[24] Nickerson, N. H. Error correcting power of small topological codes. *arXiv preprint arXiv:1609.01753* (2016).
[25] Nielsen, M., and Chuang, I. *Quantum Computation and Quantum Information*. Cambridge Series on Information and the Natural Sciences. Cambridge University Press, 2000.
[26] O'Brien, T. E., Tarasinski, B., and DiCarlo, L. Density-matrix simulation of small surface codes under current and projected experimental noise.
[27] Orús, R. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Annals of Physics 349* (2014), 117 – 158.
[28] Penrose, R. Applications of negative dimensional tensors. *Combinatorial Mathematics and its Applications* (1971).
[29] Pfeifer, R. N. C., Haegeman, J., and Verstraete, F. Faster identification of optimal contraction sequences for tensor networks. *Phys. Rev. E 90* (Sep 2014), 033315.
[30] Schuch, N., Cirac, I., and Perez-Garcia, D. Peps as ground states: Degeneracy and topology. *Annals of Physics 325*, 10 (2010), 2153 – 2192.
[31] Schuch, N., Pérez-Garcia, D., and Cirac, I. Classifying quantum phases using matrix product states and projected entangled pair states. *Phys. Rev. B 84* (Oct 2011), 165139.
[32] Schwabl, F. *Quantenmechanik für Fortgeschrittene*. Springer, 2008.
[33] Shor, P. W. Scheme for reducing decoherence in quantum computer memory. *Phys. Rev. A 52* (Oct 1995), R2493–R2496.
[34] Shor, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing 26*, 5 (oct 1997), 1484–1509.
[35] Tagliacozzo, L., Celi, A., and Lewenstein, M. Tensor networks for lattice gauge theories with continuous groups. *Phys. Rev. X 4* (Nov 2014), 041024.
[36] Tomita, Y., and Svore, K. M. Low-distance surface codes under realistic quantum noise. *Physical Review A 90*, 6 (dec 2014).
[37] Trif, M., Golovach, V. N., and Loss, D. Spin-spin coupling in electrostatically coupled quantum dots. *Physical Review B 75*, 8 (feb 2007).
[38] Trifunovic, L., Dial, O., Trif, M., Wootton, J. R., Abebe, R., Yacoby, A., and Loss, D. Long-distance spin-spin coupling via floating gates. *Physical Review X 2*, 1 (jan 2012).
[39] Vanderstraeten, L., Haegeman, J., Corboz, P., and Verstraete, F. Gradient methods for variational optimization of projected entangled-pair states. *Phys. Rev. B 94* (Oct 2016), 155123.
[40] Verstraete, F., Murg, V., and Cirac, J. Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems. *Advances in Physics 57*, 2 (2008), 143–224.
[41] Vidal, G. Efficient simulation of one-dimensional quantum many-body systems. *Phys. Rev. Lett. 93* (Jul 2004), 040502.
[42] Wen, X.-G. Quantum orders in an exact soluble model. *Phys. Rev. Lett. 90* (Jan 2003), 016803.
[43] Wiseman, H. M., and Milburn, G. J. *Quantum measurement and control*. Cambridge university press, 2009.
[44] Zumbühl, D. M. *Coherence and Spin in GaAs Quantum Dots*. phdthesis, Harvard University, September 2004.