QAMP 2025

Qiskit Module on CSS-T qLDPC Codes

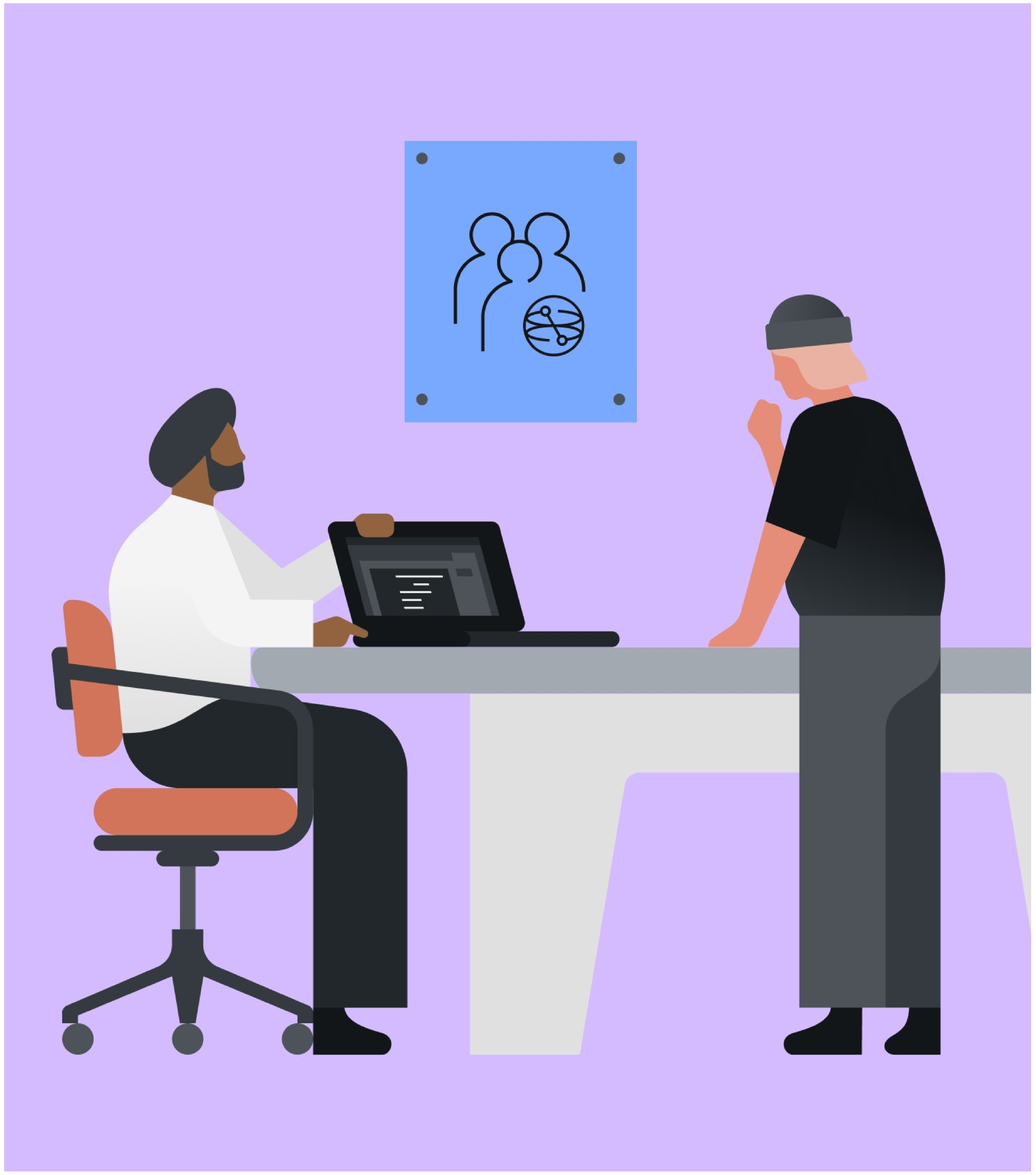
#41

Speaker name: Junseok J., Lia L.

Team members:

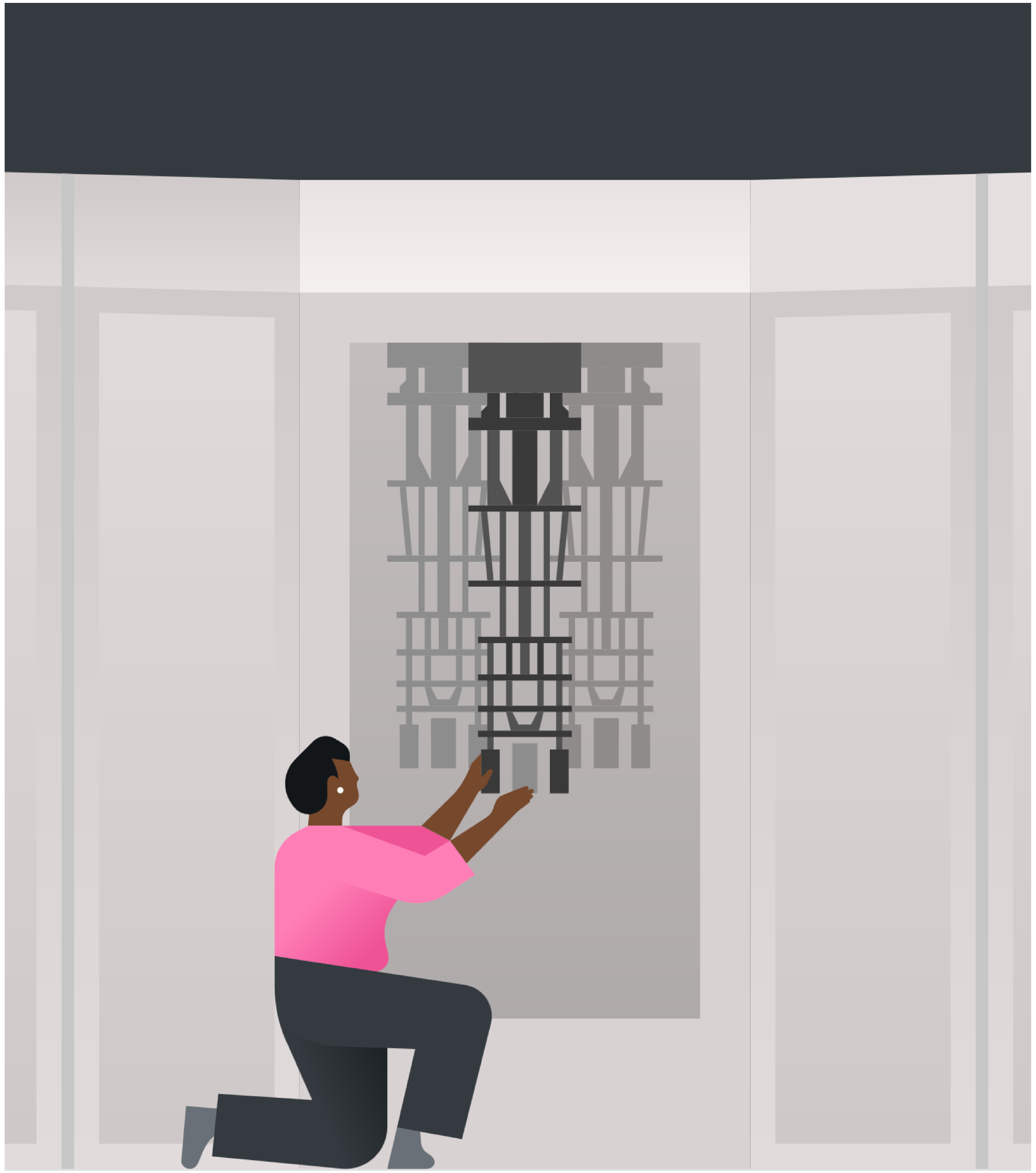
- Adithya (IN)
- Debashis (IN)
- Divyanshu S. (IN)
- Junseok J. (KOR)
- Mark A. (US)
- Lia L. (DE)

Mentor: I-chi C. (US)



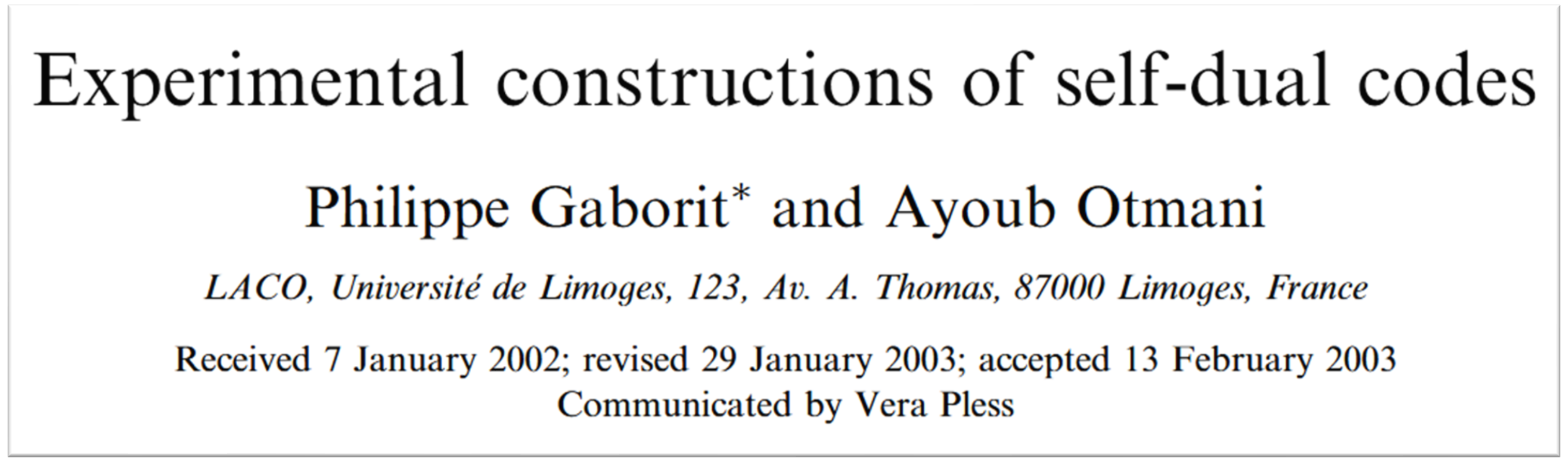
Outline

- Codes for Clifford Gates and for T Gate
- Illustration of Construction
- Decoding Performance



Codes for Transversal Clifford Gate

- Self-dual CSS codes
  - Construction: from classical self-dual codes
  - Support Clifford gates transversally

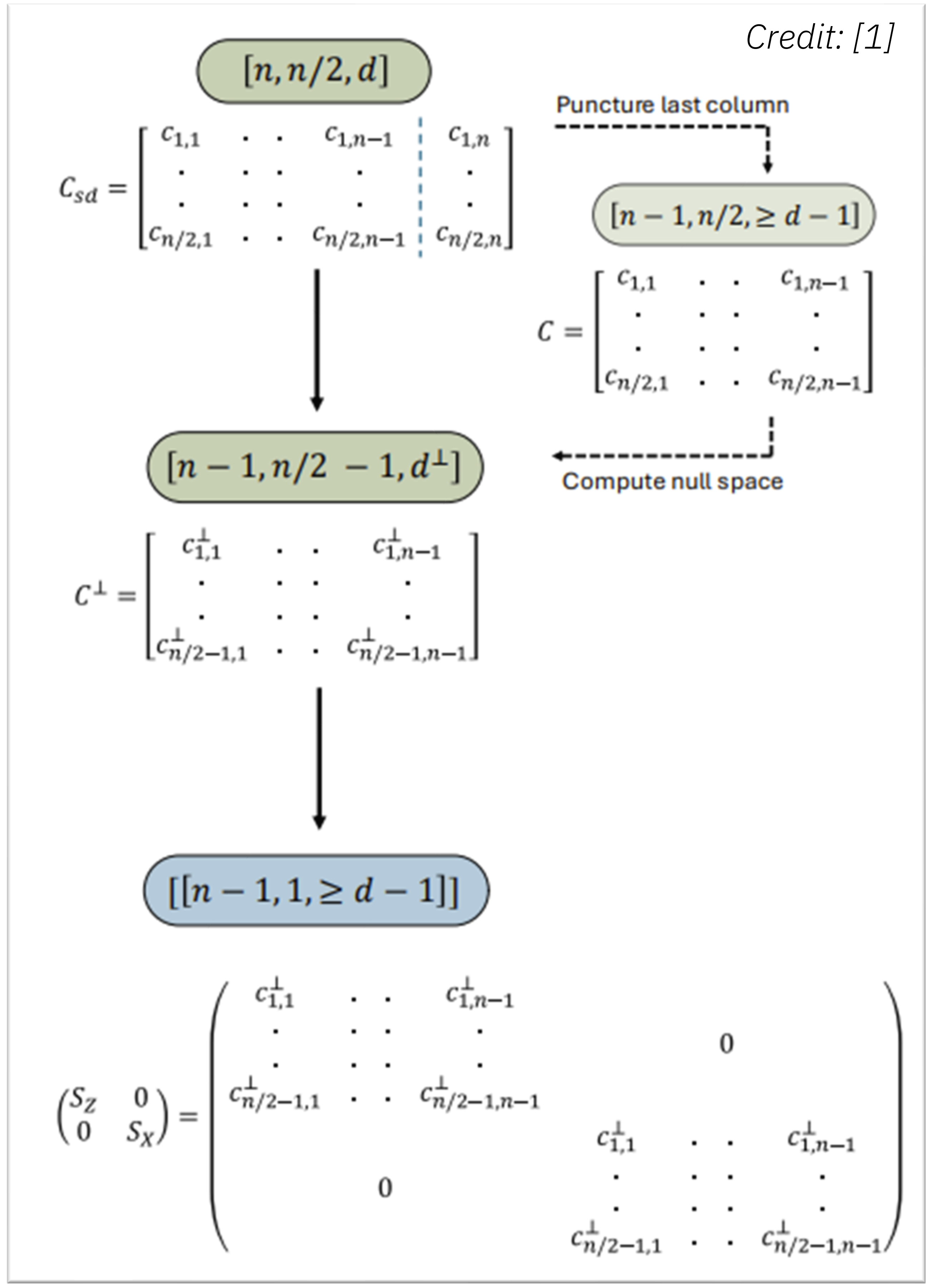


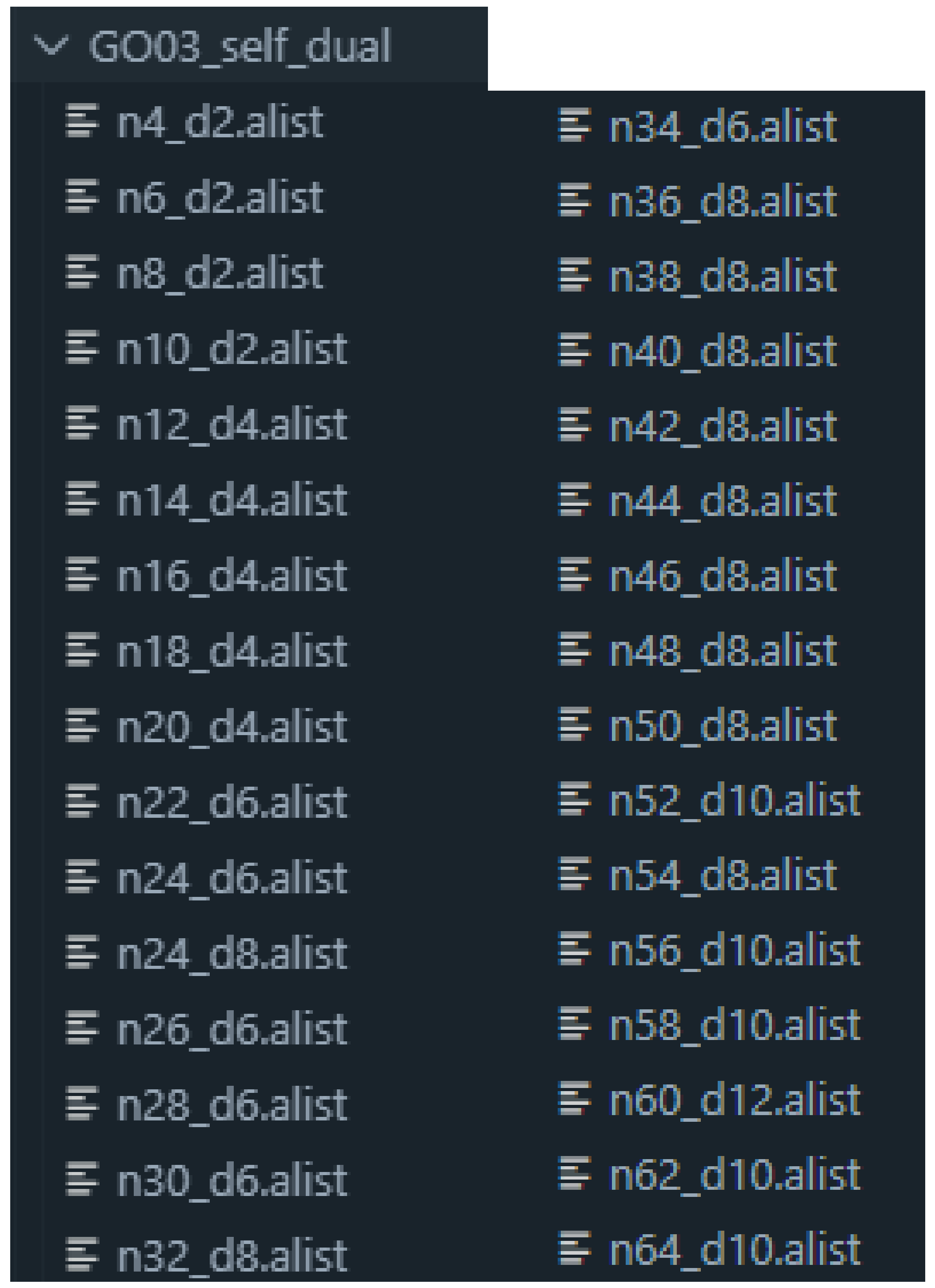
Experimental constructions of self-dual codes

Philippe Gaborit* and Ayoub Otmani

LACO, Université de Limoges, 123, Av. A. Thomas, 87000 Limoges, France

Received 7 January 2002; revised 29 January 2003; accepted 13 February 2003
Communicated by Vera Pless


Credit: [1]
$[n, n/2, d]$
Puncture last column
$[n-1, n/2, \geq d-1]$
$C_{sd} = \begin{bmatrix} c_{1,1} & \cdots & c_{1,n-1} & c_{1,n} \\ \vdots & & \vdots & \vdots \\ c_{n/2,1} & \cdots & c_{n/2,n-1} & c_{n/2,n} \end{bmatrix}$
$C = \begin{bmatrix} c_{1,1} & \cdots & c_{1,n-1} \\ \vdots & & \vdots \\ c_{n/2,1} & \cdots & c_{n/2,n-1} \end{bmatrix}$
Compute null space
$[n-1, n/2-1, d^{\perp}]$
$C^{\perp} = \begin{bmatrix} c^{\perp}_{1,1} & \cdots & c^{\perp}_{1,n-1} \\ \vdots & & \vdots \\ c^{\perp}_{n/2-1,1} & \cdots & c^{\perp}_{n/2-1,n-1} \end{bmatrix}$
$[[n-1, 1, \geq d-1]]$
$\begin{pmatrix} S_Z & 0 \\ 0 & S_X \end{pmatrix} = \begin{pmatrix} C^{\perp} & 0 \\ 0 & C^{\perp} \end{pmatrix}$

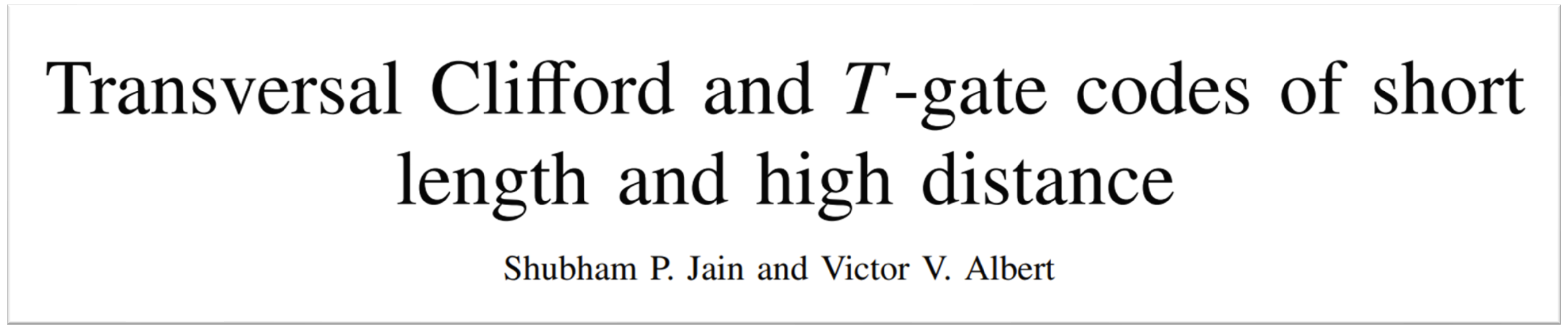


GO03_self_dual

n4_d2.alist	n34_d6.alist
n6_d2.alist	n36_d8.alist
n8_d2.alist	n38_d8.alist
n10_d2.alist	n40_d8.alist
n12_d4.alist	n42_d8.alist
n14_d4.alist	n44_d8.alist
n16_d4.alist	n46_d8.alist
n18_d4.alist	n48_d8.alist
n20_d4.alist	n50_d8.alist
n22_d6.alist	n52_d10.alist
n24_d6.alist	n54_d8.alist
n24_d8.alist	n56_d10.alist
n26_d6.alist	n58_d10.alist
n28_d6.alist	n60_d12.alist
n30_d6.alist	n62_d10.alist
n32_d8.alist	n64_d10.alist

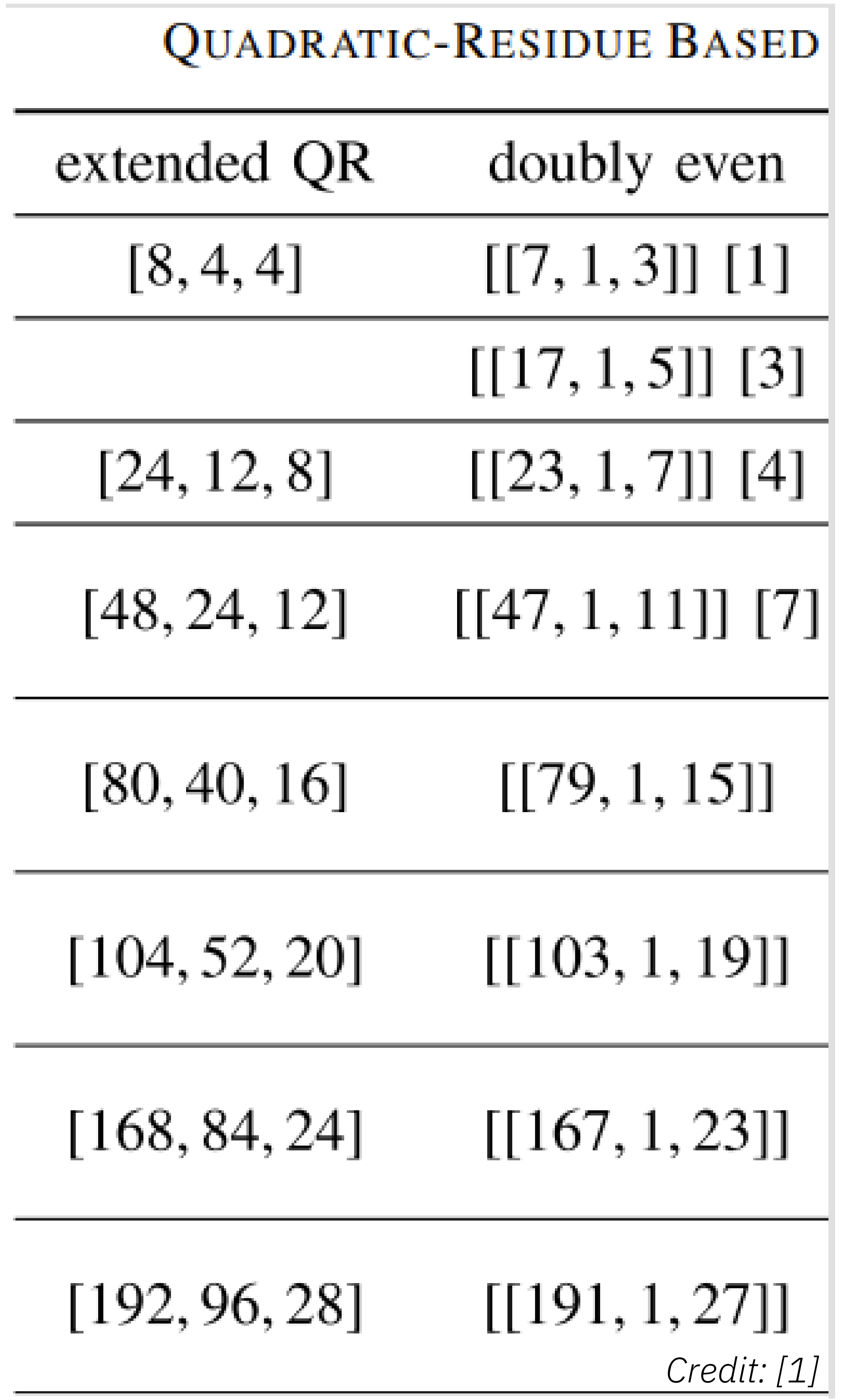
[1] S. P. Jain and V. V. Albert, "Transversal Clifford and T-gate codes of short length and high distance," IEEE Journal on Selected Areas in Information Theory, 2025.

Codes for Transversal Clifford Gate


Transversal Clifford and T-gate codes of short length and high distance

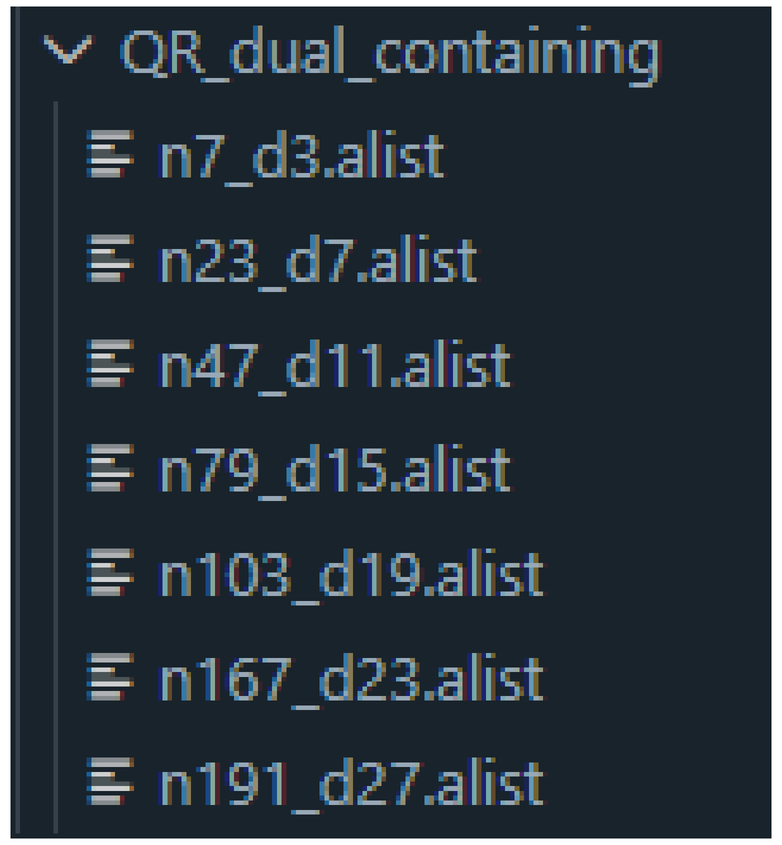
Shubham P. Jain and Victor V. Albert

- Self-dual CSS codes
  - Construction: from classical self-dual codes
  - Support Clifford gates transversally


QUADRATIC-RESIDUE BASED

extended QR	doubly even
$[8,4,4]$	$[[7,1,3]]$ [1]
	$[[17,1,5]]$ [3]
$[24,12,8]$	$[[23,1,7]]$ [4]
$[48,24,12]$	$[[47,1,11]]$ [7]
$[80,40,16]$	$[[79,1,15]]$
$[104,52,20]$	$[[103,1,19]]$
$[168,84,24]$	$[[167,1,23]]$
$[192,96,28]$	$[[191,1,27]]$

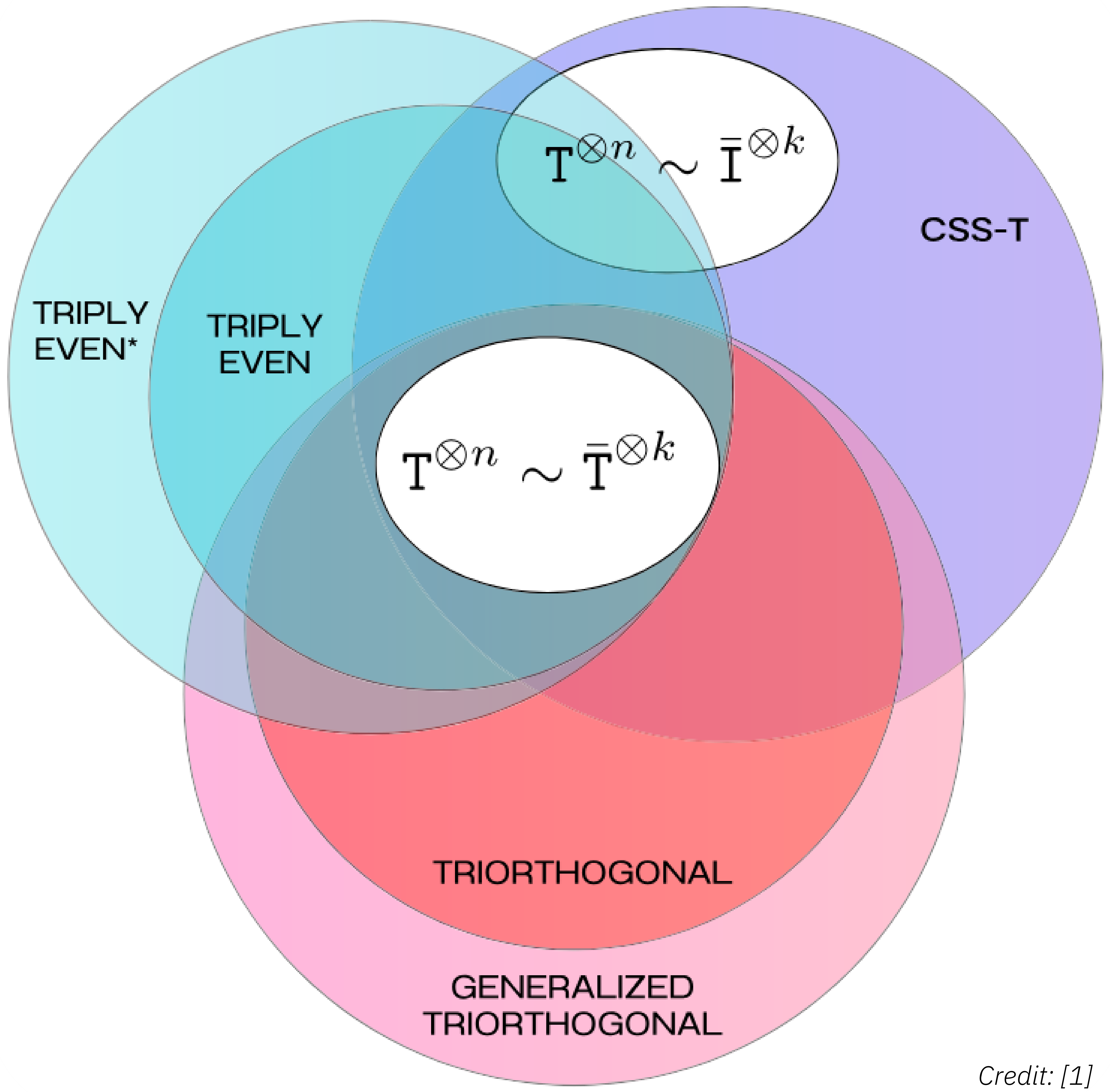
Credit: [1]


```
QR_dual_containing
  n7_d3.alist
  n23_d7.alist
  n47_d11.alist
  n79_d15.alist
  n103_d19.alist
  n167_d23.alist
  n191_d27.alist
```

[1] S. P. Jain and V. V. Albert, "Transversal clifford and t-gate codes of short length and high distance," IEEE Journal on Selected Areas in InformationTheory, 2025.

Codes for T Gate

- Self-dual CSS codes
- **Universal gate set**: Clifford + T = {H, S, CNOT, T}
- **Triorthogonal / Triply Even Codes**
  - Support T gate transversally



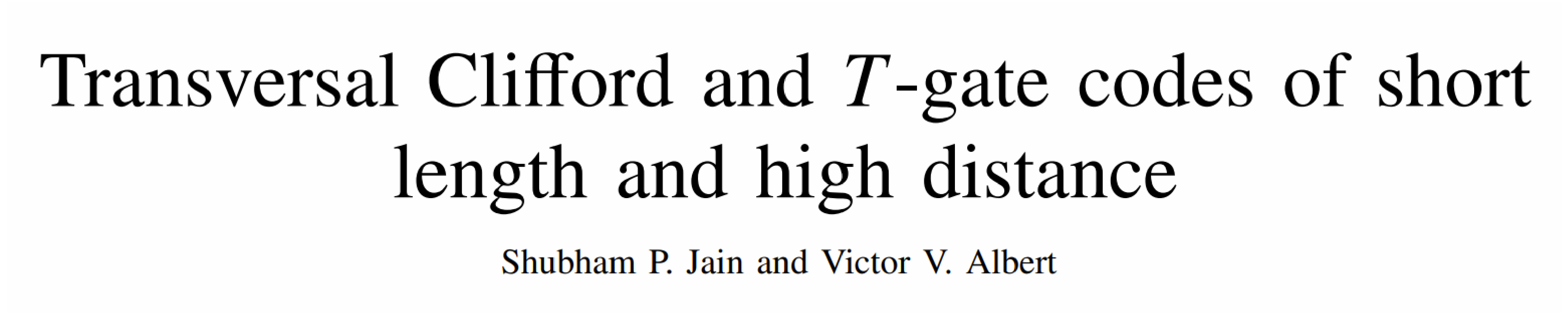
$T^{\otimes n} \sim \bar{I}^{\otimes k}$
CSS-T
TRIPLY EVEN*
TRIPLY EVEN
$T^{\otimes n} \sim \bar{T}^{\otimes k}$
TRIORTHOGONAL
GENERALIZED TRIORTHOGONAL

Credit: [1]

[1] S. P. Jain and V. V. Albert, "Transversal clifford and t-gate codes of short length and high distance," IEEE Journal on Selected Areas in InformationTheory, 2025.

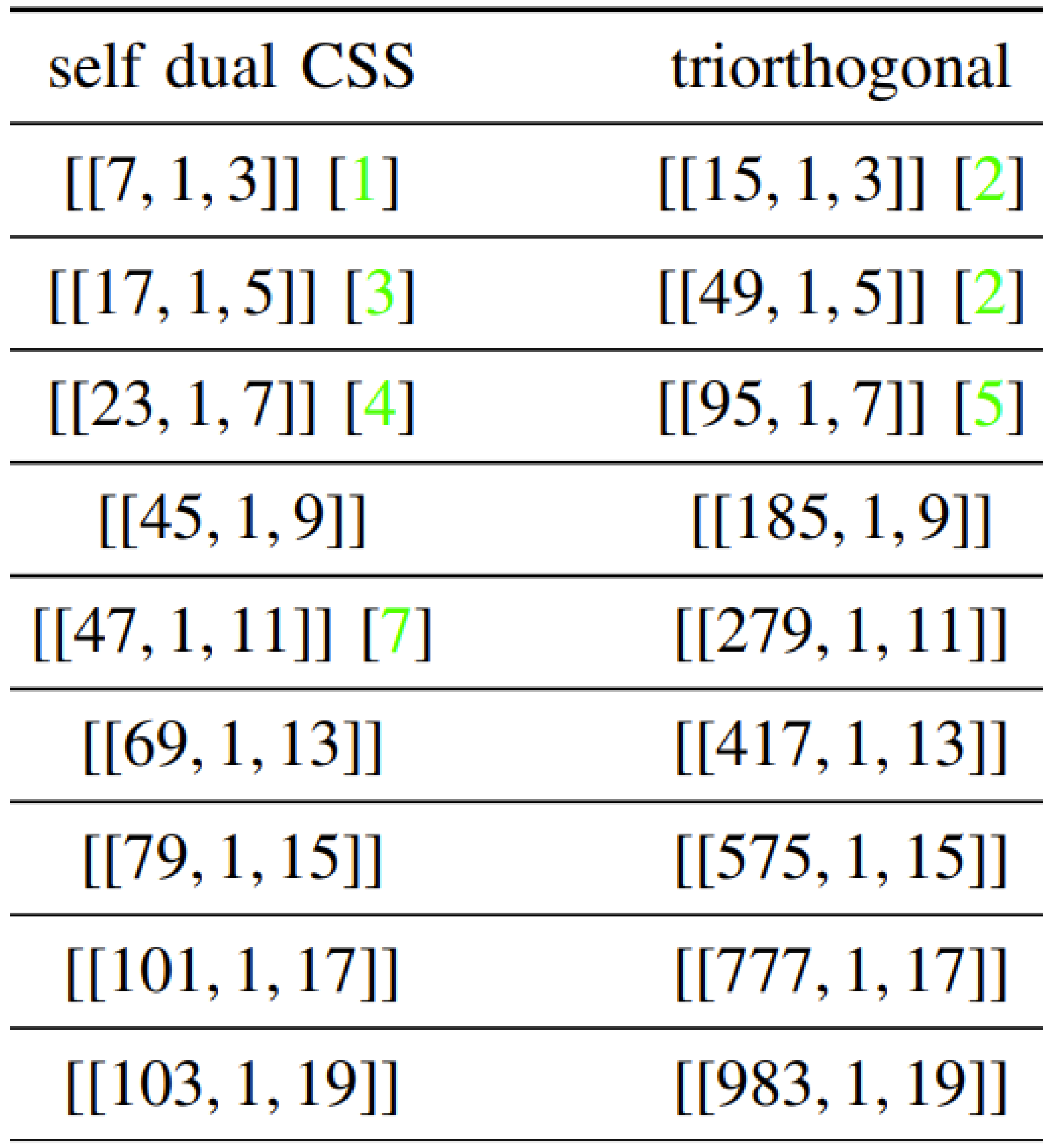
Codes for T Gate

- Self-dual CSS codes
- Universal gate set: Clifford + T = {H, S, CNOT, T}
- Triorthogonal / Triply Even Codes
  - Support T gate transversally
  - **Doubling construction**

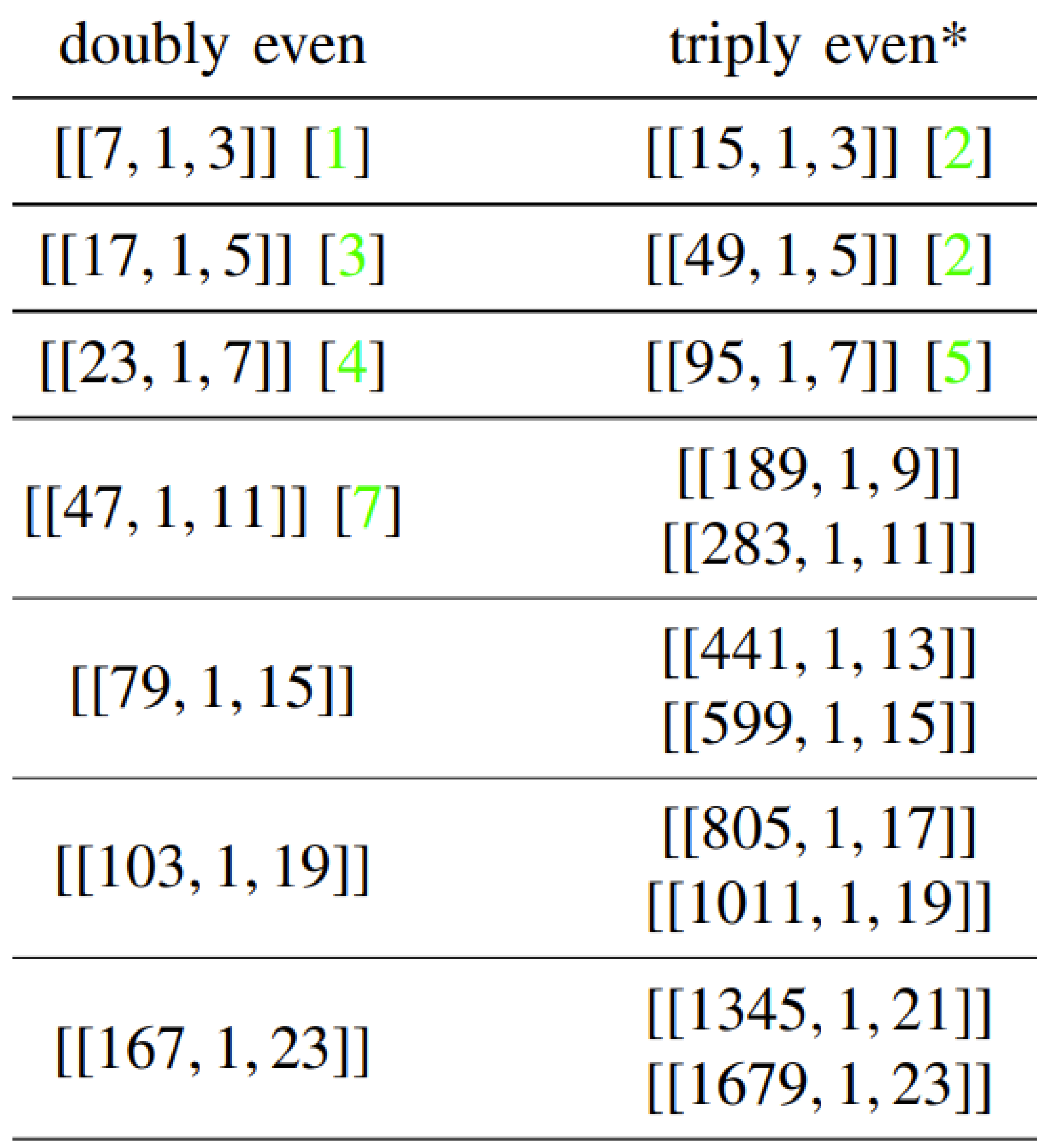


Transversal Clifford and T-gate codes of short length and high distance

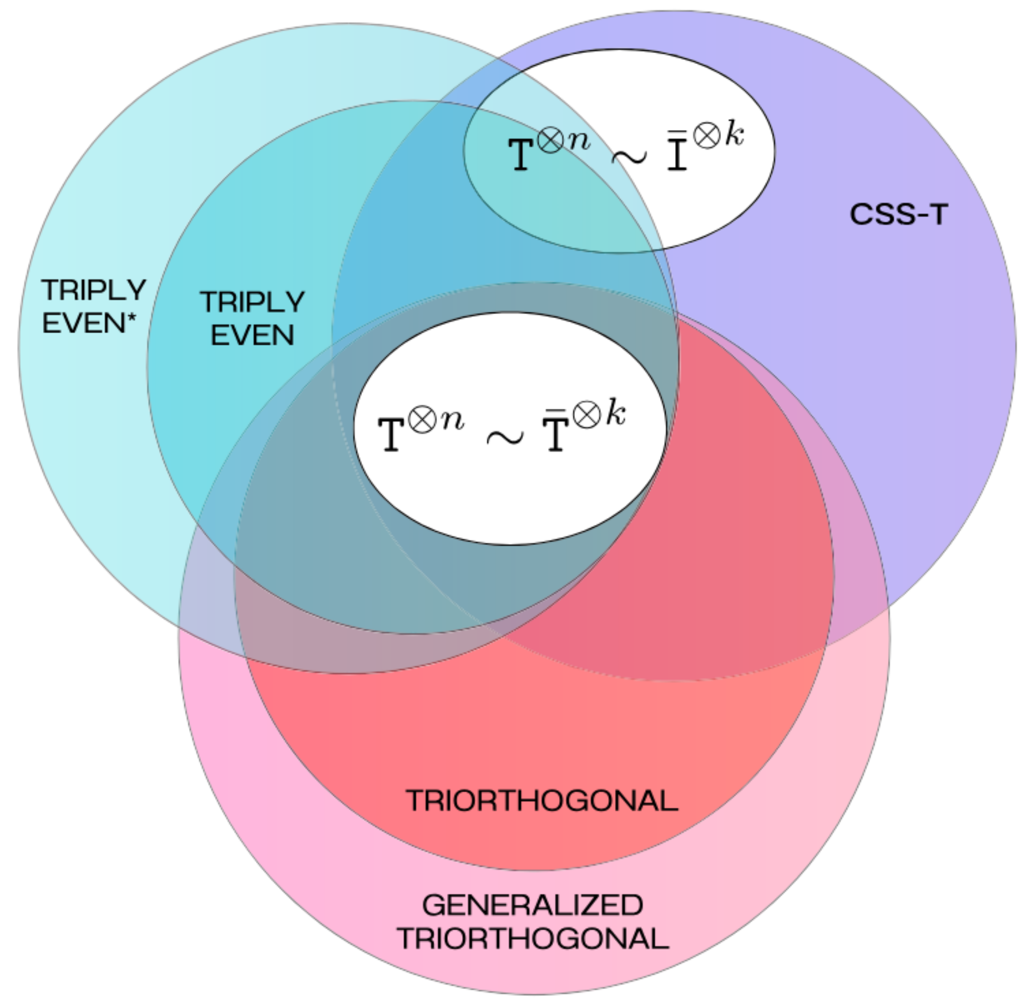
Shubham P. Jain and Victor V. Albert

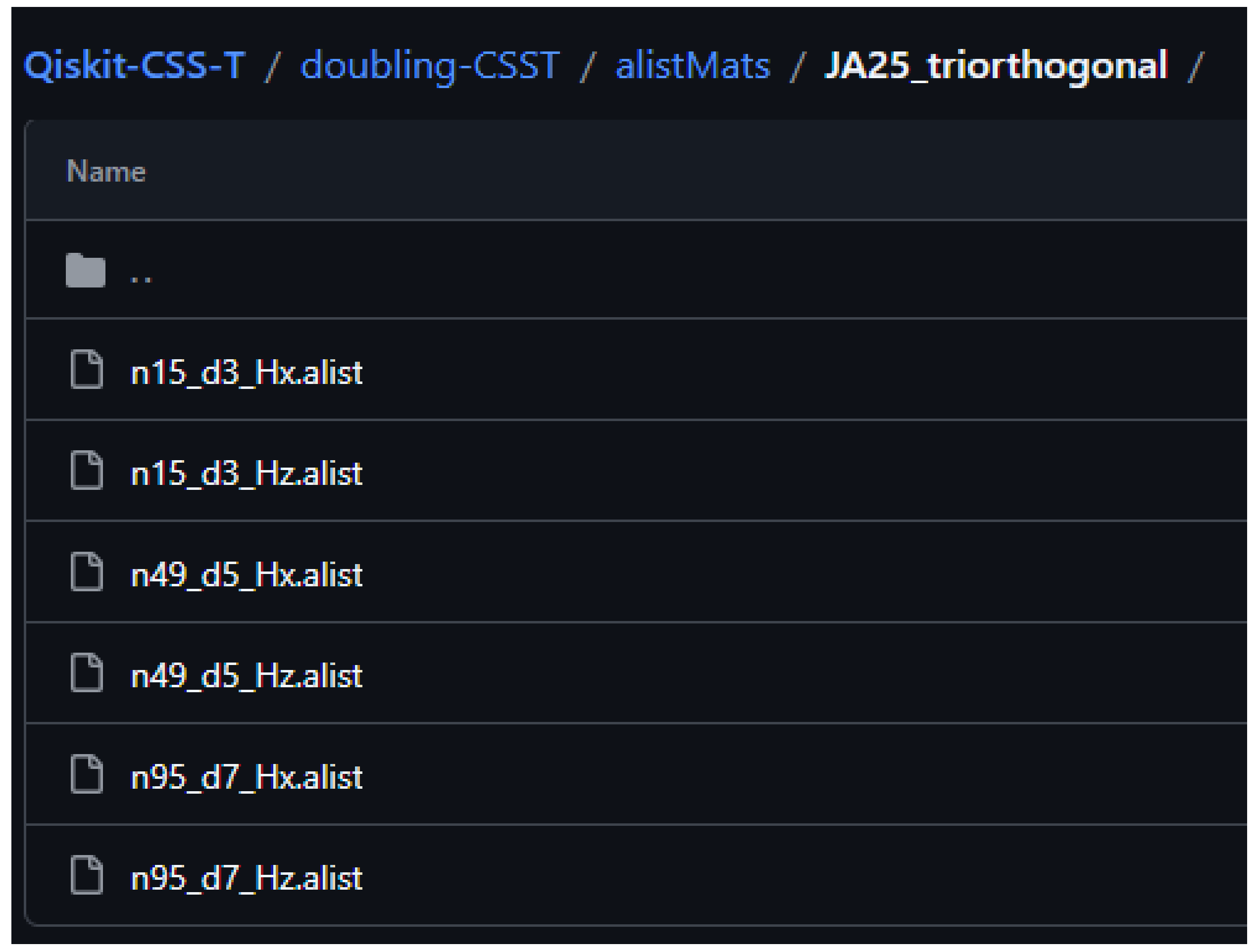


self dual CSS	triorthogonal
[[7, 1, 3]] [1]	[[15, 1, 3]] [2]
[[17, 1, 5]] [3]	[[49, 1, 5]] [2]
[[23, 1, 7]] [4]	[[95, 1, 7]] [5]
[[45, 1, 9]]	[[185, 1, 9]]
[[47, 1, 11]] [7]	[[279, 1, 11]]
[[69, 1, 13]]	[[417, 1, 13]]
[[79, 1, 15]]	[[575, 1, 15]]
[[101, 1, 17]]	[[777, 1, 17]]
[[103, 1, 19]]	[[983, 1, 19]]



doubly even	triply even*
[[7, 1, 3]] [1]	[[15, 1, 3]] [2]
[[17, 1, 5]] [3]	[[49, 1, 5]] [2]
[[23, 1, 7]] [4]	[[95, 1, 7]] [5]
[[47, 1, 11]] [7]	[[189, 1, 9]] [[283, 1, 11]]
[[79, 1, 15]]	[[441, 1, 13]] [[599, 1, 15]]
[[103, 1, 19]]	[[805, 1, 17]] [[1011, 1, 19]]
[[167, 1, 23]]	[[1345, 1, 21]] [[1679, 1, 23]]


$T^{\otimes n} \sim \bar{I}^{\otimes k}$
CSS-T
TRIPLY EVEN*
TRIPLY EVEN
$T^{\otimes n} \sim \bar{T}^{\otimes k}$
TRIORTHOGONAL
GENERALIZED TRIORTHOGONAL



Qiskit-CSS-T / doubling-CSST / alistMats / JA25_triorthogonal /

```
Name
..
n15_d3_Hx.alist
n15_d3_Hz.alist
n49_d5_Hx.alist
n49_d5_Hz.alist
n95_d7_Hx.alist
n95_d7_Hz.alist
```

Comparison of various decoder for self-dual CSS/CSS-T codes

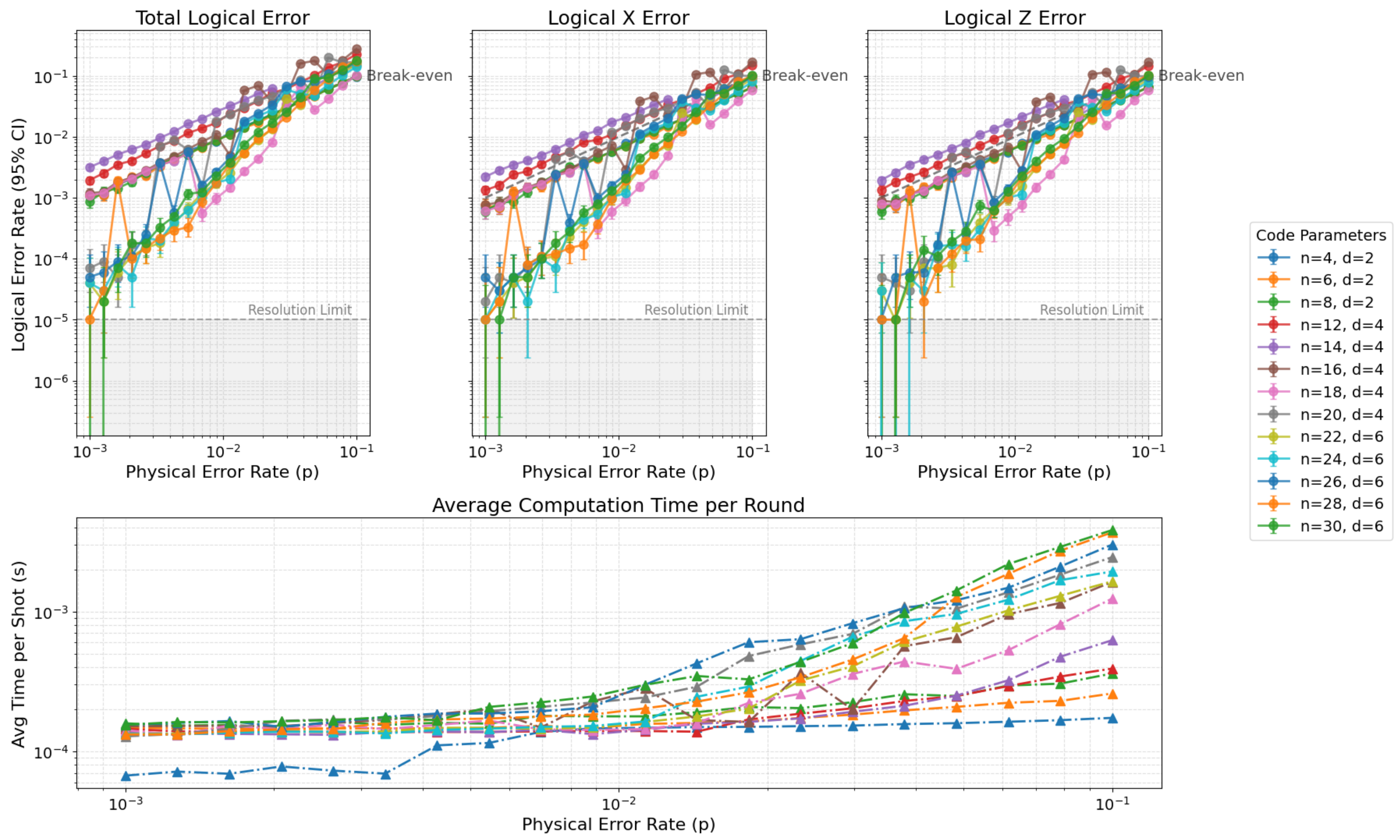
Objective

Finding a suitable decoding algorithm for target self-dual CSS or CSS-T codes codes.

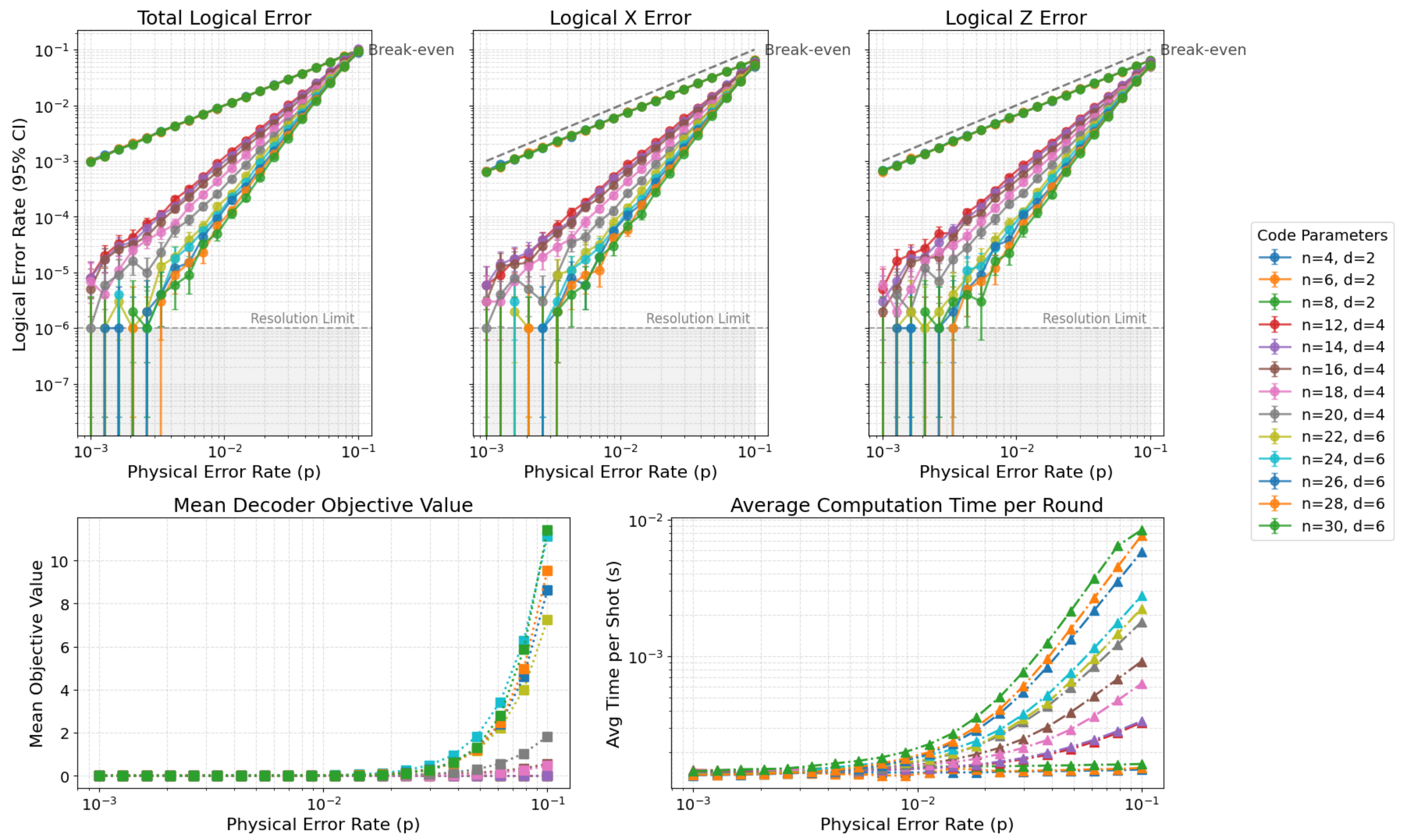
Observation and Discussion

- Conducted Monte Carlo simulations for self-dual CSS codes by directly decoding sampled Pauli error strings.
- Results from Hyperion showed one-level higher accuracy than BP+LSD decoders with comparable computation time.
- Since Hyperion works for finding a parity factor for a general Hypergraph (Tanner graph), it was more suitable for our self-dual CSS or CSS-T codes that doesn't start from geometric structures.
- Hyperion also provides rigorous optimality, and proximity bounds through the objective value (primal-dual gap).

BP+LSD Logical Error Rates (Bias: 0.5)



Hyperion Logical Error Rates (Bias: 0.5)



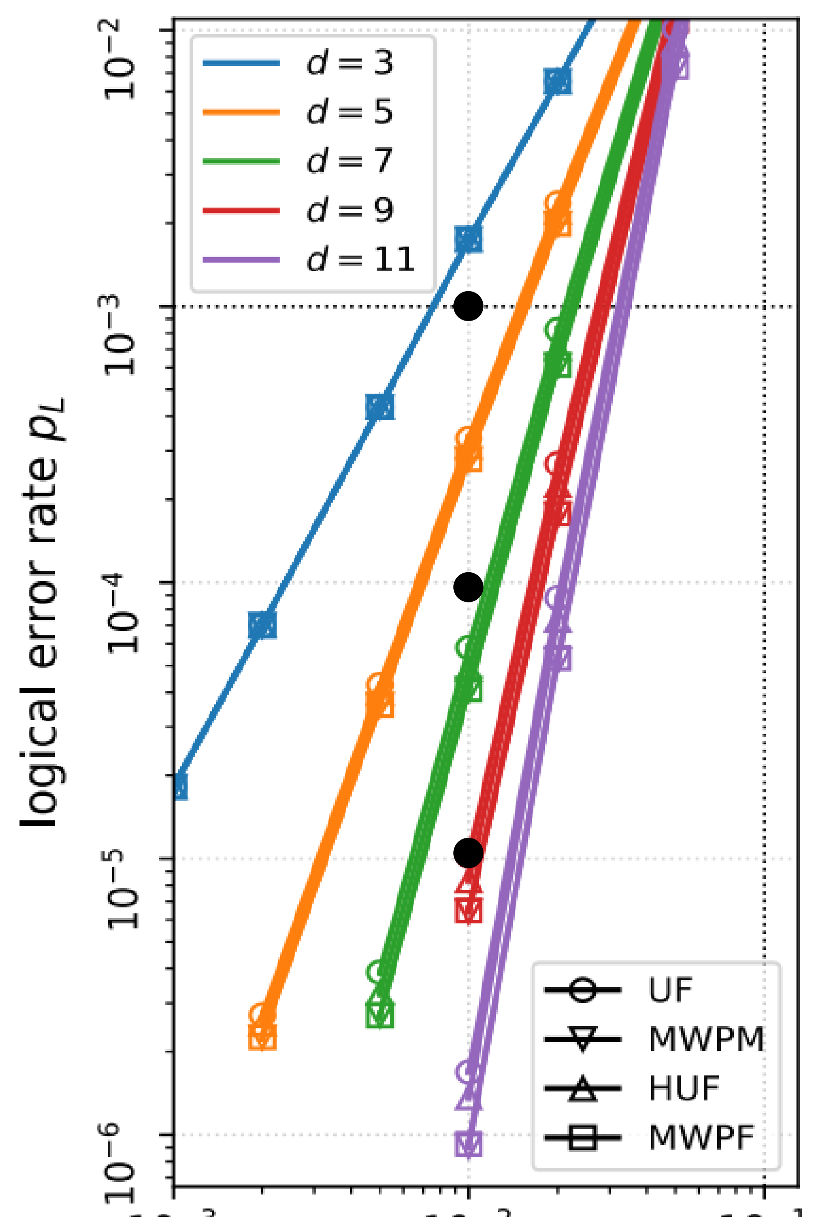
Recreating the results from "Minimum-Weight Parity Factor Decoder for Quantum Error Correction"

Objective

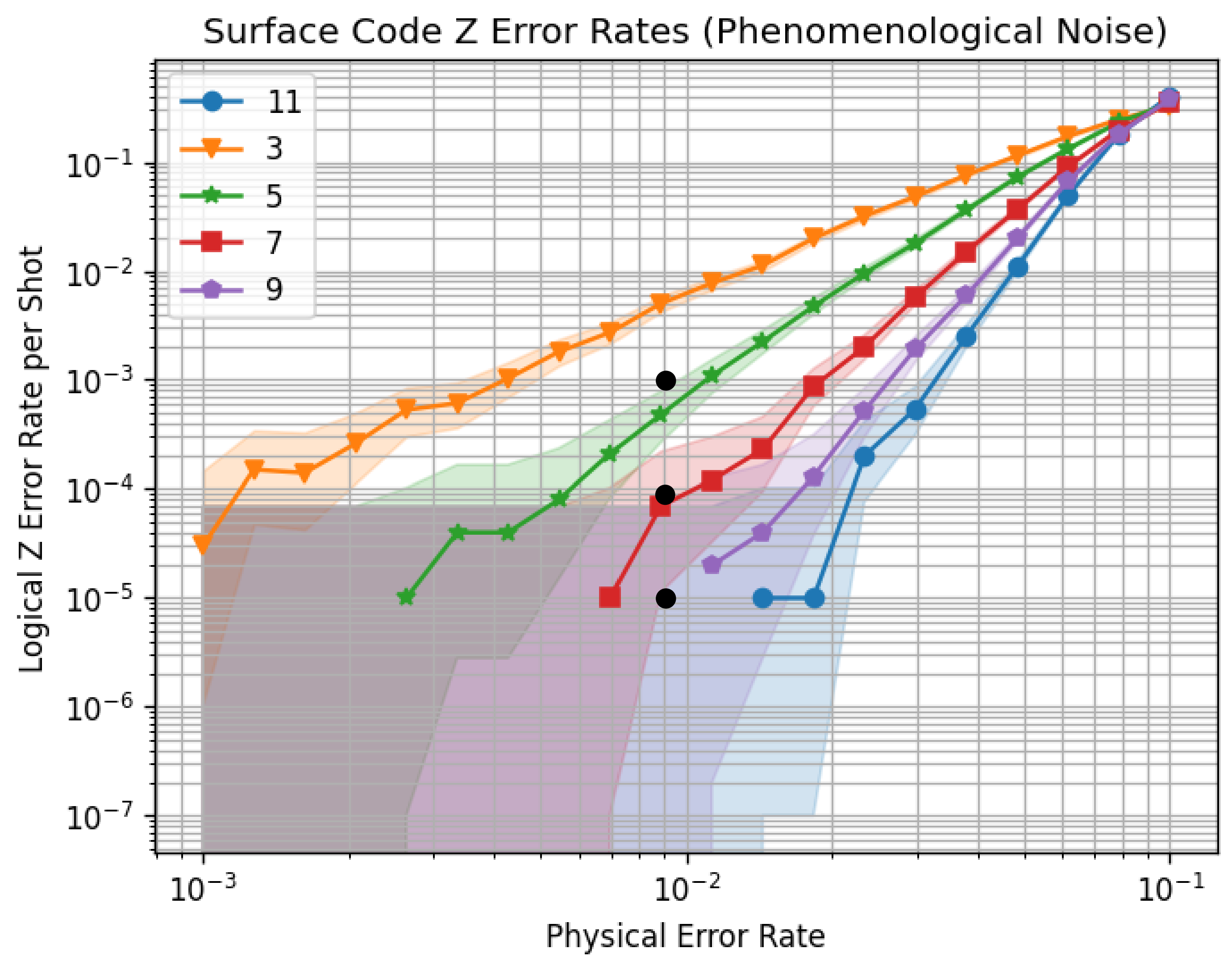
Verifying the circuit-level noise simulation by comparing the results in the reference.

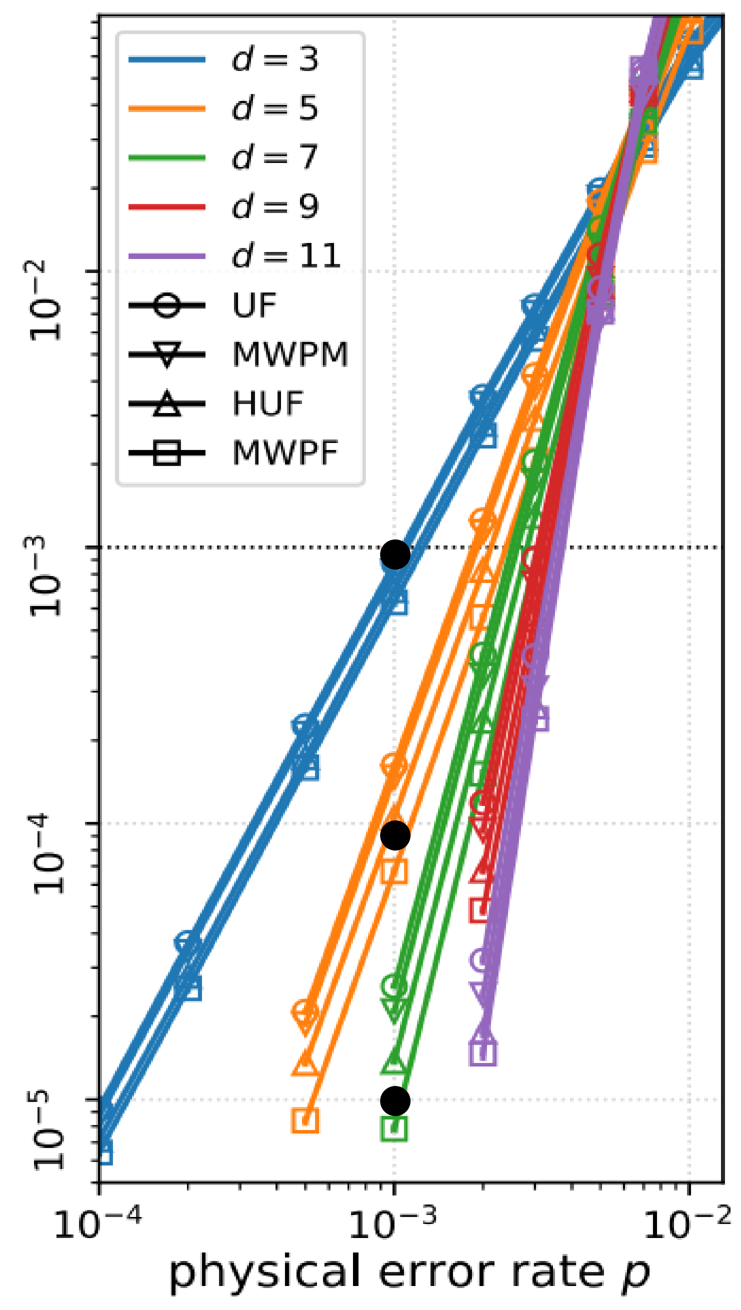
Progress

- Conducted circuit-level noise simulation for Rotated Surface codes and Triangular Color codes provided in Stim package.
- Compared to the results in the reference, Our results were almost identical for the Surface codes and little-bit better for Color codes of same size.

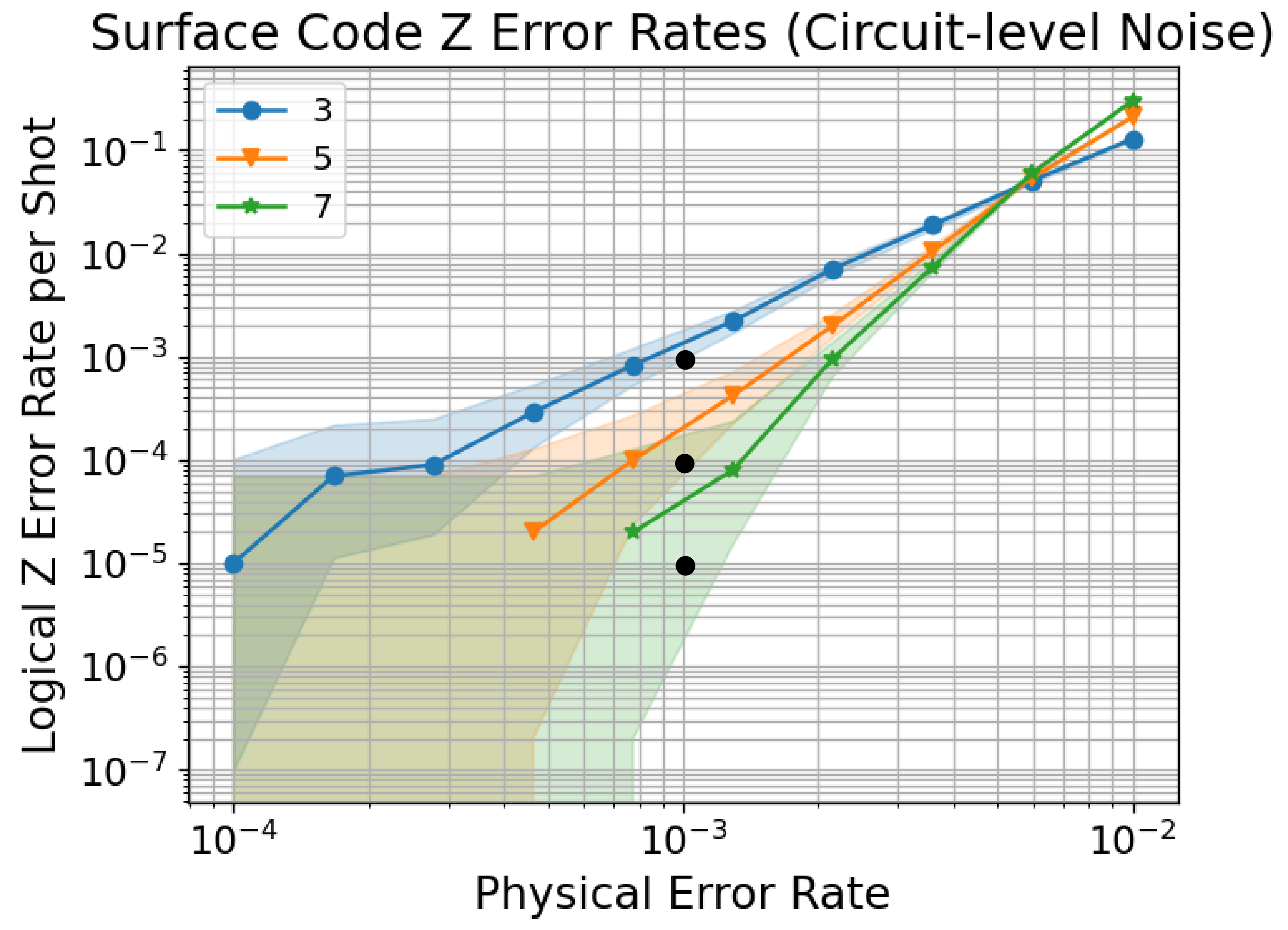


(a) bit-flip noise





(d) circuit-level noise



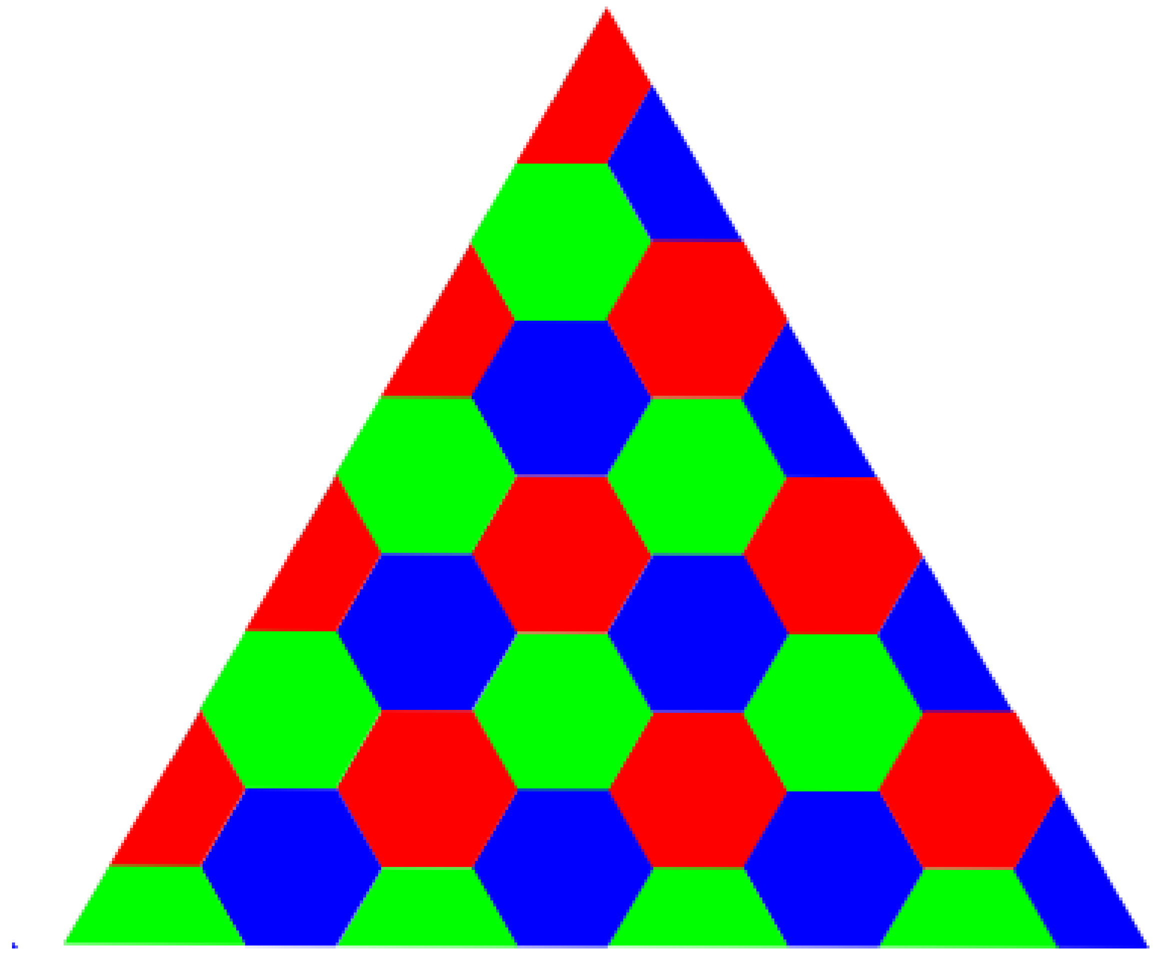
Recreating the results from “Minimum-Weight Parity Factor Decoder for Quantum Error Correction”

Objective

Verifying the circuit-level noise simulation by comparing the results in the reference.

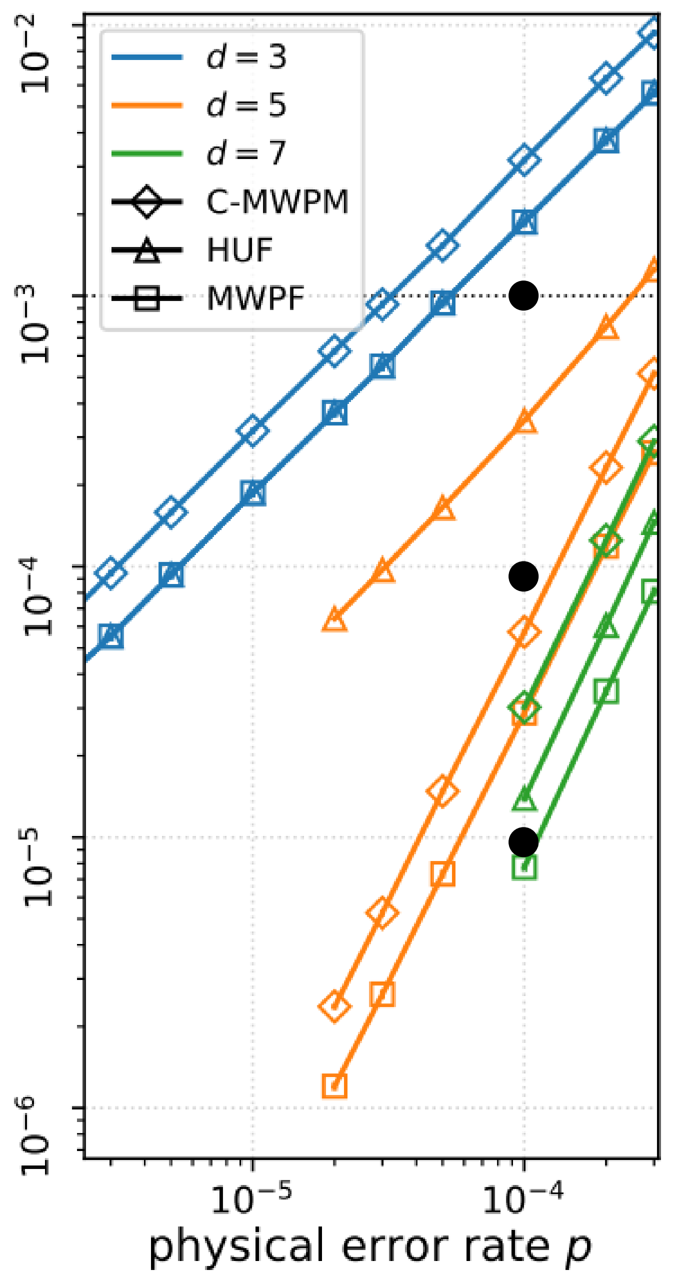
Progress

- Conducted circuit-level noise simulation for Rotated Surface codes and Triangular Color codes provided in Stim package.
- Compared to the results in the reference, Our results were almost identical for the Surface codes and little-bit better for Color codes of same size.

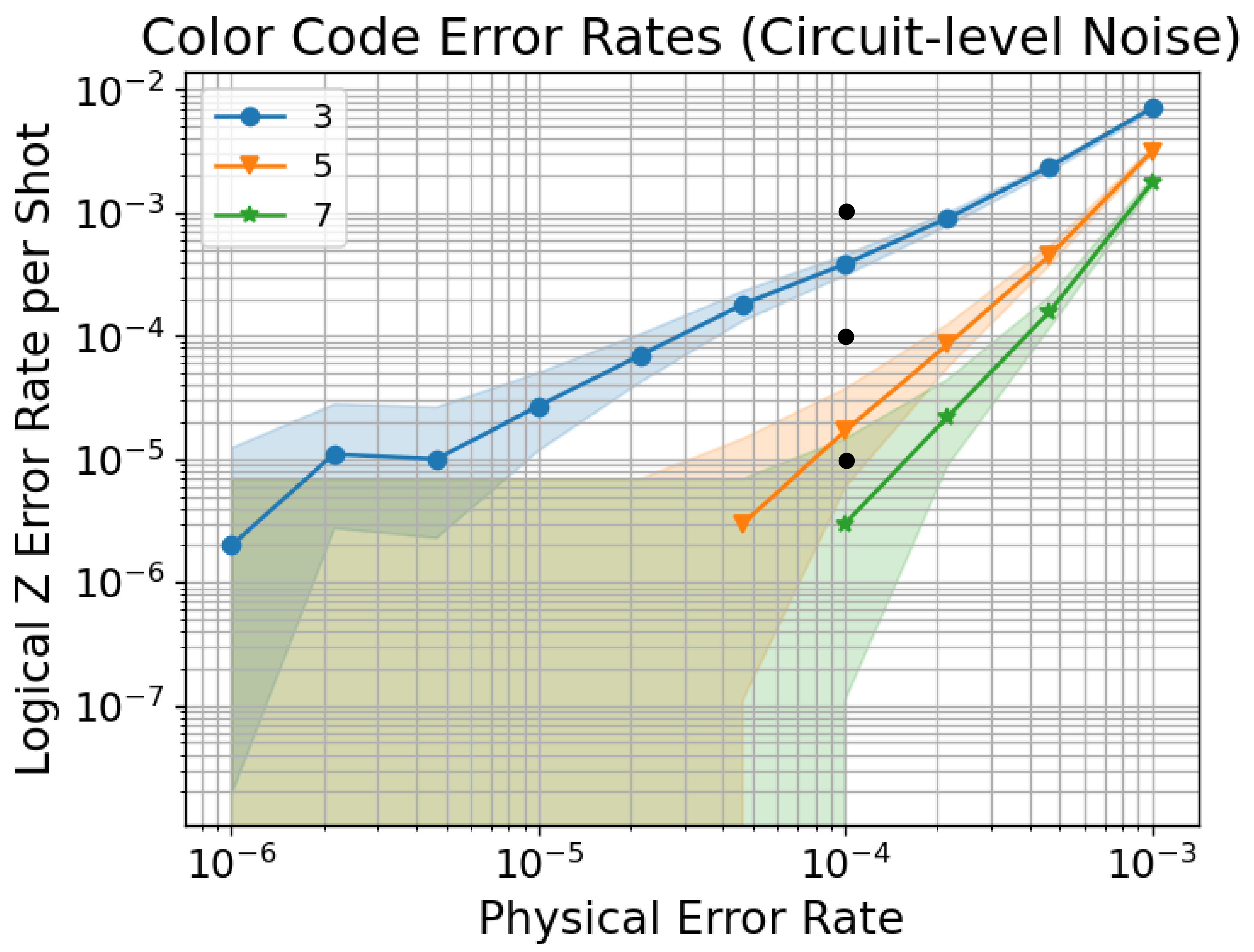


6.6.6 code

$[[(3d^2+1)/4, 1, d]]$



(b) circuit-level noise



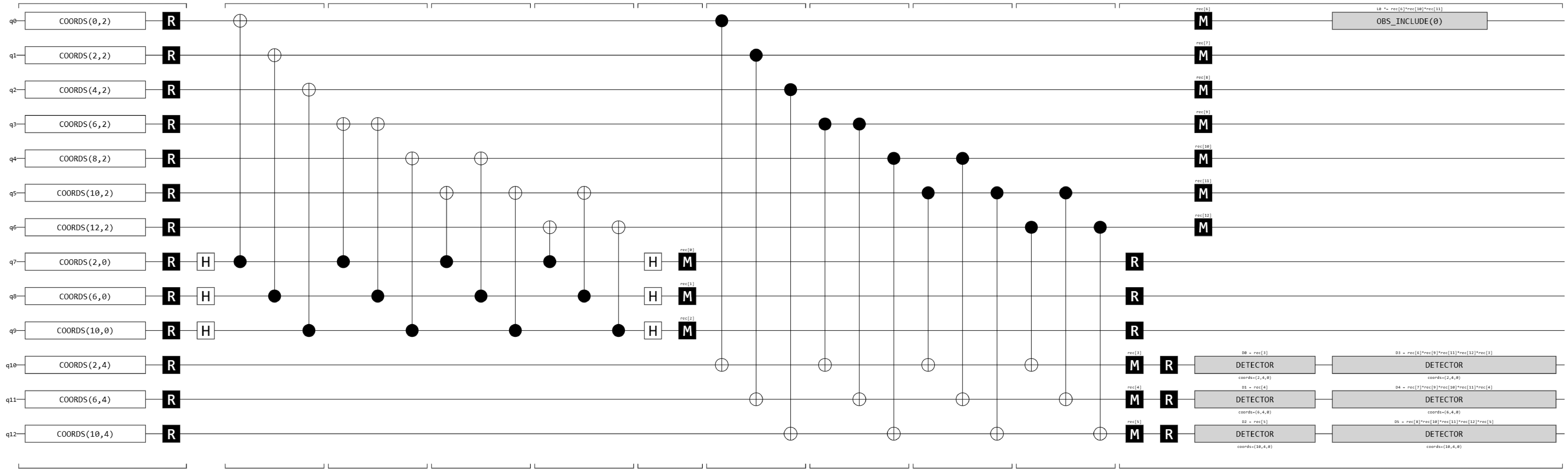
Circuit-level noise simulation for self-dual CSS codes via Stim

Objective

Studying the error capacity of our self-dual CSS/CSS-T codes.

Progress

- Designed measurement circuit for syndrome extraction.



Circuit-level noise simulation for self-dual CSS codes via Stim

Objective

Studying the error capacity of our self-dual CSS/CSS-T codes.

Progress

- Added noise for various noise models(Standard circuit-level Noise model, SI1000 Noise model).

COORDS(0,2)
COORDS(2,2)
COORDS(4,2)
COORDS(6,2)
COORDS(8,2)
COORDS(10,2)
COORDS(12,2)
COORDS(2,0)
COORDS(6,0)
COORDS(10,0)
COORDS(2,4)
COORDS(6,4)
COORDS(10,4)
R
H
M
DEP2
0.001

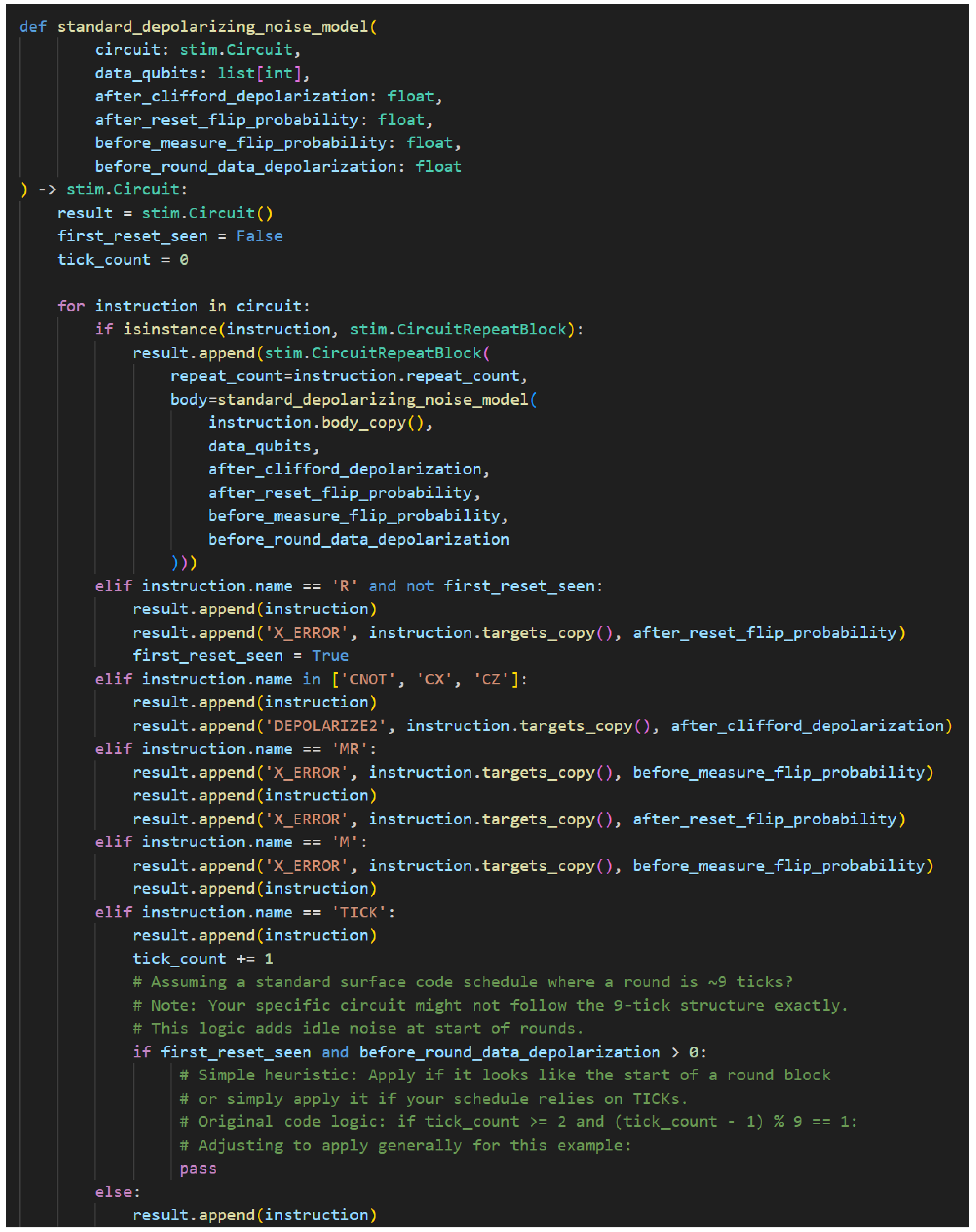
Circuit-level noise simulation for self-dual CSS codes via Stim

Objective

Studying the error capacity of our self-dual CSS/CSS-T codes.

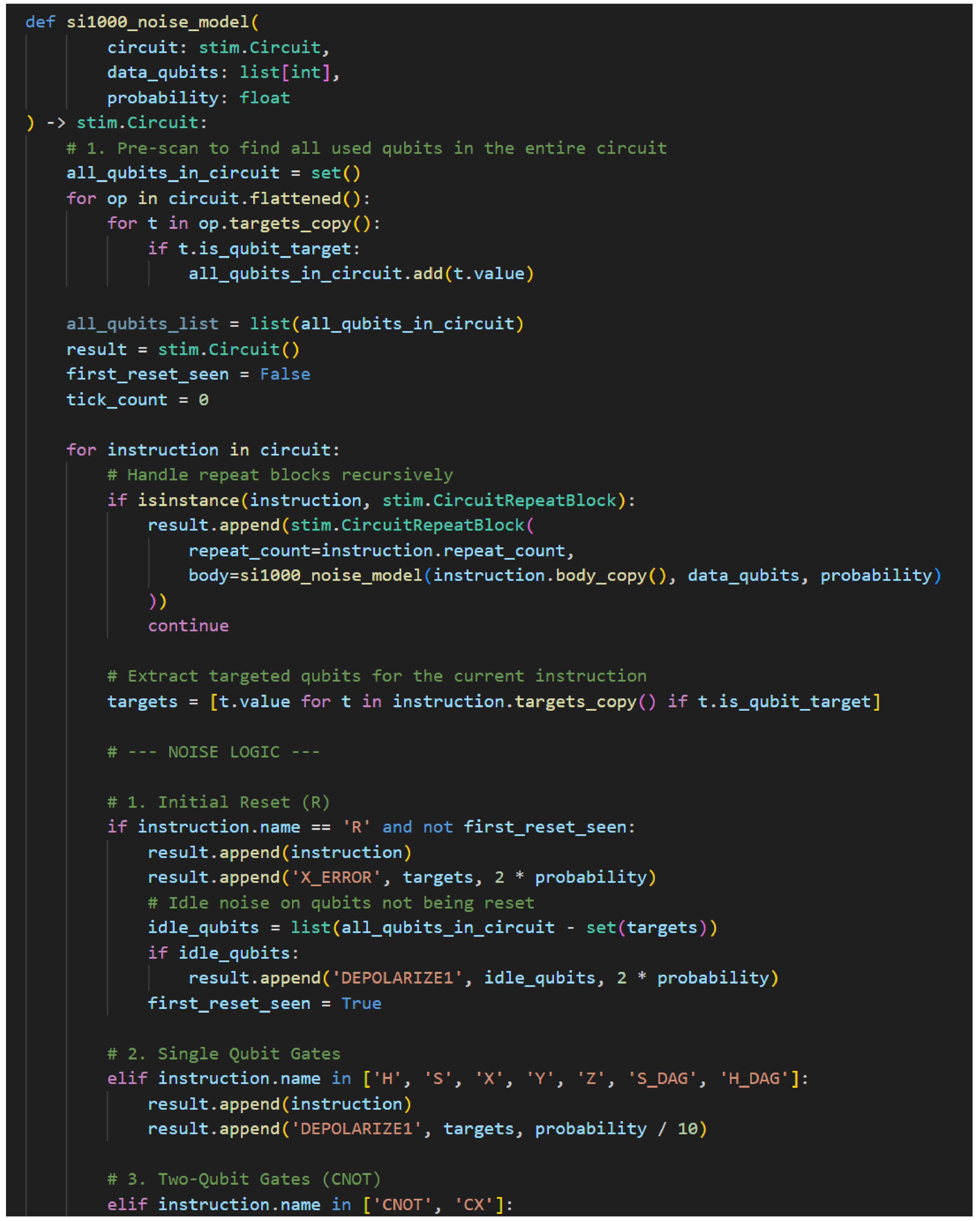
Progress

- Added noise for various noise models(Standard circuit-level Noise model, SI1000 Noise model).



```python
def standard_depolarizing_noise_model(
        circuit: stim.Circuit,
        data_qubits: list[int],
        after_clifford_depolarization: float,
        after_reset_flip_probability: float,
        before_measure_flip_probability: float,
        before_round_data_depolarization: float
) -> stim.Circuit:
    result = stim.Circuit()
    first_reset_seen = False
    tick_count = 0

    for instruction in circuit:
        if isinstance(instruction, stim.CircuitRepeatBlock):
            result.append(stim.CircuitRepeatBlock(
                repeat_count=instruction.repeat_count,
                body=standard_depolarizing_noise_model(
                    instruction.body_copy(),
                    data_qubits,
                    after_clifford_depolarization,
                    after_reset_flip_probability,
                    before_measure_flip_probability,
                    before_round_data_depolarization
                )))
        elif instruction.name == 'R' and not first_reset_seen:
            result.append(instruction)
            result.append('X_ERROR', instruction.targets_copy(), after_reset_flip_probability)
            first_reset_seen = True
        elif instruction.name in ['CNOT', 'CX', 'CZ']:
            result.append(instruction)
            result.append('DEPOLARIZE2', instruction.targets_copy(), after_clifford_depolarization)
        elif instruction.name == 'MR':
            result.append('X_ERROR', instruction.targets_copy(), before_measure_flip_probability)
            result.append(instruction)
            result.append('X_ERROR', instruction.targets_copy(), after_reset_flip_probability)
        elif instruction.name == 'M':
            result.append('X_ERROR', instruction.targets_copy(), before_measure_flip_probability)
            result.append(instruction)
        elif instruction.name == 'TICK':
            result.append(instruction)
            tick_count += 1
            # Assuming a standard surface code schedule where a round is ~9 ticks?
            # Note: Your specific circuit might not follow the 9-tick structure exactly.
            # This logic adds idle noise at start of rounds.
            if first_reset_seen and before_round_data_depolarization > 0:
                # Simple heuristic: Apply if it looks like the start of a round block
                # or simply apply it if your schedule relies on TICKs.
                # Original code logic: if tick_count >= 2 and (tick_count - 1) % 9 == 1:
                # Adjusting to apply generally for this example:
                pass
        else:
            result.append(instruction)
```



```python
def si1000_noise_model(
        circuit: stim.Circuit,
        data_qubits: list[int],
        probability: float
) -> stim.Circuit:
    # 1. Pre-scan to find all used qubits in the entire circuit
    all_qubits_in_circuit = set()
    for op in circuit.flattened():
        for t in op.targets_copy():
            if t.is_qubit_target:
                all_qubits_in_circuit.add(t.value)

    all_qubits_list = list(all_qubits_in_circuit)
    result = stim.Circuit()
    first_reset_seen = False
    tick_count = 0

    for instruction in circuit:
        # Handle repeat blocks recursively
        if isinstance(instruction, stim.CircuitRepeatBlock):
            result.append(stim.CircuitRepeatBlock(
                repeat_count=instruction.repeat_count,
                body=si1000_noise_model(instruction.body_copy(), data_qubits, probability)
            ))
            continue

        # Extract targeted qubits for the current instruction
        targets = [t.value for t in instruction.targets_copy() if t.is_qubit_target]

        # --- NOISE LOGIC ---

        # 1. Initial Reset (R)
        if instruction.name == 'R' and not first_reset_seen:
            result.append(instruction)
            result.append('X_ERROR', targets, 2 * probability)
            # Idle noise on qubits not being reset
            idle_qubits = list(all_qubits_in_circuit - set(targets))
            if idle_qubits:
                result.append('DEPOLARIZE1', idle_qubits, 2 * probability)
            first_reset_seen = True

        # 2. Single Qubit Gates
        elif instruction.name in ['H', 'S', 'X', 'Y', 'Z', 'S_DAG', 'H_DAG']:
            result.append(instruction)
            result.append('DEPOLARIZE1', targets, probability / 10)

        # 3. Two-Qubit Gates (CNOT)
        elif instruction.name in ['CNOT', 'CX']:
```

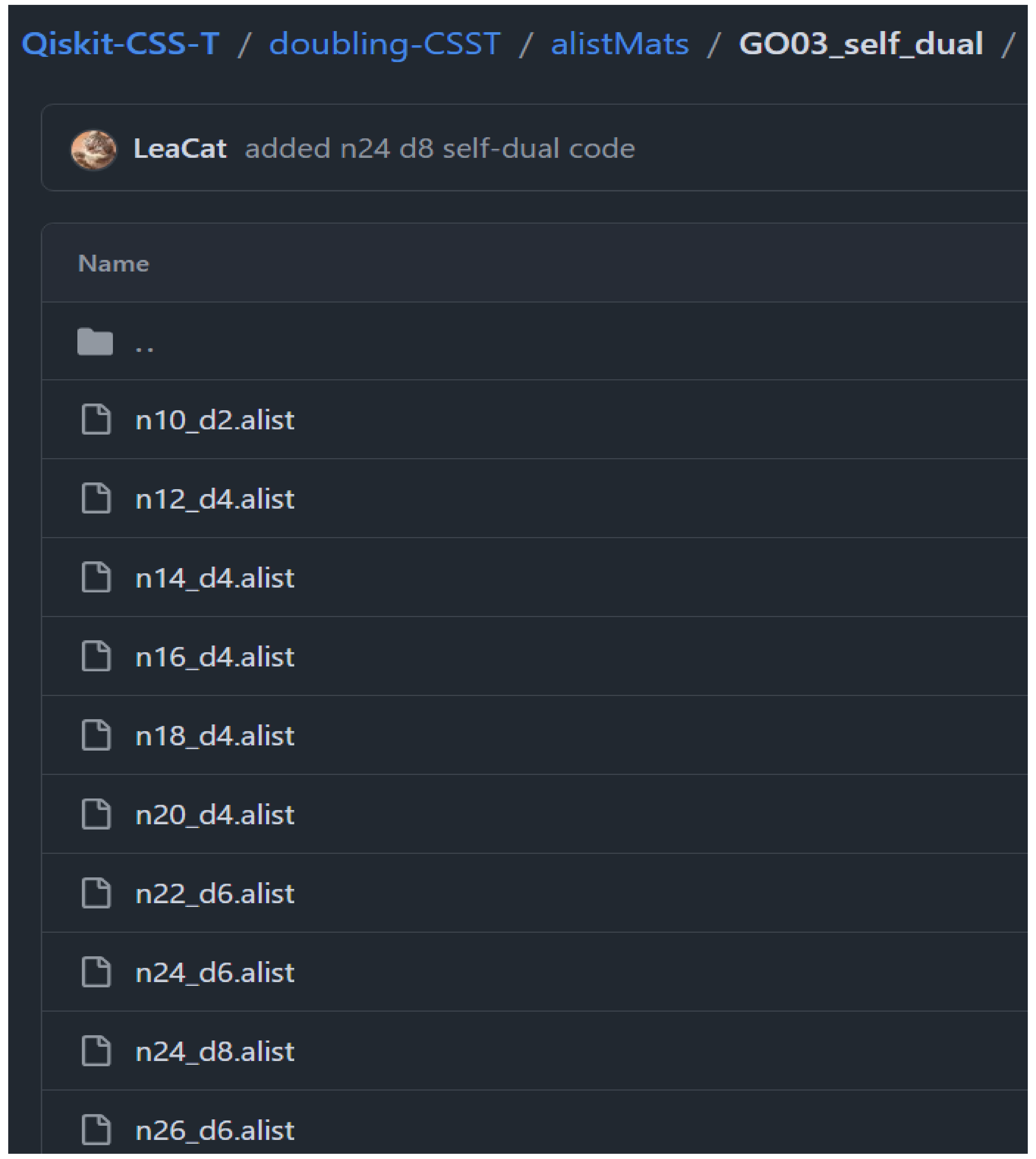
Circuit-level noise simulation for self-dual CSS codes via Stim

Objective

Studying the error capacity of our self-dual CSS/CSS-T codes.

Observation and Discussion

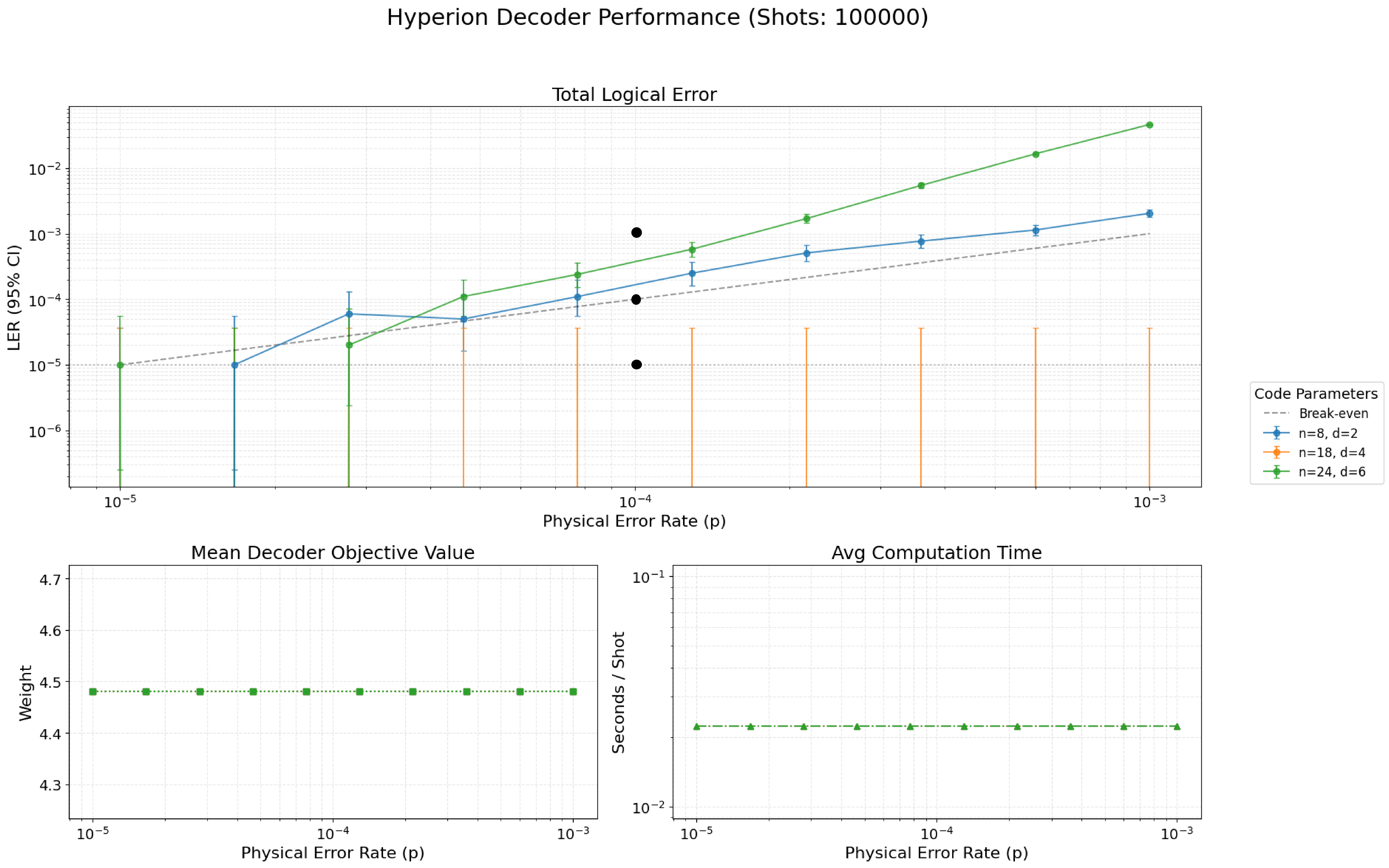
- Implanted the self-dual CSS codes and got the standard depolarizing circuit-level noise simulation results.



Qiskit-CSS-T / doubling-CSST / alistMats / GO03_self_dual /

LeaCat added n24 d8 self-dual code

Name
..
n10_d2.alist
n12_d4.alist
n14_d4.alist
n16_d4.alist
n18_d4.alist
n20_d4.alist
n22_d6.alist
n24_d6.alist
n24_d8.alist
n26_d6.alist



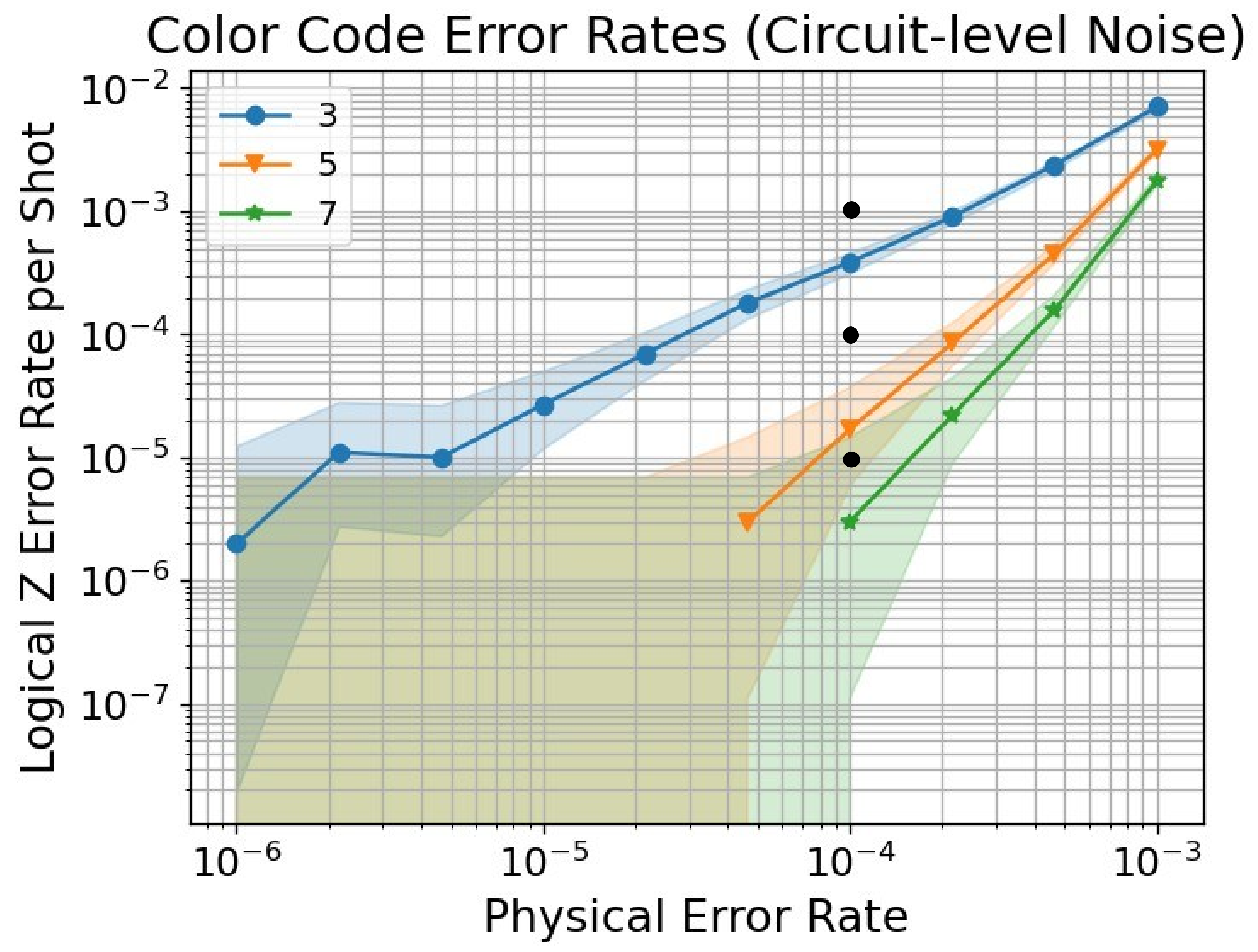
Circuit-level noise simulation for self-dual CSS codes via Stim

Objective

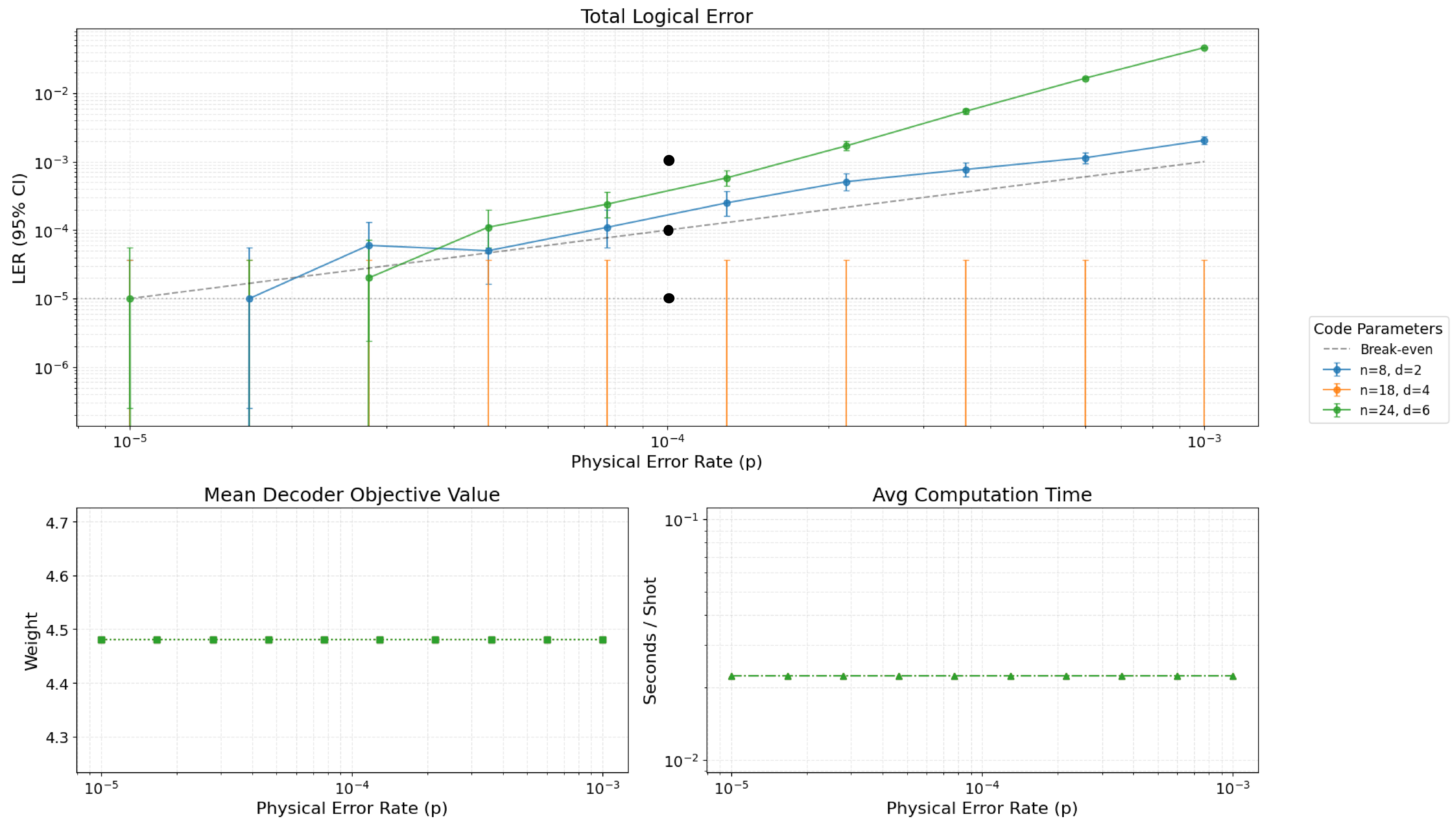
Studying the error capacity of our self-dual CSS/CSS-T codes.

Observation and Discussion

- Implanted the self-dual CSS codes and got the standard depolarizing circuit-level noise simulation results.
- Distance 2 results are comparable and distance 6 worked worse than the Color codes, but distance 4 results went under the resolution limit which is quite exciting!



Hyperion Decoder Performance (Shots: 100000)



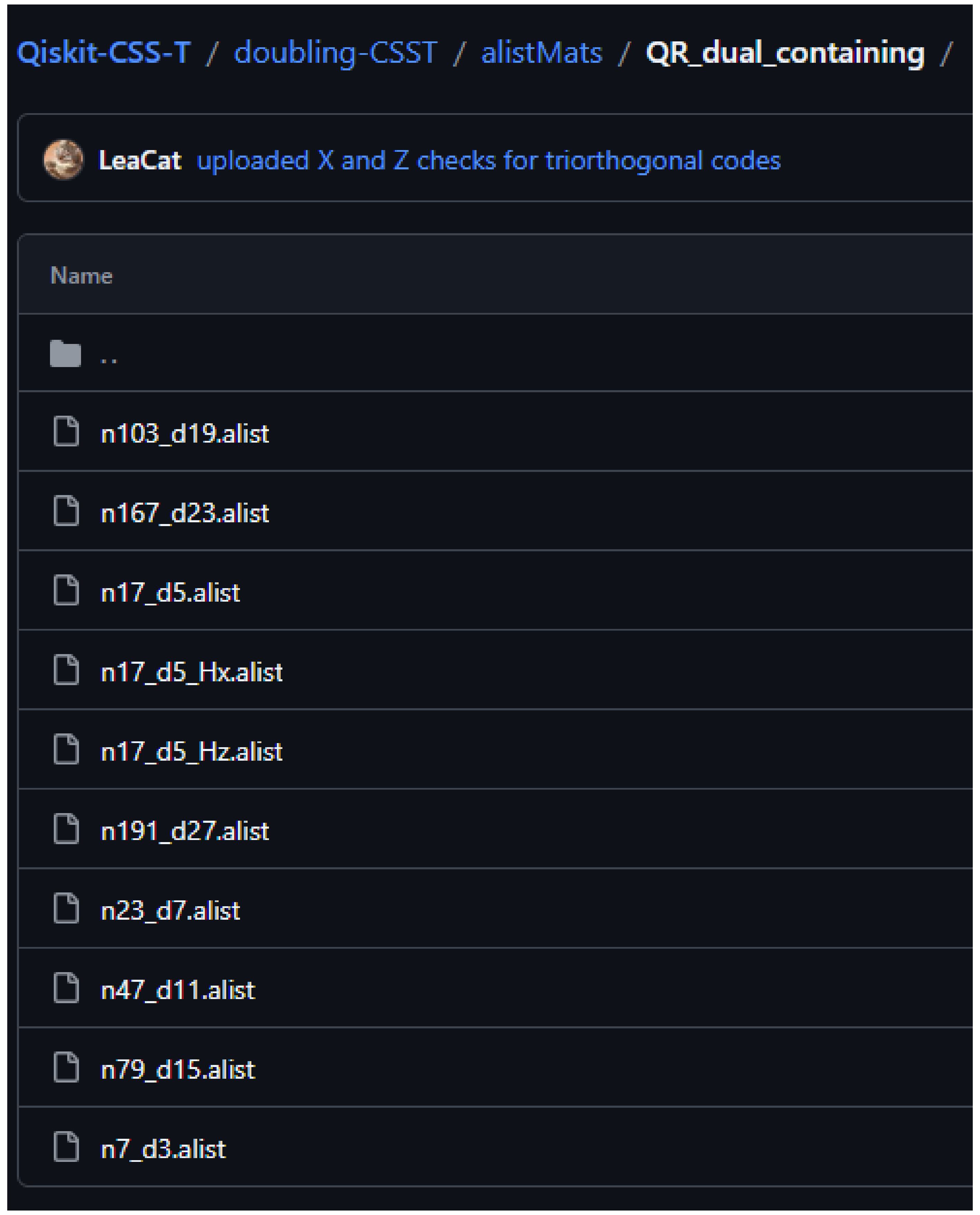
Circuit-level noise simulation for self-dual CSS codes via Stim

Objective

Studying the error capacity of our self-dual CSS/CSS-T codes.

Observation and Discussion

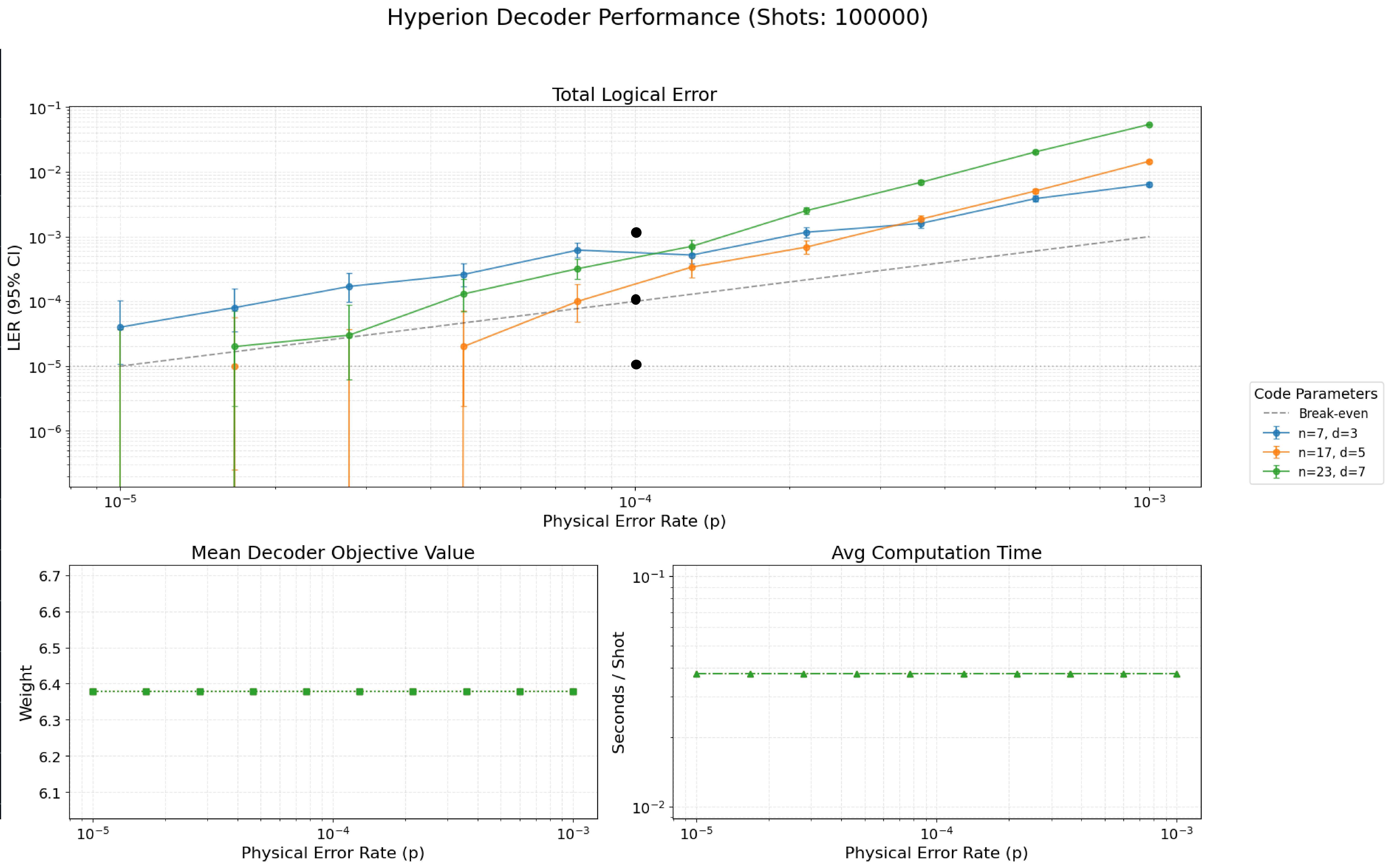
- Implanted the dual-containing CSS codes and got the standard depolarizing circuit-level noise simulation results.



Qiskit-CSS-T / doubling-CSST / alistMats / QR_dual_containing /

LeaCat uploaded X and Z checks for triorthogonal codes

Name
..
n103_d19.alist
n167_d23.alist
n17_d5.alist
n17_d5_Hx.alist
n17_d5_Hz.alist
n191_d27.alist
n23_d7.alist
n47_d11.alist
n79_d15.alist
n7_d3.alist



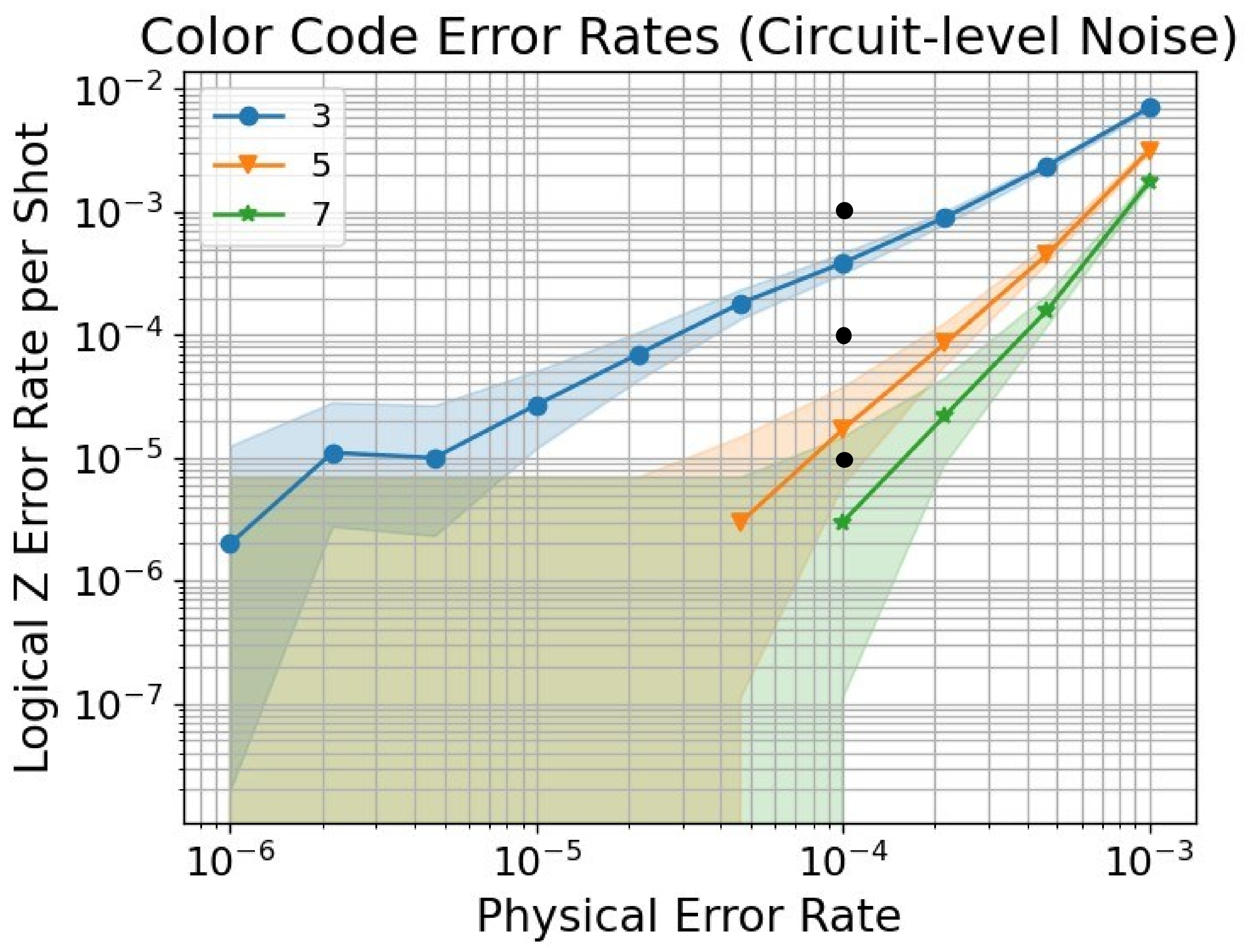
Circuit-level noise simulation for self-dual CSS codes via Stim

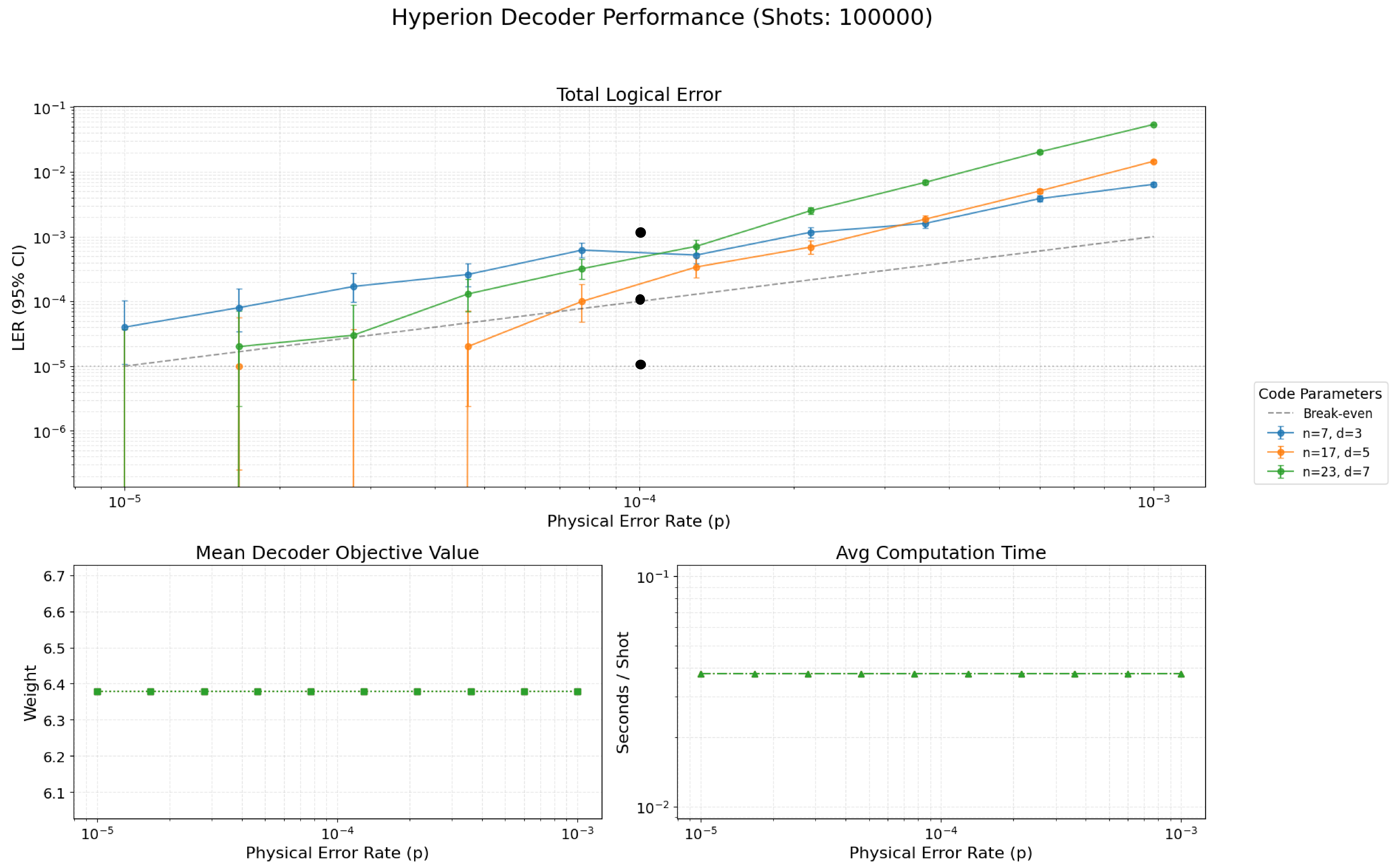
Objective

Studying the error capacity of our self-dual CSS/CSS-T codes.

Observation and Discussion

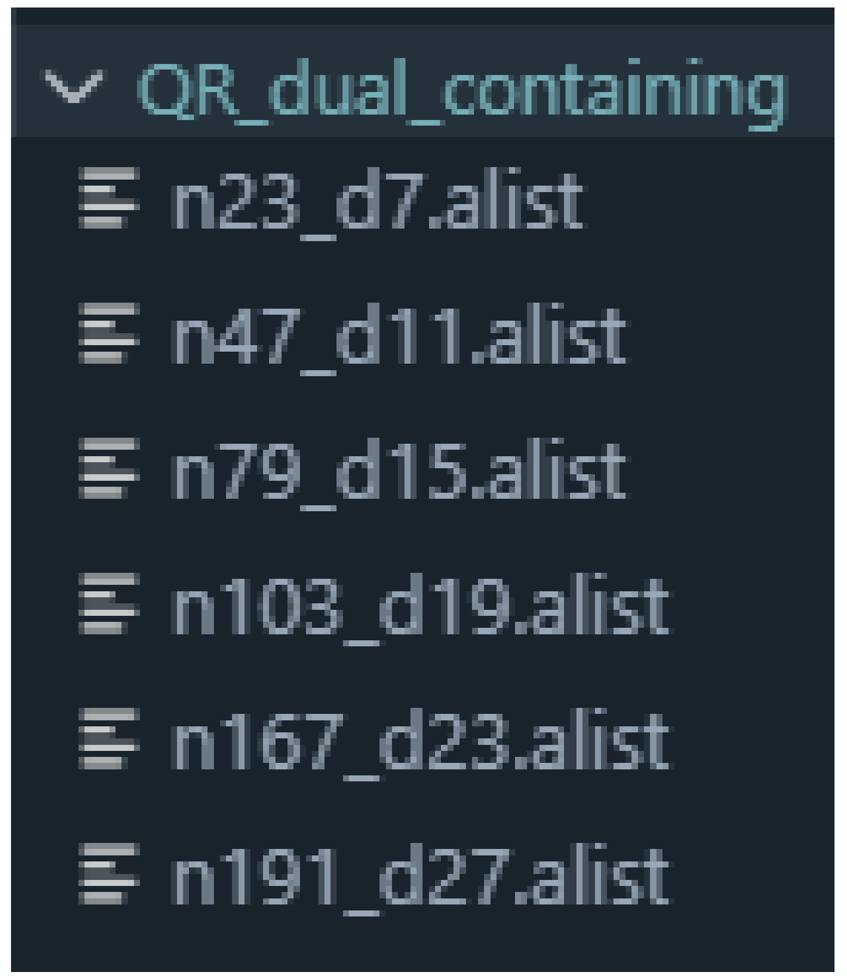
- Implanted the dual-containing CSS codes and got the standard depolarizing circuit-level noise simulation results.
- The dual containing codes results were worse from the examples tested, barely touching the break-even.



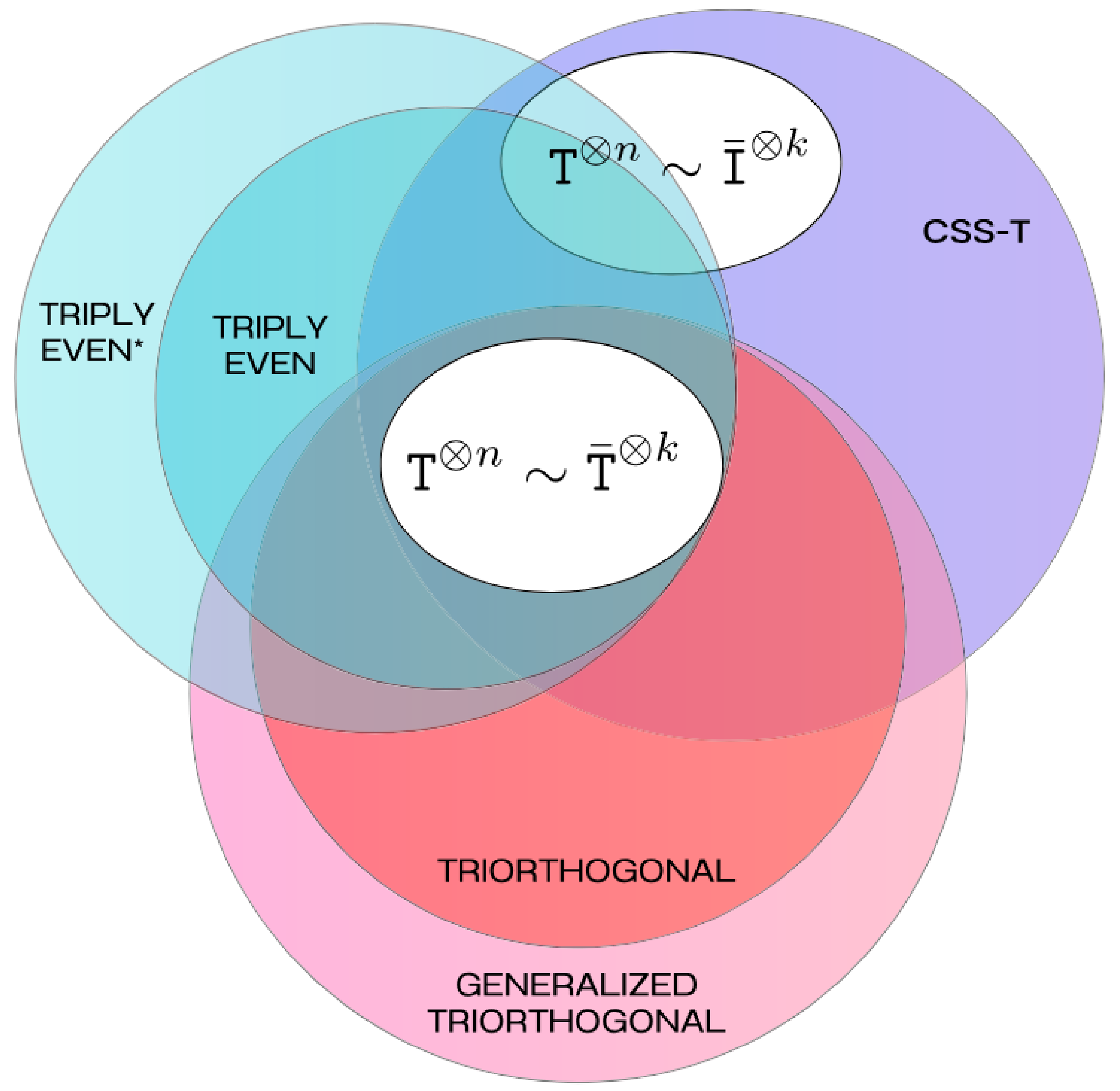


Future work

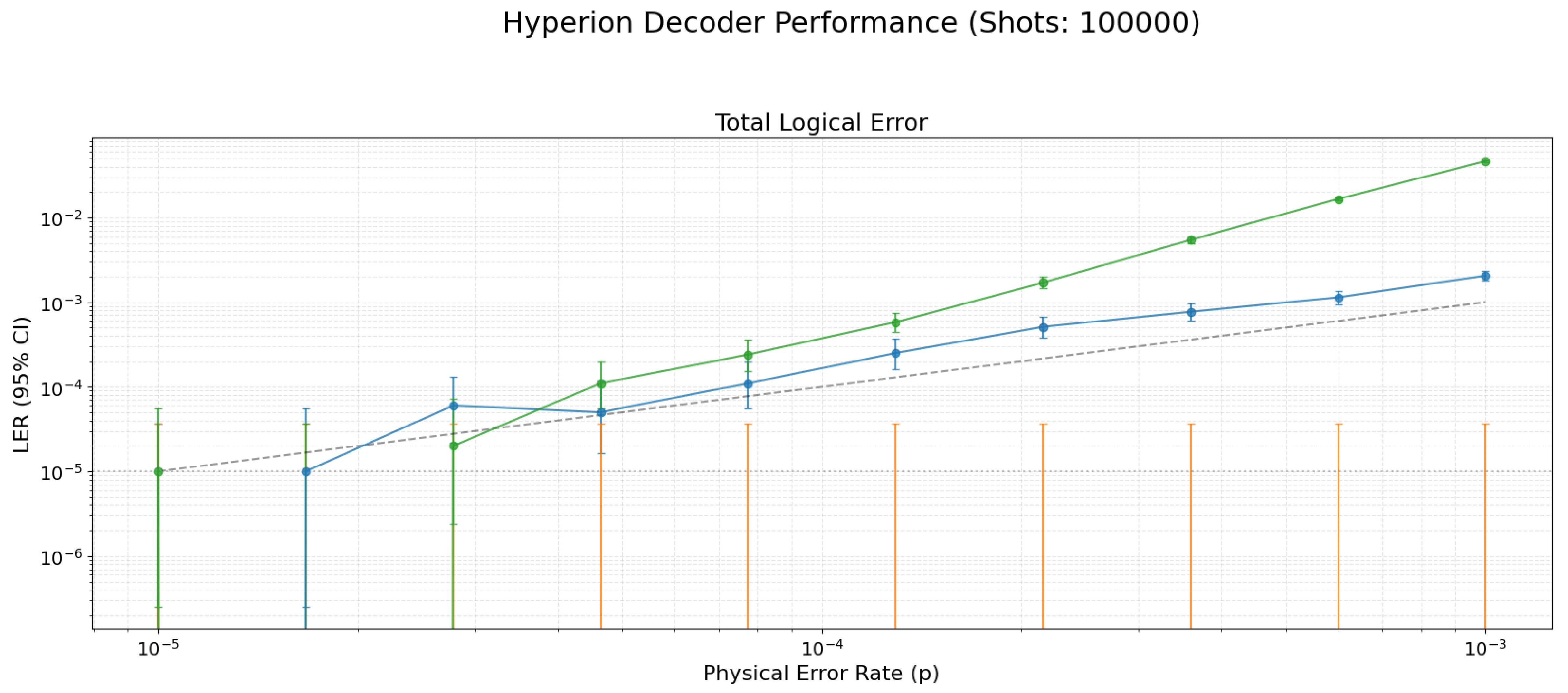
- Looking for self-dual codes bridging the distance gap
- Going to test decoding of Tri-orthogonal codes which has Transversal T gates.



```
QR_dual_containing
  n23_d7.alist
  n47_d11.alist
  n79_d15.alist
  n103_d19.alist
  n167_d23.alist
  n191_d27.alist
```



$T^{\otimes n} \sim \bar{I}^{\otimes k}$
CSS-T
TRIPLY EVEN*
TRIPLY EVEN
$T^{\otimes n} \sim \bar{T}^{\otimes k}$
TRIORTHOGONAL
GENERALIZED TRIORTHOGONAL



QISKIT ADVOCATE

2025 2025

MENTORSHIP PROGRAM