Cat-Optimized RS and Quantum Tornado Error-Correction

Team YQuambinator
Cheng-You Ho, Daniel Wang, Henry Ng, Justin Luo, Simba Shi

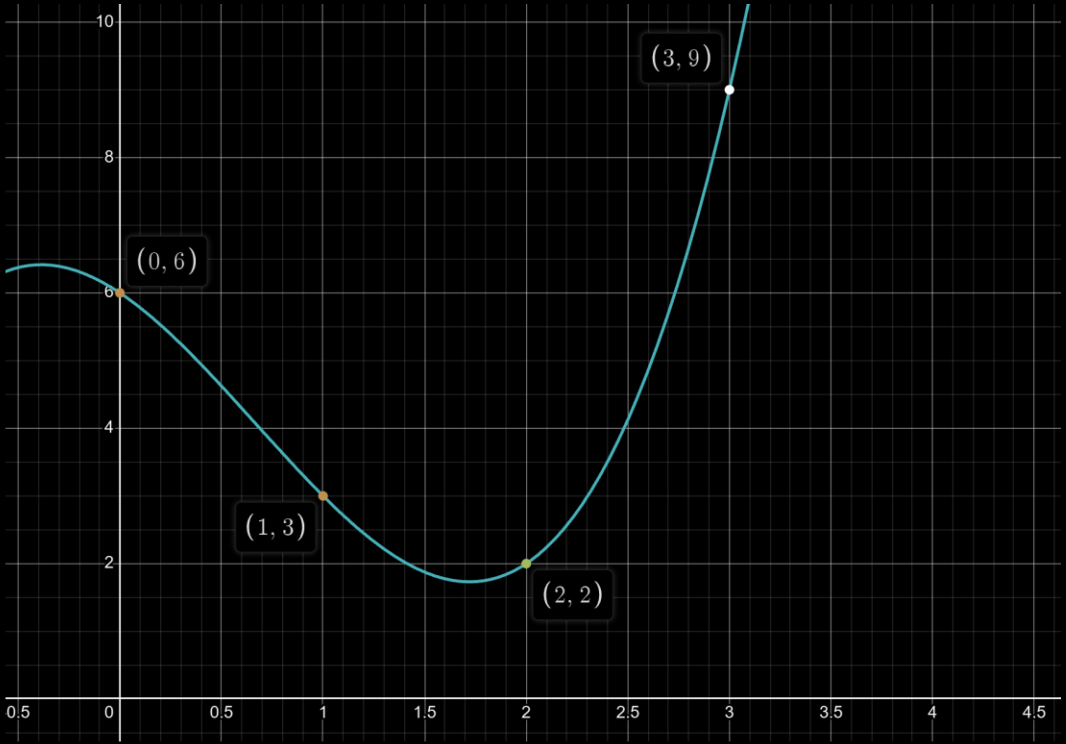
Alice & Bob iQuHack 2026

Classical Reed–Solomon Error Correction

$$m = (m_0, m_1, m_2, m_3)$$

$$f(x) = m_0 + m_1 x + m_2 x^2 + m_3 x^3$$

$$f(x) = m_0 + m_1 x + m_2 x^2 + m_3 x^3$$



$$GF(8)$$

$GF(8)$

A field with elements $\{0, 1, 2, 3, 4, 5, 6, 7\}$ and operations $+$ and $\cdot$ similar to polynomials of degree ≤ 2 with coefficients mod 2.

$GF(8)$

A field with elements $\{0, 1, 2, 3, 4, 5, 6, 7\}$ and operations $+$ and $\cdot$ similar to polynomials of degree ≤ 2 with coefficients mod 2.

$$5 = 101$$

$$1x^2 + 0x + 1$$

$GF(8)$

A field with elements $\{0, 1, 2, 3, 4, 5, 6, 7\}$ and operations $+$ and $\cdot$ similar to polynomials of degree ≤ 2 with coefficients mod 2.

Ex.

$$3 \cdot 3 = 011 \cdot 011 = (x+1)(x+1) = x^2 + 2x + 1 = x^2 + 1 = 101 = 5$$

$$3 + 3 = 011 + 011 = (x+1) + (x+1) = 2x + 2 = 000 = 0$$

$GF(8)$

Primitive element: a magic number which, when self-multiplied, generates every non-zero element in the field.

$$1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 6 \rightarrow 7 \rightarrow 5 \rightarrow 2$$

$$f(x) = m_0 x^0 + m_1 x + m_2 x^2 + m_3 x^3$$

$$[1 \;\; 2 \;\; 4 \;\; 3 \;\; 6 \;\; 7 \;\; 5]$$

$$f(x) = m_0 x^0 + m_1 x + m_2 x^2 + m_3 x^3$$

$$[1 \;\; 2 \;\; 4 \;\; 3 \;\; 6 \;\; 7 \;\; 5]$$

$$f(x) = m_0 x^0 + m_1 x + m_2 x^2 + m_3 x^3$$

1	1	1	1	1	1	1
1	2	4	3	6	7	5
1	4	6	5	2	3	7
1	3	5	4	7	2	6

$$f(x) = m_0 x^0 + m_1 x + m_2 x^2 + m_3 x^3$$

$$G_{sym} = \begin{pmatrix} x^0 \\ x^1 \\ x^2 \\ x^3 \end{pmatrix} = \begin{matrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 3 & 6 & 7 & 5 \\ 1 & 4 & 6 & 5 & 2 & 3 & 7 \\ 1 & 3 & 5 & 4 & 7 & 2 & 6 \end{matrix}$$

G_{sym} encodes a 4-digit message m into a 7-digit code c.

$$m \cdot G = (m_0\ m_1\ m_2\ m_3) \cdot \begin{pmatrix} x^0 \\ x^1 \\ x^2 \\ x^3 \end{pmatrix} = f(x) = c$$

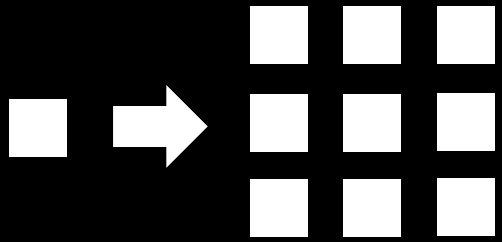
Quantum Reed–Solomon Error Correction

$[[21,12,4]]$
$[[21,9,6]]$

Goal: Implement G_{sym} in quantum circuit.

Goal: Implement G_{sym} in quantum circuit.

4×7



12×21

$$G_{sym} \longrightarrow G_{bin}$$

Every entry in G_{sym} is transformed into a 3×3 matrix.

Goal: Implement G_{sym} in quantum circuit.

12×21

$$G_{bin} \xrightarrow{\text{RREF}} G_s = [I|P]$$

Parity Matrix

If $P[i,j] = 1$, CX(i-th data, j-th parity)

Goal: Implement G_{sym} in quantum circuit.

12×21

$$G_{bin} \xrightarrow{\text{RREF}} G_s = [I|P]$$

Parity Matrix

If $P[i,j] = 1$, CX(i-th data, j-th parity)

Goal: Implement G_{sym} in quantum circuit.

$$\overset{12 \times 21}{G_{bin}} \longrightarrow \begin{array}{l} G_s = [\overset{12 \times 12}{I} | \overset{12 \times 9}{P}] \\ H = [P^T | I] \end{array}$$

Goal: Implement G_{sym} in quantum circuit.

Encoder

$$G_s = [I|P]$$

Decoder

$$H = [P^T|I]$$

Quantum Circuit

Goal: Implement G_{sym} in quantum circuit.

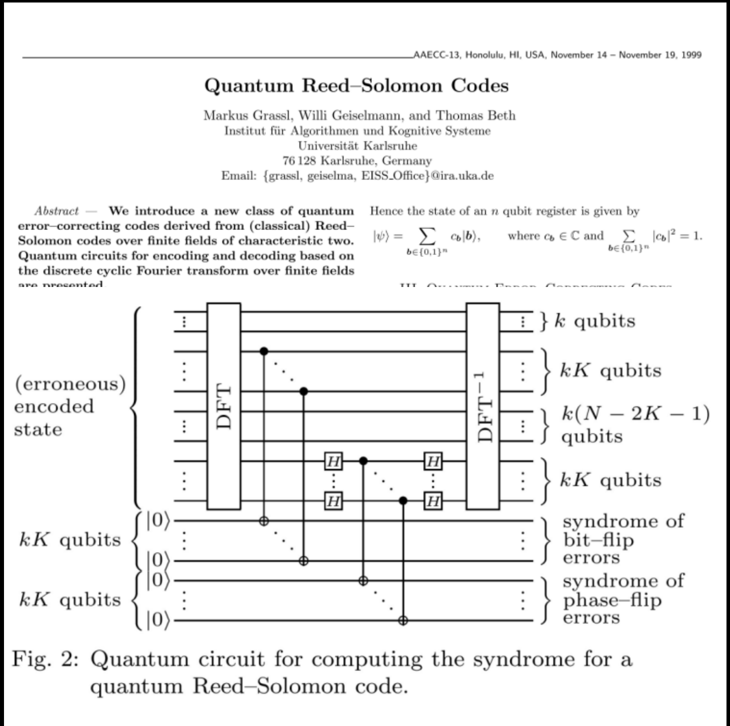
Encoder

$$G_s = [I\,|\,P]$$

Decoder

$$H = [P^T\,|\,I] \Longrightarrow \text{Syndrome}$$

Existing Quantum RS Implementations


AAECC-13, Honolulu, HI, USA, November 14 – November 19, 1999

Quantum Reed–Solomon Codes

Markus Grassl, Willi Geiselmann, and Thomas Beth
Institut für Algorithmen und Kognitive Systeme
Universität Karlsruhe
76 128 Karlsruhe, Germany
Email: {grassl, geiselma, EISS_Office}@ira.uka.de

Abstract — **We introduce a new class of quantum error–correcting codes derived from (classical) Reed–Solomon codes over finite fields of characteristic two. Quantum circuits for encoding and decoding based on the discrete cyclic Fourier transform over finite fields are presented.**

Hence the state of an n qubit register is given by

$$|\psi\rangle = \sum_{\boldsymbol{b}\in\{0,1\}^n} c_{\boldsymbol{b}}|\boldsymbol{b}\rangle, \qquad \text{where } c_{\boldsymbol{b}} \in \mathbb{C} \text{ and } \sum_{\boldsymbol{b}\in\{0,1\}^n} |c_{\boldsymbol{b}}|^2 = 1.$$

Fig. 2: Quantum circuit for computing the syndrome for a quantum Reed–Solomon code.

- For cat qubits, unnecessarily corrects phase flip errors
- DFT requires non-Cliffords and is not implementable with Stim

Repetition vs. Reed-Solomon

Pros: Repetition

Simple, easy to implement and decode, robust against errors

Pros: Reed-Solomon

High code rate, robust against bursts and erasures, optimal code distance

Cons: Repetition

Poor code rate, bad at handling bursts of errors

Cons: Reed-Solomon

Computationally intensive, harder to implement and decode

Tornados: Best of Both Worlds

Complexity separation allows both codes to contribute

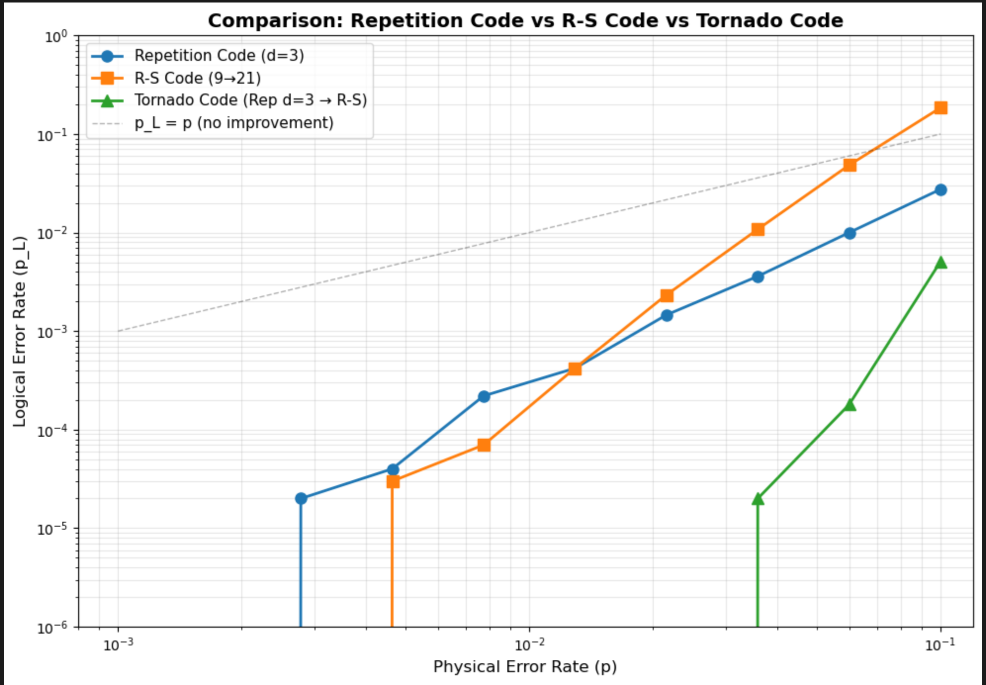
- Repetition (or other LDPC codes) deal with sparse error cases
- R-S code receives dramatically reduced workload
- Result is much faster than R-S and much more reliable than repetition

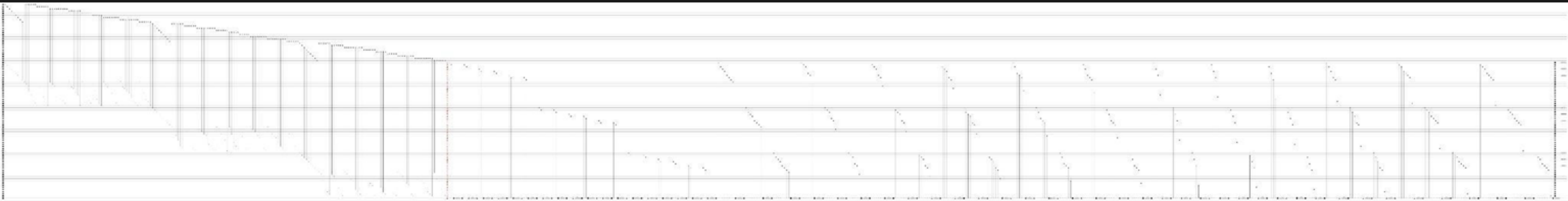
Raw message is encoded using repetition (fast) → Repeated message is encoded using Reed Solomon (reliable) → Decryption occurs in the opposite direction

Quantum Tornado

To the best of our knowledge, we introduce the first instance of a tornado code for quantum error correction

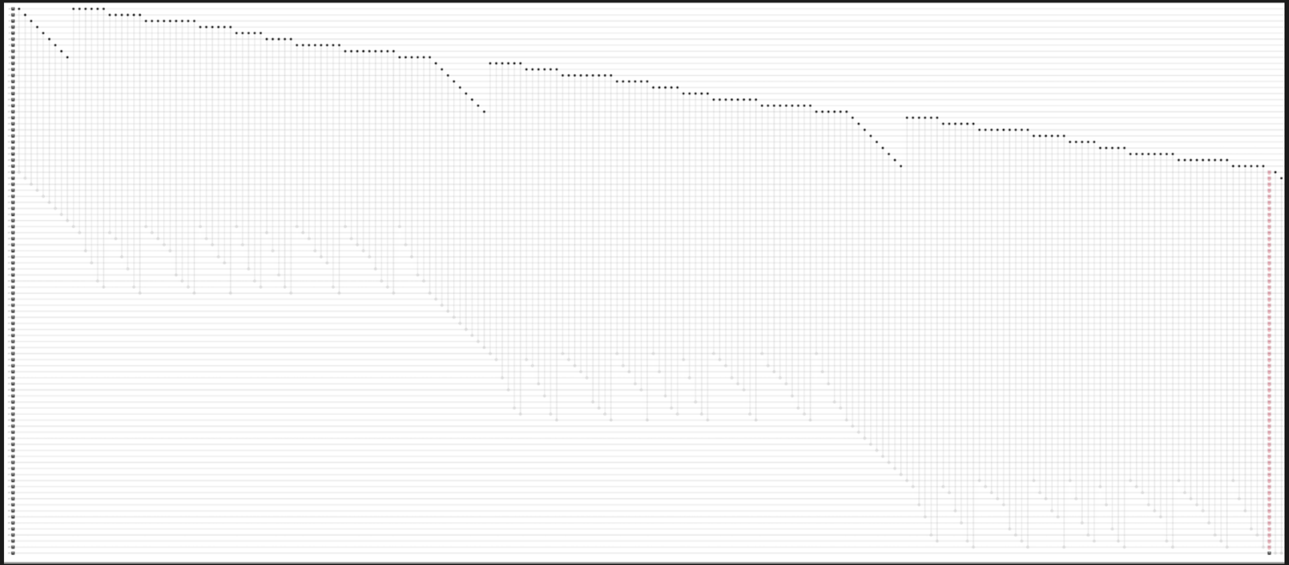
- Layers a repetition code and a quantum Reed-Solomon code
- Shows significant advantage over either algorithm
    - Over 10^5 tests at each error probability, tornado suppresses 100% of physical error for p<0.03
    - Up to p=0.1, tornado is far better than each individual algorithm
- 10^6 tests simulated in 8.5 seconds with Stim





9 logical qubits
27 intermediate qubits
63 final message qubits

Can be changed for any repetition count and R-S code



Repetition module

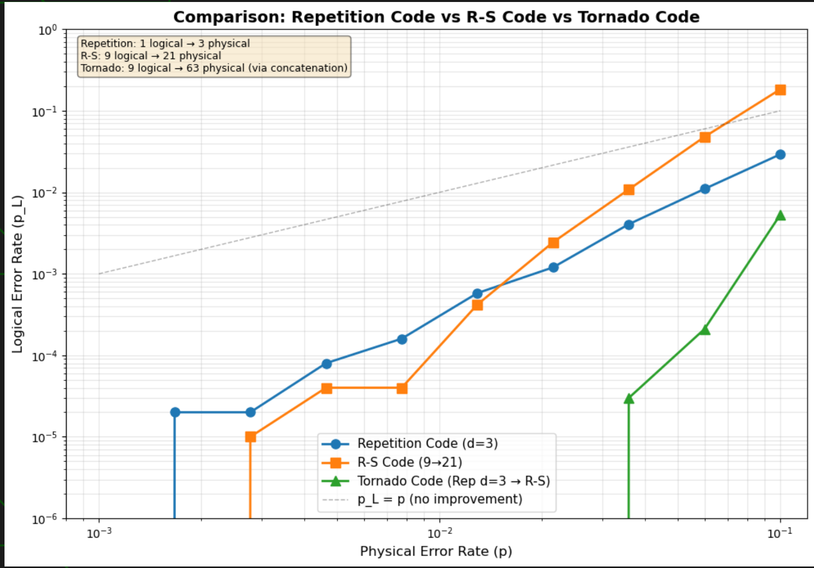
Key Metrics

Connectivity

Code Rate

Code Distance

Results



Comparison: Repetition Code vs R-S Code vs Tornado Code
Repetition: 1 logical → 3 physical
R-S: 9 logical → 21 physical
Tornado: 9 logical → 63 physical (via concatenation)
Repetition Code (d=3)
R-S Code (9→21)
Tornado Code (Rep d=3 → R-S)
p_L = p (no improvement)
Logical Error Rate (p_L)
Physical Error Rate (p)
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-3}
10^{-2}
10^{-1}