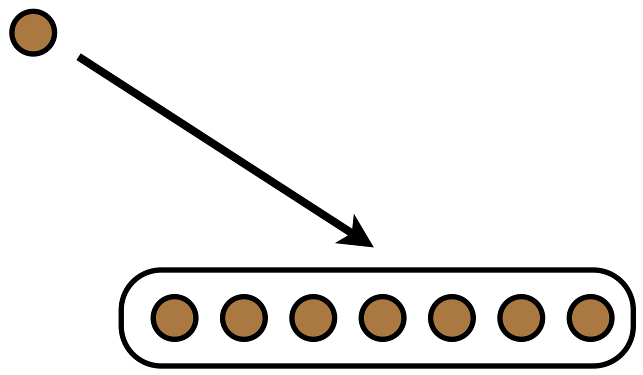
What is the Overhead Required for Fault Tolerance?

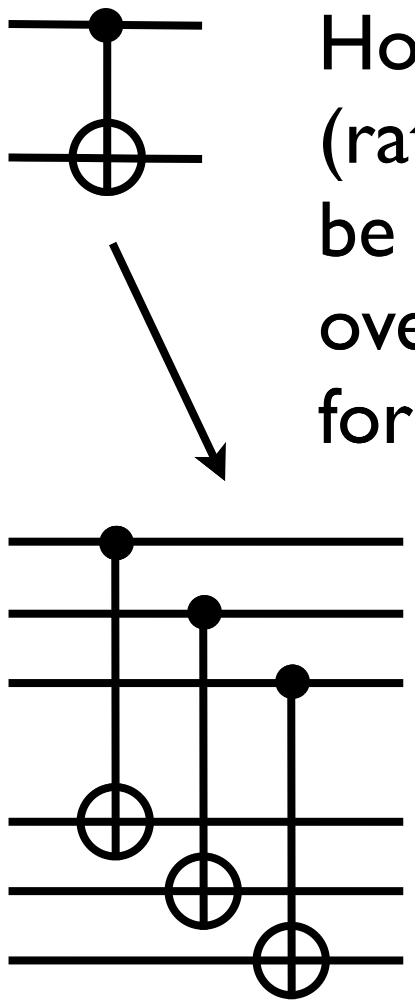
Daniel Gottesman
Perimeter Institute

arXiv:1310.2984

Fault Tolerance and Overhead

In order to build a large quantum computer, we will probably need to encode it using a fault-tolerant protocol. The logical qubits of the circuit are encoded using a quantum error-correcting code (QECC); each logical gate is replaced by a fault-tolerant gadget.





However, fault-tolerant protocols often have a high overhead (ratio physical qubits/logical qubits). A factor of 100 would be considered low overhead by today's standards. Also, overhead depends on the size of the computation: $O(\log^a T)$ for a computation with T locations.

How small can the overhead be?
Answer: Arbitrarily small constant*

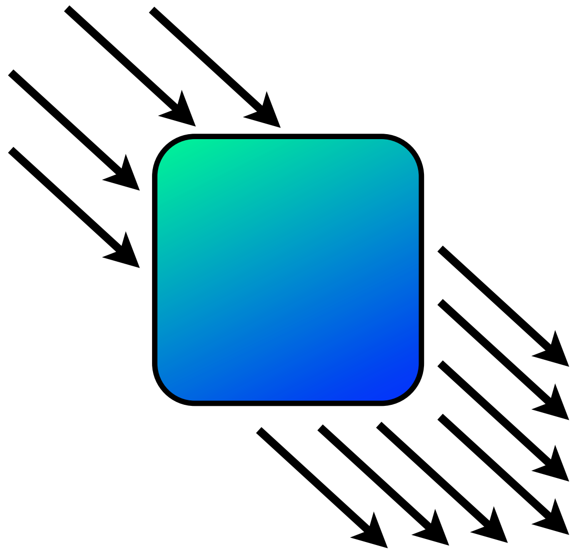
* For error rates below a threshold. LDPC code sold separately. Not available for all implementations. Additional restrictions may apply. If errors remain, apply another fault-tolerant protocol.

The Issues and Solutions

Why do previous protocols require extra overhead?

- Problem: A QECC of fixed size will always fail with a constant probability. Therefore large computations need large code blocks to lower the logical error rate.

But note that no tradeoff is required for channel capacity: encode k logical qubits in n physical qubits at a constant rate R=k/n with logical error rate exp(-O(n)).



Solution: Encode many qubits per block.

- Problem: Large block codes are hard to error correct.

Solution: Use quantum LDPC (Low-density parity check) codes.

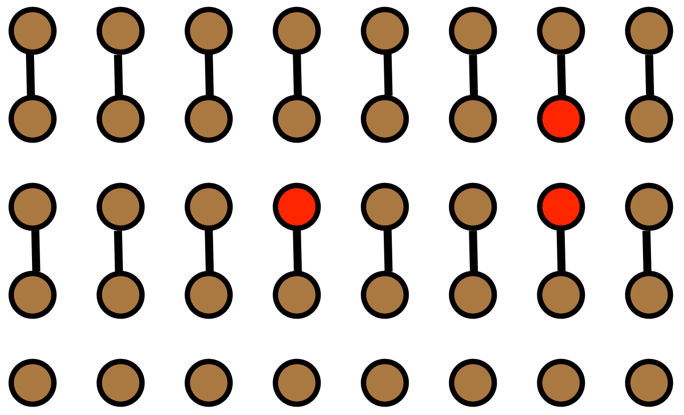
- Problem: Ancillas for large blocks are hard to create.

Solution: Use blocks that are large but not too large. Perform logical gates sequentially.

Basic Model of Fault Tolerance

What assumptions will I make about the system?

- Local stochastic noise
- No leakage errors
- Parallel gates
- Fresh ancilla qubits
- Long-range gates
- Fast and reliable classical computation

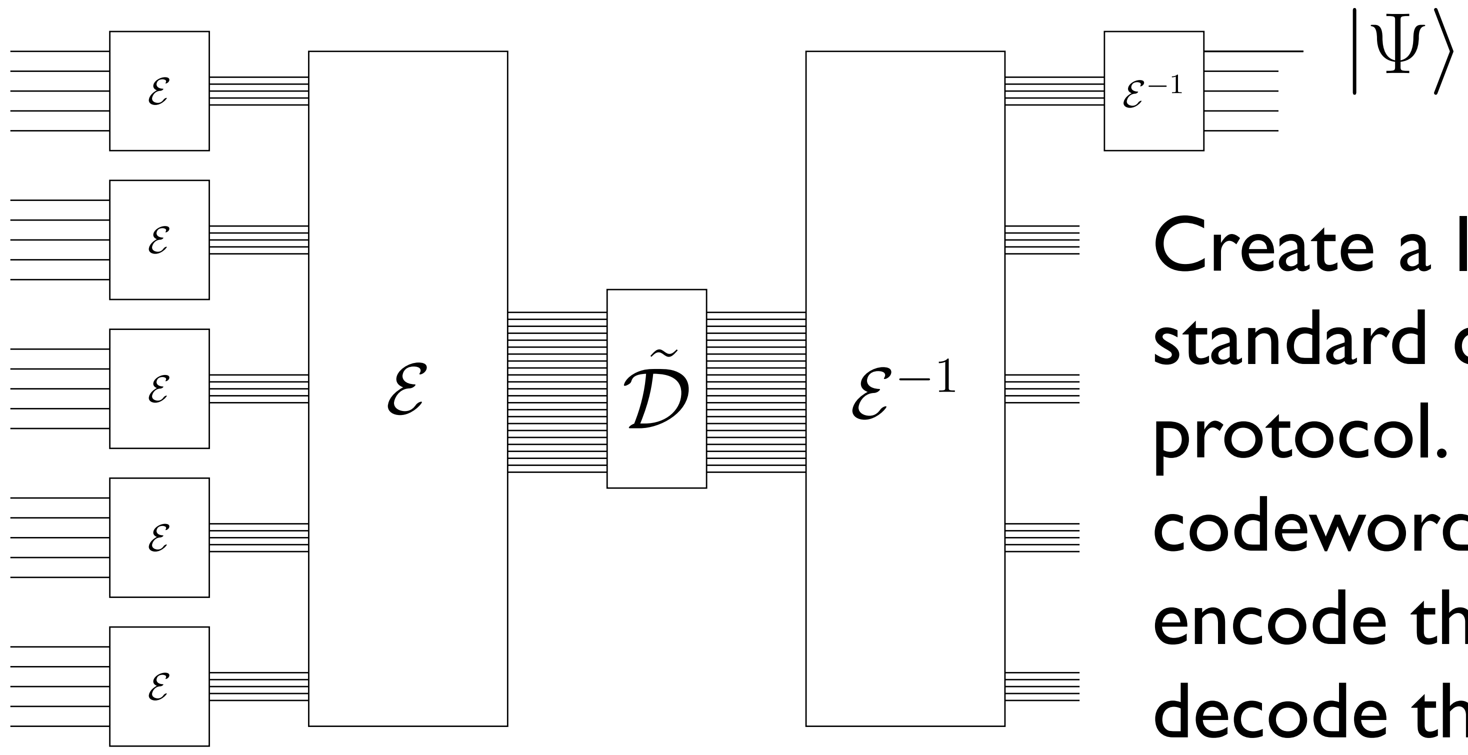


The last two assumptions are not needed to have a threshold, but are necessary for my result.

Note that I am assuming any amount of classical computation is free, but to make this reasonable, we should count the amount of classical computation needed and make sure it is not too big. This becomes an issue as some of the interesting quantum error-correcting codes do not have a good decoding algorithm known.

Making Ancillas

We can perform gates by gate teleportation, but this requires large ancillas the size of a code block. In any case, we need to be able to create new code blocks to start the computation.



Create a large code state using the standard concatenated code FT protocol. I.e., create concatenated codewords, run a logical circuit to encode the desired state, then decode the concatenated code.

However, if the size of the code block is n physical qubits and the encoding circuit uses n^2 gates, the total number of physical qubits used to create one code block is $O(n \log^a n)$ for some a. This is too many if we encode all logical qubits in a single block.

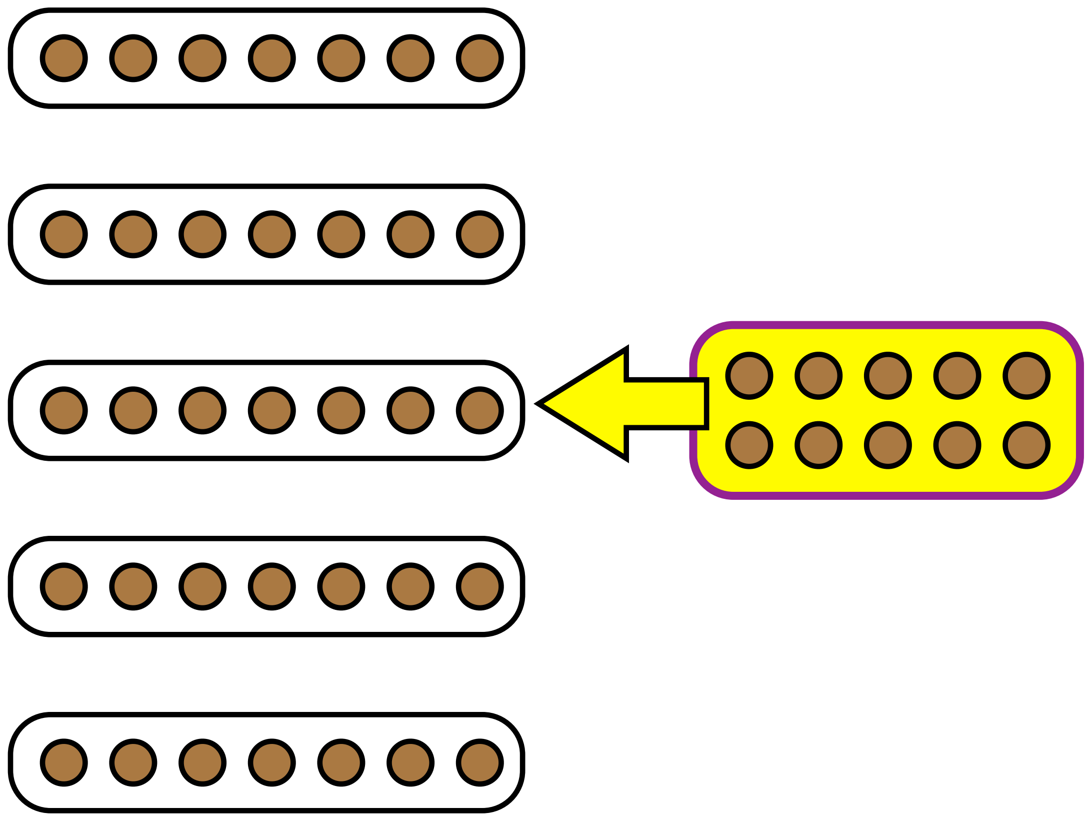
Don’t Put All Qubits in One Basket

But we don’t have to encode all logical qubits in a single block.

Suppose the ideal circuit involves K logical qubits. Let each code block have size $n \leqslant K/(R \log^{a+1} K)$, R the constant rate of the code. Then the total number of qubits used to make one ancilla block is

$$n \log^a n \leqslant (K/R) \log^{-1} K + \ldots = o(K/R).$$

- We can make ancillas using asymptotically negligible overhead.
- But we can only make one at a time.



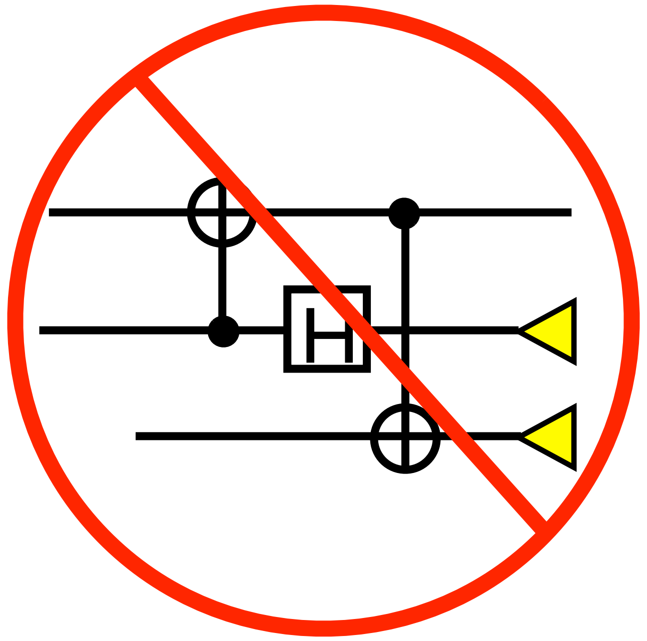
This is sufficient for a universal set of fault-tolerant gates.

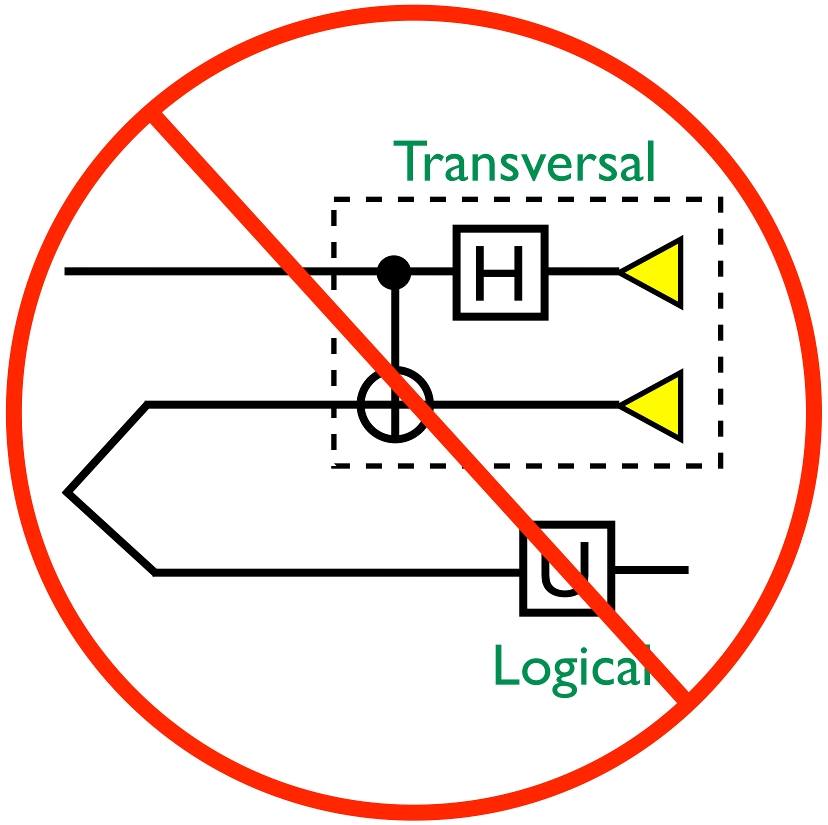
There will be a total of about K/nR blocks in the computation.

Fault-Tolerant Error Correction

There are three standard methods of fault-tolerant error correction:

- Shor FT EC
- Steane FT EC (for CSS codes)
- Knill FT EC





Steane and Knill FT EC use ancilla states which are code blocks in a particular logical state. If we did this, we could only do error correction on one block at a time. This could not correct storage errors.

Shor EC uses cat states of size equal to the weight of the stabilizer generators.

→ Use Shor EC and LDPC codes.

The Perfect Code Family

We want a code family with the following properties:

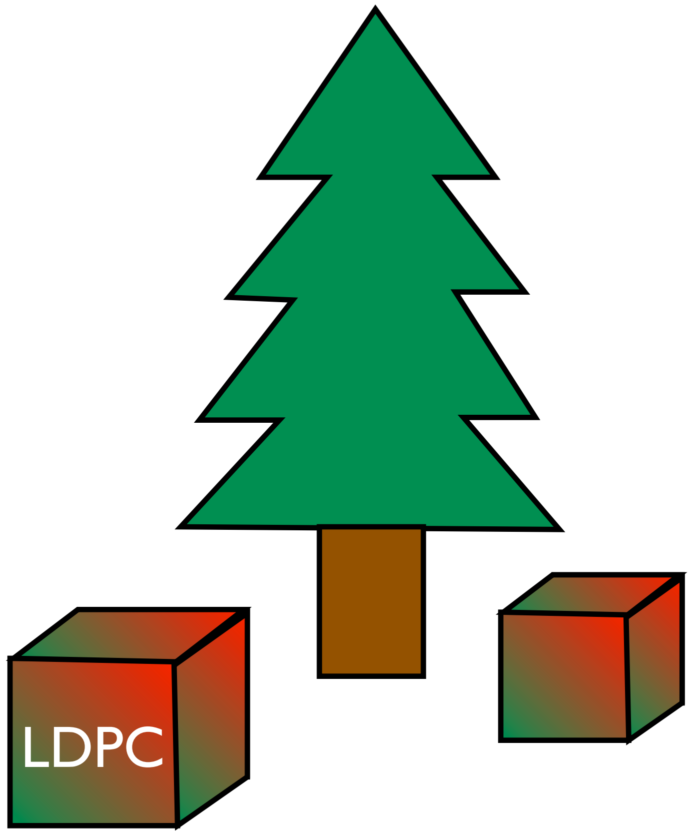
(i) Low density parity check codes.

(ii) Constant rate R as $n \rightarrow \infty$.

(iii) Possible values of n not too rare.

(iv) Corrects a constant fraction of likely errors (prob. p per qubit), including syndrome errors, with logical error suppression as $1/g(n)$.

(v) Efficient decoding algorithm.



Then:

There exists a threshold error rate for length $o(g(K))$ computations, with overhead arbitrarily close to $1/R$.

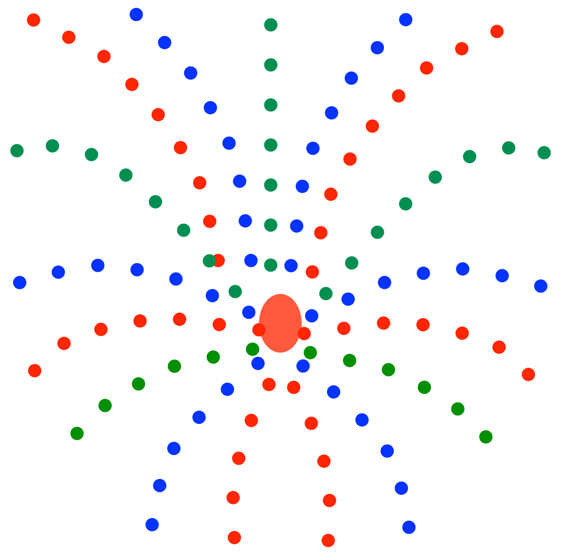
The (Almost) Precise Theorem

Specifically, given a code family satisfying (i)-(iv):

- We have some $\alpha < 1$.
- The computation uses K logical qubits and uses a sequential circuit with f(K) locations, with $f(K) = o(g(K^\alpha))$.
- We wish to achieve overhead within a factor η of the rate of the code.
- We want overall logical error probability less than ϵ.

Then:

- There exists a threshold error rate $p_t(\eta)$.
- There exists a threshold size $K_0(\eta, f, \epsilon)$.
- If physical error rate $p < p_t$ and $K > K_0$, then a fault-tolerant circuit simulation exists with overall error $<\epsilon$ and overhead $<\eta/R$.

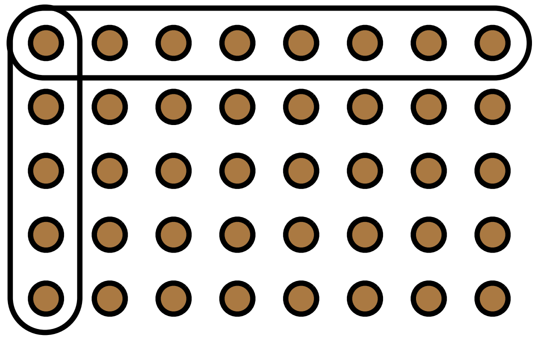


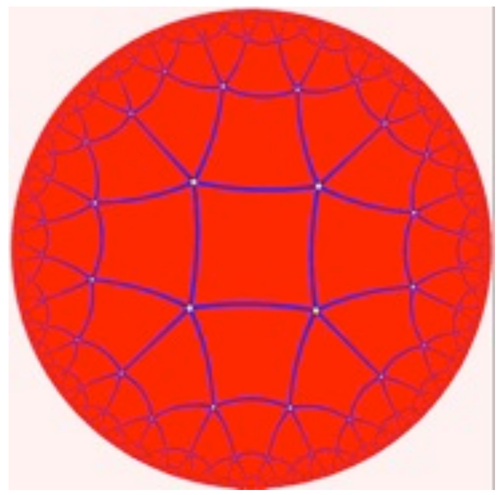
If (v) is also true (efficient decoding), then the FT simulation uses only polynomial classical computation.

Shopping for LDPC Codes

An (r-c)-LDPC code is a code where each stabilizer generator has weight at most r and each qubit is involved in at most c generators. We want LDPC codes with a constant rate R but that are still good at correcting errors. There are a limited set of known choices:

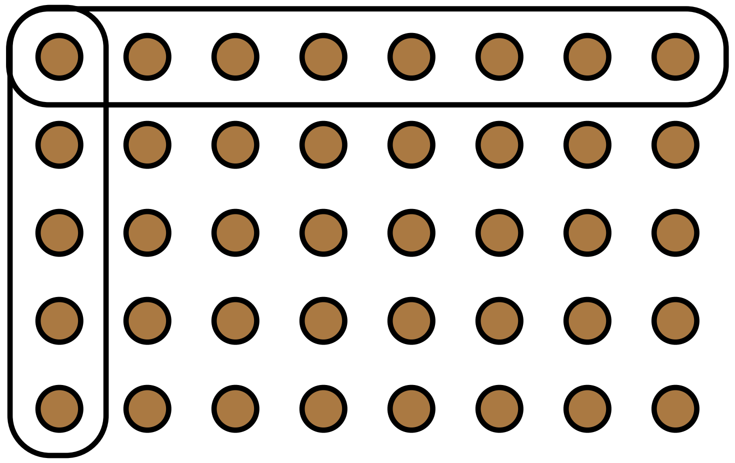
Code Family	Distance	Decoding
Hypergraph product codes	$n^{1/2}$	Exponential
Hyperbolic surface codes	Log n	Polynomial
4-D hyperbolic homology codes	n^{ε}	Exponential?
4-D hyp. with Hasting's alg.	Log n	Polynomial





Hypergraph Product Codes

The hypergraph product code construction (Tillich, Zemor, arXiv:0903.0566) takes two classical codes and produces a quantum code. If the classical codes are LDPC, the quantum code is too.



Using the same classical LDPC code twice, we get

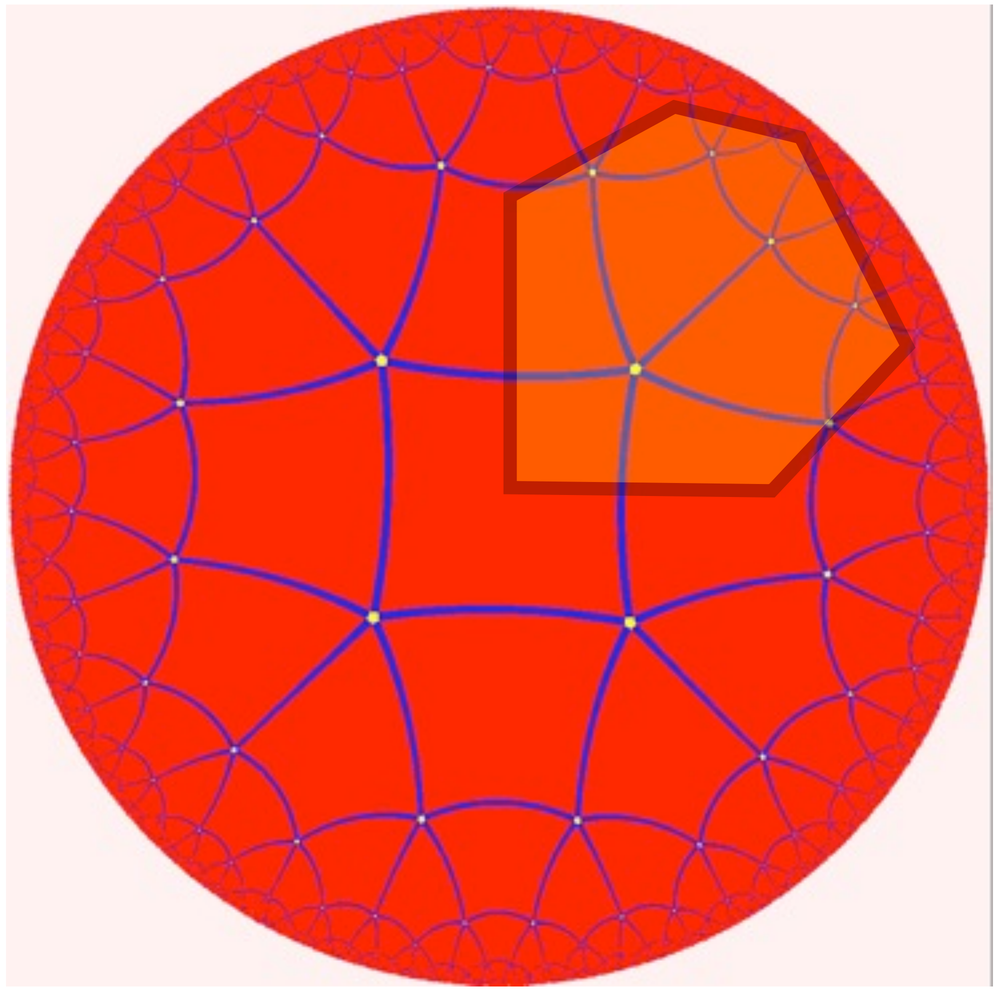
Classical [n,k,d] $\Rightarrow$ Quantum $[[n^2+(n-k)^2, k^2, d]]$

In particular, if k/n and d/n = constant, we have a good classical LDPC code, and the quantum code has rate R constant and distance $d = O(n^{1/2})$. In fact, we can choose a classical code with k/n close to 1 and make R as close to 1 as desired.

Unfortunately, the hypergraph product codes have no known efficient syndrome decoding algorithm, even when the classical code has one.

Hyperbolic Surface Codes

Freedman, Meyer, and Luo (2002) considered surface codes (like the toric code) on a hyperbolic manifold.



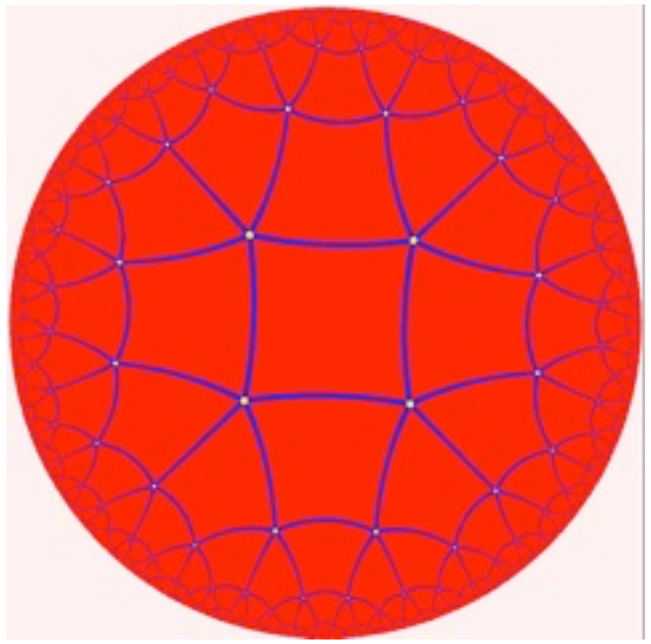
Hyperbolic space has the property that the volume of a ball is exponential in the radius. We can consider the hyperbolic plane modded out by some group Γ. This gives a compact space with some genus.

The number of logical qubits is given by twice the genus. The distance is the size of the smallest non-trivial cycle. There exist manifolds such that $d = O(\log n)$, $k = O(n)$.

An error produces a pair of point defects. Decoding can be done via matching defects, as with the toric code.

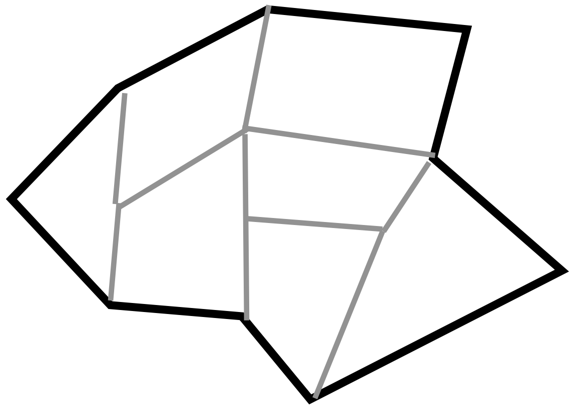
4-D Hyperbolic Codes

It is also possible to consider the 4-D toric code construction on 4-dimensional hyperbolic manifolds. Again, by choosing an appropriate manifold, we can have the number of encoded qubits $k = O(n)$ and the length of a non-trivial loop at least $O(\log n)$.



In 4 dimensions, errors cause defects which are string loops. Because this is a hyperbolic space, n^ε errors are needed to get a loop of radius log n. Therefore, the distance of the code is n^ε, $\varepsilon < 0.3$.

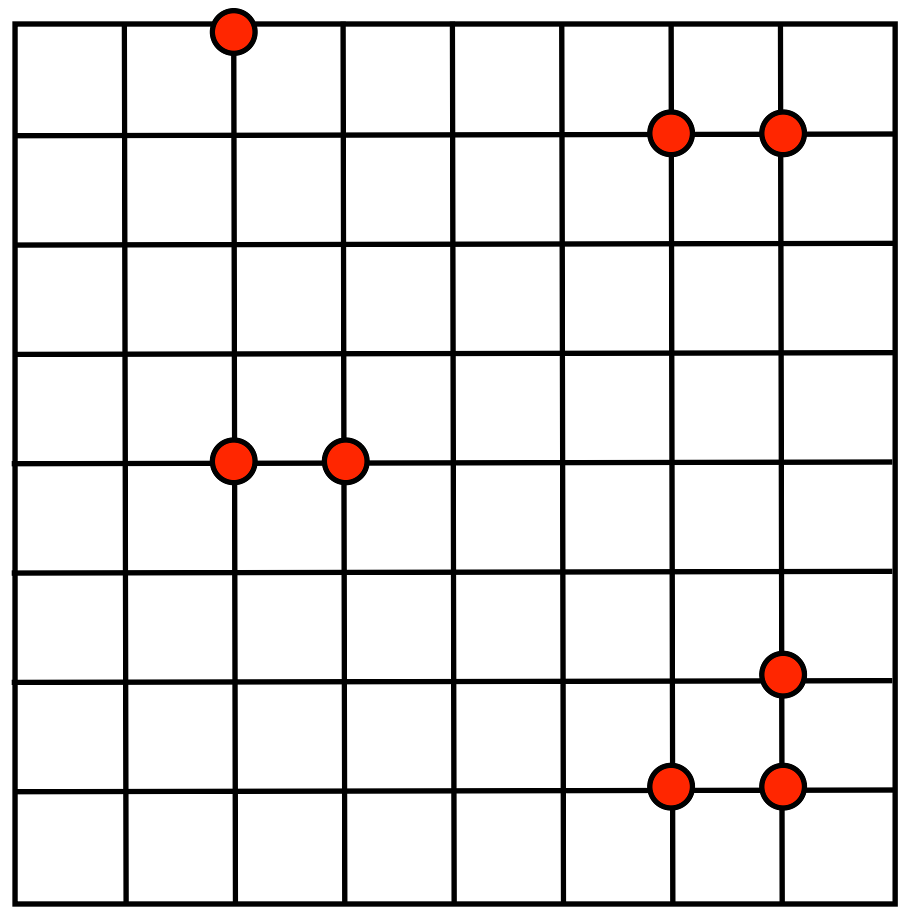
(Guth, Lubotzky, arXiv:1310.5555)



It is not clear if there is an efficient decoding algorithm for the full distance of these 4-D hyperbolic codes, but Hastings has a polynomial suboptimal decoding algorithm.

Typical Errors

All of these codes have sublinear distance. Luckily, that is sufficient to correct typical errors when each qubit has an error with prob. p.



A surface code can correct errors at a constant error rate despite sublinear distance because when the error rate is below the percolation threshold, errors tend to form small clusters which can be corrected separately.

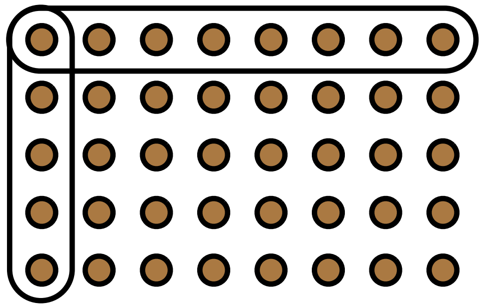
Kovalev and Pryadko showed that the same is true of general quantum LDPC codes. (arXiv:1208.2317)

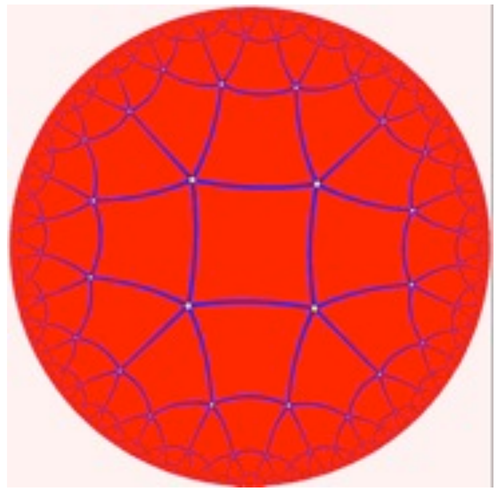
For fault tolerance, we need robust decoding, when there can be errors in the measured syndromes. This can be done by adding an extra dimension to the graph of the code to represent time. A similar percolation argument lets us locate both syndrome and data errors.

Logical Error Suppression

Percolation argument tells us that an LDPC code with distance d has logical error rate $(p/p_t)^{O(d)}$ under minimum-weight decoding with threshold p_t.

- Hypergraph product codes: logical error rate $\exp(-n^{1/2})$, but inefficient decoding.
- Hyperbolic surface codes: logical error rate $(p/p_t)^{O(\log n)} = n^{-O(\log p_t/p)}$ and efficient decoding.
- 4-D hyperbolic codes: logical error rate $\exp(-n^{\varepsilon})$, but optimal decoding may be inefficient.
  - However, Hastings (arXiv:1312.2546) has given a robust decoding algorithm that has depth $\log n$ and suppresses logical error rate as at least $n^{-O(\log p_t/p)}$, possibly better.





Polynomial error suppression can be made into any polynomial rate n^{-s} at the cost of choosing p to be sufficiently far below p_t.

Final Words

More fine print enlarged:

- Only works for a sequential circuit. Thus the depth blow-up could be O(K) if the original circuit is parallelized.
- Only works for sufficiently large computations. We need time for the efficient coding to kick in, and there are sub-leading terms.
- Non-local gates and fast classical computation seem necessary.

Conclusions:

- There is no real limit to the overhead needed for fault tolerance. We should seek practical protocols much better than current ones.
- Better LDPC codes are needed which give exponential error suppression and efficient decoding.
- Perhaps better protocols could lead to overhead reduction for small computations.