Single-Shot Decoding of Biased-Tailored Quantum LDPC Codes

Devon Campbell[*]
Columbia University
(Dated: May 15, 2025)

Current quantum devices can suffer performance degradation due to biased physical noise and faulty stabilizer readout. This work unifies two previously separate strategies for addressing these challenges—bias tailoring and single-shot decoding—within a single architecture. I construct a four-dimensional lifted homological-product (4D-LHP) code from four classical quasi-cyclic matrices, embedding syndrome-validating *metachecks* and enabling adaptation to biased noise channels. A targeted Hadamard rotation aligns the stabilizer structure with the dominant error axis. Using this construction, this work presents a $[[384, 48, 6]]$ 4D-LHP code that corrects both data and measurement faults.

Monte Carlo simulations under a circuit-level noise model show that the tailored rotation lowers the word error rate (WER) by $20-60\%$ for bias ratios $10^{-3} \leq \eta \leq 10^{-1}$, and still yields a $\sim 10\%$ gain at $\eta \approx 1$. With 4% measurement error, a *single* SS round recovers roughly one-third of the logical performance lost to readout noise. Metachecks detect 99.1% of faulty syndromes and correct 7.4%—enough to materially reduce the overall WER. These results demonstrate that combining bias tailoring with SS decoding charts a viable path toward bias-tailored, measurement-robust quantum memories on near-term hardware.

I. INTRODUCTION

Noisy intermediate-scale quantum (NISQ) devices are fundamentally constrained by the high error rates of quantum bits (qubits), making robust error correction essential for reliable computation [1]. However, quantum error correction (QEC) is itself an imperfect process, which can misidentify errors or introduce new ones through faulty corrections [2]. To mitigate the introduction of errors during QEC, stabilizer measurements are performed repeatedly and analyzed via classical post-processing to estimate the most likely syndrome outcomes [3].

However, the number of required repetitions typically scales with code size, increasing circuit depth, execution time, and the number of physical operations—all of which exacerbate the likelihood of introducing noise [3]. Alternative strategies have been proposed to improve the reliability of stabilizer measurements. For instance, *single-shot* (SS) error correction avoids repeated measurements by enabling the correction of both data and measurement errors in a single noisy measurement round, significantly reducing QEC overhead [4].

SS decoding relies on geometric redundancy: additional structure in the code that enables *metachecks*, which apply parity constraints directly to the stabilizer measurement outcomes [3]. This redundancy allows a single noisy syndrome extraction to reveal information about both the underlying data error and any faults in the measurement process itself.

Conventional stabilizer codes are typically two-dimensional, with independent X- and Z-type checks. However, 2D constructions lack sufficient degrees of freedom to impose independent parity checks on the syndrome bits. To support metachecks, one must increase the dimensionality of the code to embed additional constraints on the measured syndromes.

For example, *three-dimensional* homological product codes achieve SS decoding in the *infinite-bias limit*, where only a single Pauli error type (e.g., Z) is dominant. In this regime, the code can detect and correct both data and measurement errors, but it only supports metachecks for the relevant stabilizer type—leaving the complementary sector (e.g., X stabilizer checks) unprotected [5].

To achieve full single-shot capability under mixed Pauli noise, an additional homological layer is required. Four-dimensional chain complexes provide two distinct metacheck matrices, M_X and M_Z, which impose independent constraints on X- and Z-type stabilizers, respectively [6]. This structure enables the simultaneous detection of measurement faults in both sectors, supporting reliable single-round decoding under realistic, circuit-level noise.

Even when temporal overhead is minimized, QEC still demands many physical qubits to realize a single logical qubit. Most canonical codes are engineered for *depolarizing* noise, where bit-flip (X) and phase-flip (Z) errors occur with equal probability. In real hardware, however, the error channel is often *biased*: one Pauli error dominates while the others are strongly suppressed. For instance, superconducting cat qubits exponentially suppress bit-flips, producing a noise profile in which Z errors vastly outnumber X errors [7].

Ignoring this asymmetry squanders coding resources. A depolarizing-optimized code allocates half of its check operators to detect the minority error, leaving the dominant error under-protected. The result is a lower effective distance in the high-bias direction, a dramatically reduced threshold, and a larger qubit overhead to reach a given logical-error target. In extreme cases, the logical error rate can be *worse* than if one simply encoded the qubit in a repetition code matched to the dominant er-

[*] dec2180@columbia.edu

ror. Consequently, a code that remains "un-twisted" in a biased channel wastes parity checks on rare events while offering little resistance to the common ones.

To exploit the underlying asymmetry, researchers have devised bias-preserving gates and *bias-tailored* codes that realign stabilizers with the physical noise, cutting overhead and boosting performance [8]. A prominent example is the *XZZX surface code*, obtained by applying a Hadamard rotation to half of the qubits so that each plaquette involves a balanced mix of X and Z operators [9].

In the infinite-bias limit this "twist" decomposes the lattice into independent repetition codes; the logical Z distance grows, allowing the code to tolerate Z-error rates up to 50 %. Even at moderate bias the threshold climbs to $= 10.9\%$, compared with 1% for the untailored surface code.

Geometrically, the twist forces logical operators—originally short, straight-line paths across the untwisted surface code—to follow longer, diagonal trajectories. This increases the number of dominant errors required to span the code and cause a logical failure. In short, bias tailoring converts hardware asymmetry from a liability into an advantage, whereas neglecting bias leaves the code fragile along its most vulnerable axis.

Further, bias-tailoring is not confined to the XZZX surface code. The XZZX construction can be understood as a special instance of the *hypergraph product* (HGP) code, a broader framework introduced by Tillich and Zémor [10]. This construction takes as input two classical binary linear codes—specified by their parity-check matrices H_1 and H_2—and produces a quantum CSS code whose stabilizers are derived from the tensor product structure of the seeds. The resulting quantum code inherits the LDPC properties of the classical inputs and can achieve favorable distance and rate scaling depending on the seed choices.

In particular, the XZZX surface code arises when both classical seeds are length-L repetition codes. This yields a square lattice with stabilizers formed from combinations of X and Z operators on four-qubit plaquettes. A local Hadamard rotation on half the qubits then transforms the original surface code into its XZZX variant, aligning the stabilizers to better withstand biased noise.

More generally, Roffe *et al.* showed that the same "twist"—a tailored Hadamard rotation that reorients the stabilizer structure—can be applied to *any* HGP code [11]. Their work then investigates the *lifted product* framework introduced by Panteleev and Kalachev, which generates sparse quantum codes from a pair of quasi-cyclic matrices and achieves asymptotically growing distance and rate [12].

By adapting the twisting procedure beyond repetition codes to arbitrary seed pairs, Roffe's approach broadens the applicability of bias tailoring from a single surface-code variant to an entire class of quantum LDPC codes. This unlocks new opportunities for optimizing rate–distance trade-offs in the presence of asymmetric noise.

This work combines bias-tailoring design principles with the *single-shot* (SS) decoding paradigm. Starting from *four* classical seeds $\{\delta_A, \delta_B, \delta_C, \delta_D\}$, I build a four-dimensional lifted hypergraph-product (4D-LHP) code.

Two key features distinguish this construction:

- **Hadamard-based bias tailoring.** By applying local Hadamard rotations to half of the qubits, the code's stabilizers are "twisted" in the manner of bias-tailored lifted product codes. This operation realigns the stabilizer support to increase the minimum weight of logical operators in the dominant error basis—effectively increasing the distance of the code under dephasing-biased noise.

- **Metachecks for single-shot decoding.** Additional parity checks, known as metachecks, are embedded one homological layer above and below the conventional stabilizer layers. These checks enable single-shot decoding à la Bombín [3, 6], allowing both data and measurement faults to be inferred from a single round of noisy syndrome extraction.

These mechanisms address complementary challenges: bias tailoring improves resilience against *data errors*—particularly those concentrated in a single Pauli basis—while SS decoding mitigates *measurement errors*, which would otherwise necessitate repeated and costly rounds of syndrome readout. Their combination results in a doubly protected architecture with minimal overhead.

The remainder of this paper is organized as follows. Section II reviews the relevant background on stabilizer codes, biased noise, and single-shot decoding. Section III presents the 4D-LHP construction, detailing the chain complex, metacheck structure, and bias-tailored Hadamard transformation. Further, this section describes the simulation methodology, including the noise model, decoding algorithm, and protograph lifting. Section IV reports numerical results, evaluating performance under varying bias and measurement noise. Finally, Section V discusses the synergy of the two design principles, potential optimizations, and directions for future work.

II. BACKGROUND

A. Stabilizer Codes

A quantum error-correcting (QEC) code typically uses n physical qubits to encode k logical qubits. The resulting code is denoted as $[n, k, d]$, where d is the *distance*—the minimum number of physical qubit errors required to induce a logical failure. A larger distance implies stronger error protection.

Error correction is achieved by defining a set of mutually commuting operators called *stabilizers*, which constrain the system to a protected subspace known as the code space. A stabilizer code has $n - k$ independent stabilizer generators S_i, each composed of Pauli operators acting on the n qubits. Valid codewords are quantum states $|\psi\rangle$ that satisfy $S_i |\psi\rangle = |\psi\rangle$ for all i.

When errors occur, measuring the stabilizers yields a binary syndrome that encodes partial information about the type and location of the error. The decoder uses this syndrome to infer a correction that returns the system to the code space, ideally without disturbing the encoded logical state.

A CSS code is a stabilizer code defined by two binary parity-check matrices H_X and H_Z, which determine the structure of the stabilizer generators:

1. Each row of H_X defines an **X-type stabilizer** (composed of Pauli-X operators), which detect Z errors (phase-flips)
2. Each row of H_Z defines a **Z-type stabilizer** (composed of Pauli-Z operators), which detect X errors (bit-flips)

These matrices must satisfy the commutation condition: $H_X H_Z^T = 0 \in \mathbb{F}_2$. Further, the columns of these matrices correspond to the physical qubits that upon which these stabilizers act. For any entry $a = (i, j) \in H_X$, a value of $a = 1$ indicates that stabilizer S_i acts nontrivially on qubit j, and can detect a Z-error via anticommutation. Similarly, entries in H_Z indicate sensitivity to X-errors.

Suppose an error $e \in \mathbb{F}_2^n$ occurs on the physical qubits. The syndromes associated with that error are: $s_X = H_X e, \quad s_Z = H_Z e$. If $s_X \neq 0$, at least one X=type stabilizer detected an inconsistency, implying there was likely a Z-type error. The same logic applies to s_Z for X-errors. The role of the *decoder* is to infer a correction operation $r \in \mathbb{F}_2^n$ based on the observed syndrome s, such that applying r cancels out the error and returns the system to the code space.

The hypergraph product (HGP) constructs a CSS code from any pair of classical binary codes with parity-check matrices $H_1 \in \mathbb{F}_2^{m_A \times n_A}$ and $H_2 \in \mathbb{F}_2^{m_B \times n_B}$. The stabilizer generators are given by:

$$\begin{aligned} H_{\text{HGP}} &= [H_X \mid H_Z] \qquad (1) \\ &= \left[\begin{array}{cc|cc} 0 & 0 & \mathbb{I}_{n_1} \otimes H_2 & H_1^T \otimes \mathbb{I}_{m_2} \\ H_1 \otimes \mathbb{I}_{n_2} & \mathbb{I}_{m_1} \otimes H_2^T & 0 & 0 \end{array}\right] \end{aligned}$$

This guarantees $H_X H_Z^T = 0$, so the stabilizers commute and define a valid CSS code. The resulting code is sparse, LDPC, and inherits its logical structure and distance scaling from the input codes [10].

More generally, the HGP construction is a special case of the *homological product code* framework. In this setting, stabilizer codes are modeled using a chain complex—a sequence of vector spaces $\mathcal{C}_j$ connected by boundary maps δ_j satisfying $\delta_j \circ \delta_{j+1} = 0$ [13, 14]. Each space $\mathcal{C}_j$ represents a collection of objects (e.g., checks or qubits), and the maps encode how those objects interact.

The *dimension* of a homological code refers to the length of this chain complex. For example, a two-dimensional code, like those produced by the HGP construction, corresponds to a chain complex consisting of three vector spaces connected by two boundary maps, :

$$\mathcal{C}_{-1} \xrightarrow{\delta_{-1}} \mathcal{C}_0 \xrightarrow{\delta_0} \mathcal{C}_1,$$

where $\mathcal{C}_0$ represents the physical qubits, $\mathcal{C}_1$ represents the Z-type stabilizer checks, and $\mathcal{C}_{-1}$ represents the X-type stabilizer checks. The maps δ_0 and δ_{-1} correspond to the parity-check matrices H_X and H_Z^T, respectively.

This structure allows one to generalize the construction to higher dimensions by adding additional layers of checks and metachecks, as is required for single-shot decoding [3]. In particular, increasing the number of classical seed matrices used in the tensor construction extends the length of the chain complex: each additional seed introduces a new homological layer and a corresponding boundary map.

For example, while the HGP uses two classical codes and forms a length-2 complex, a 3D homological product code uses three seeds to form a length-3 complex [5], and a 4D construction requires four seeds and yields a length-4 chain. This expansion enables the inclusion of metachecks—higher-layer parity checks that constrain the syndromes themselves—and is essential for achieving full single-shot decoding in the presence of general Pauli noise.

B. Single-Shot Decoding

In realistic devices, syndrome extraction is itself error-prone: each stabilizer measurement may flip due to gate or readout noise. The conventional solution is to repeat the measurement several times and use temporal correlations to infer the true syndrome [6].

By contrast, single-shot (SS) codes can tolerate a single round of noisy syndrome measurements and still reliably identify data and measurement errors. To achieve this, SS codes are designed with redundant stabilizers that ensure measurement faults imprint a consistent and detectable signature—known as the *metasyndrome*—onto the measured syndrome [6].

This redundancy is encoded through metachecks, which impose consistency conditions on syndrome measurements. Formally, the metacheck matrix M imposes a consistency condition on the observed syndrome s. For a physical error E, the syndrome map $\sigma(E)$ satisfies $Ms = M\sigma(E) = 0$ in the absence of measurement noise. If a measurement error u occurs, the observed syndrome becomes $s = \sigma(E) + u$, and the metacheck output becomes $Ms = Mu$. Because the metacheck output depends only on the measurement error u, a nonzero metasyndrome Ms signals the presence of measurement faults and can be used to localize and correct them.

Recent work proved that *three-dimensional* homological product (3D-HP) codes exhibit *confinement*, a structural property that plays a critical role in enabling single-shot decoding [5]. Confinement ensures that the effects of physical and measurement errors remain spatially localized—preventing small faults from producing widespread, correlated syndrome patterns. This locality makes the decoding problem tractable even when syndromes are noisy: faulty measurements generate constrained, predictable patterns that can be detected and corrected without requiring repeated rounds of syndrome extraction.

In the 3D setting, confinement is sufficient to support single-shot decoding in the *infinite-bias* limit, where only one Pauli error (e.g., Z) dominates. However, because 3D-HP codes can only embed metachecks for a single stabilizer type, they are not fully single-shot under general Pauli noise.

From a practical standpoint, these codes can be decoded efficiently using *belief propagation and ordered statistics decoding* (BP+OSD), a two-stage strategy that combines speed with robustness. The first stage applies BP to estimate a likely error configuration by passing probabilistic messages across the Tanner graph.

The second stage then refines this estimate using OSD, a greedy search that selects low-weight corrections consistent with the observed syndrome [15]. This approach handles both data and measurement noise and is well suited to codes with sparse, redundant checks—conditions naturally met by homological product constructions.

Importantly, BP+OSD benefits from *syndrome redundancy*—the presence of overlapping constraints that provide multiple ways to validate or refute a syndrome bit. In HP codes, this redundancy arises geometrically: each physical qubit or stabilizer participates in multiple tensor-product layers, allowing small faults to be detected through several independent syndromes. This multi-layered structure not only boosts detection but also improves decoder convergence and accuracy under realistic noise.

In the 4D-LHP code introduced in this work, these principles are extended one dimension higher. The use of four classical seed matrices generates a length-4 chain complex, in which both X- and Z-type stabilizers are independently metachecked. The same BP+OSD machinery applies to each decoding stage: one round to clean the noisy syndrome using metachecks, and another to recover from data errors using the corrected syndrome. Together, these steps constitute a practical, scalable single-shot decoder for general Pauli noise.

C. Biased Noise

Exploiting biased noise has become a prominent strategy to improve QEC performance. The archetypal example of this strategy is the *XZZX surface code*: by applying a Hadamard to half of the qubits on a square lattice, the code's stabilizers change from pure-X and pure-Z plaquettes, to mixed four-body XZZX operators (See Fig. 1) [9, 16]. This Hadamard rotation has two immediate impacts under a dephasing-dominated channel ($p_Z \gg p_X$)

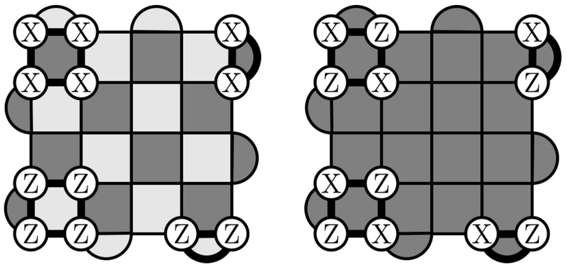

FIG. 1: Transformation from a standard surface code (left) to the XZZX code (right) via Hadamard rotation on alternating qubits. In the standard code, stabilizers alternate between pure X and Z types. The XZZX variant mixes Pauli operators within each stabilizer (e.g., $X\otimes Z\otimes Z\otimes X$) [9].

1. **Distance boost.** Logical Z operators must follow longer, diagonal paths across the lattice, increasing the minimum weight for damaging Z-error chains. For a rotated $L \times L$ code, this boosts the Z-distance from $O(L)$ to $O(L^2)$ in ideal bias-tailored layouts. More generally, the logical Z-distance of an N-dimensional Clifford-deformed code scales as $O(L^N)$ [17, 18].

2. **Higher thresholds.** In the infinite bias regime, where only Z errors occur, the XZZX code achieve a threshold of $p_Z^{\text{th}} = 50\%$ [9]. Under moderate bias, such as a 4:1 ratio of Z to X errors, the threshold remains high at $p_Z^{\text{th}} \approx 10\%$. Both biased thresholds represent substantial improvements over the threshold of the standard surface code under depolarized noise: $p^{\text{th}} \approx 1\%$ [19].

Roffe *et al.* extended the bias-tailoring strategy of the XZZX code to general quantum LDPC constructions by incorporating Hadamard rotations directly into the code's structure. Since the columns of the parity-check matrices H_X and H_Z index physical qubits, applying Hadamard gates to half of the qubits corresponds to applying the Hadamard transformation to the matching columns in both matrices.

Because the Hadamard swaps Pauli bases ($X \overset{H}{\leftrightarrow} Z$), this operation effectively exchanges the roles of X- and Z-type checks on the targeted qubits. As a result, applying Hadamards to half of the qubits is equivalent to swapping the rightmost halves of H_X and H_Z in Eq. 1, yielding mixed-type stabilizers optimized for dephasing-biased noise (Eq. 2).

$$\begin{aligned} H' &= [H_{X_1}\ H_{Z_2} \mid H_{Z_1}\ H_{X_2}] \\ &= \left[\begin{array}{cc|cc} 0 & H_1^T \otimes \mathbb{I}_{m_2} & \mathbb{I}_{n_1} \otimes H_2 & 0 \\ H_1 \otimes \mathbb{I}_{n_2} & 0 & 0 & \mathbb{I}_{m_1} \otimes H_2^T \end{array}\right] \end{aligned} \tag{2}$$

This column-swapping approach to bias tailoring is broadly applicable. For example, Hadamard rotations can be incorporated directly into the *lifted hypergraph product* (LHP) construction [11]. LHP codes follow the same structural blueprint as standard hypergraph product (HGP) codes, but instead of binary parity-check matrices, they take *protographs* as inputs—small symbolic matrices with entries drawn from the *ring of circulants*.

Each ring element λ_L^α represents a rightward cyclic shift of the $L \times L$ identity matrix by α positions. For example, λ_3^1 corresponds to the 3×3 identity matrix with its rows shifted one position to the right:

$$\lambda_3^1 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

.

Each ring entry expands into an $L \times L$ binary matrix, where the lift parameter L specifies the block size. The use of circulants preserves structure, sparsity, and decoding efficiency while allowing algebraic design flexibility at the protograph level.

A matrix composed of these ring elements produces a protograph. For example, the protograph:

$$A_L = \begin{bmatrix} \lambda_L^1 + \lambda_L^2 & \lambda_L^0 & 0 \\ 0 & \lambda_L^0 + \lambda_L^1 & \lambda_L^1 \end{bmatrix}$$

defines a compact, high-level representation of a parity-check matrix [11]. When expanded with lift parameter $L = 3$, each entry becomes a sum of permutation matrices, and the full protograph expands into a binary matrix of size 6×9. This lifted binary matrix is shown below:

$$\mathfrak{B}(A_3) = \left[\begin{array}{ccc|ccc|ccc} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{array}\right].$$

This transformation from symbolic protograph to binary matrix retains structure and sparsity, and it enables systematic design of large, high-performance LDPC codes. The use of quasi-cyclic circulant blocks facilitates efficient encoding and decoding while maintaining desirable properties such as regularity and large girth [11].

Because the LHP code is an HGP code that uses protographs as input seeds, the qLDPC code is sparse, scalable, and exhibits highly efficient decoding. The protograph-level representation enables compact design and algebraic manipulation before expansion into full binary matrices.

Given the seed protographs A_1 and A_2, the lifted parity-check matrices are defined as:

$$\begin{aligned} A &= [A_X \mid A_Z] \\ &= \left[\begin{array}{cc|cc} 0 & 0 & \mathbb{I} \otimes A_2 & A_1^T \otimes \mathbb{I} \\ A_1 \otimes \mathbb{I} & \mathbb{I} \otimes A_2^T & 0 & 0 \end{array}\right] \end{aligned} \tag{3}$$

These matrices are then lifted to binary form using a chosen lift parameter L, where each protograph entry expands into an $L \times L$ circulant matrix. The H_X and H_Z matrices are derived by lifting A_X and A_Z into their binary forms. Like the hypergraph product for binary matrices, the lifted product enables quantum code construction from arbitrary pairs of protographs [11].

Critically, bias tailoring can be applied directly at the protograph level: a Hadamard rotation on half the qubits corresponds to swapping symbolic blocks within the protograph. After this transformation, the lifted product matrix takes the following form:

$$\begin{aligned} A' &= [A_X \mid A_Z] \\ &= \left[\begin{array}{cc|cc} 0 & A_1^T \otimes \mathbb{I} & \mathbb{I} \otimes A_2 & 0 \\ A_1 \otimes \mathbb{I} & 0 & 0 & \mathbb{I} \otimes A_2^T \end{array}\right] \end{aligned} \tag{4}$$

Once again, this transformation yields mixed-type stabilizers that are naturally adapted to biased noise channels, particularly those dominated by dephasing errors. However, unlike conventional HGP codes built from arbitrary binary parity-check matrices, this construction uses structured protographs as seeds. By working at the symbolic level—before expansion into large binary matrices—the lifted hypergraph product (LHP) framework preserves algebraic structure that facilitates more efficient and scalable code design.

Notably, LHP codes derived from protographs often exhibit significantly better parameters than their binary HGP counterparts. They tend to achieve a higher rate (k/n), larger minimum distance, and improved decoding performance under belief propagation with ordered statistics decoding (BP+OSD). This advantage arises from the quasi-cyclic structure and controlled sparsity of protograph-based codes, which promote better girth, regularity, and convergence properties during iterative decoding [11].

III. METHODOLOGY

A. 4D Chain Complex

This work expands the two-dimensional HGP construction to four dimensions by introducing *four* classical seed matrices, $\delta_A, \delta_B, \delta_C, \delta_D$. See Fig. 2 for a visualization of the multi-dimensional chain complexes and the relationships induced by the boundary maps between adjacent layers.

Tensor-embedding these seeds produces the length-four chain complex. These matrices generate a length-four chain complex (details shown in Fig. 3). Each node in the diagram represents a tensor product of classical code components of the form

$$C_A^i \otimes C_B^j \otimes C_C^k \otimes C_D^l,$$

where each C_X^0 denotes the code space (primal) and C_X^1 the parity-check space (dual) of a classical seed code C_X, with $X \in \{A, B, C, D\}$. The superscripts $i, j, k, l \in \{0, 1\}$ indicate whether each factor is a codeword space or a check space. The total degree $i+j+k+l$ determines the vertical position of the node in the complex.

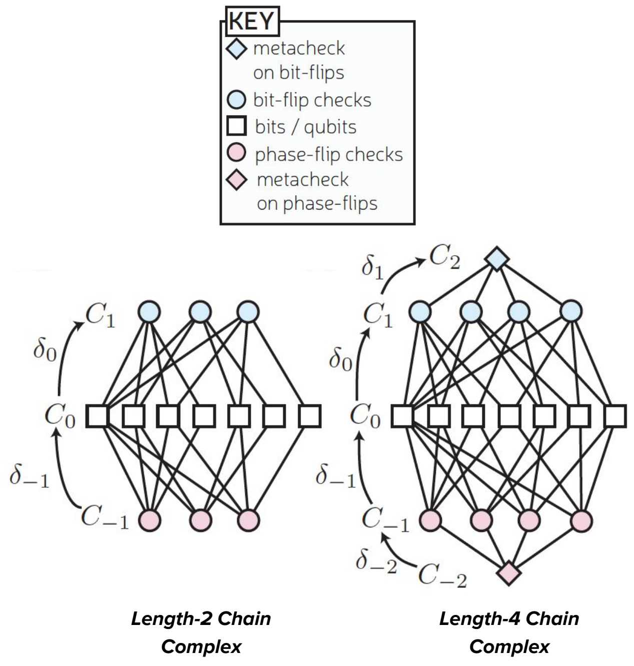

FIG. 2: Left: a standard CSS code corresponds to a length-2 complex, where qubit errors are detected by bit- and phase-flip checks ($\delta_{\pm 1}$). **Right**: a length-4 complex introduces metachecks ($\delta_{\pm 2}$) that validate the syndrome bits themselves [6].

Edges between nodes correspond to boundary maps—linear transformations defined via Kronecker products of identity matrices and seed code maps δ_X. These maps link adjacent layers and encode the syndrome relationships used for error detection.

Each space $\mathcal{C}_i$ corresponds to a distinct role in the code: $\mathcal{C}_0$ encodes the physical qubits, $\mathcal{C}_{\pm 1}$ represent the X- and Z-type stabilizers, and $\mathcal{C}_{\pm 2}$ define metachecks that validate the syndromes:

$$\mathcal{C}_{-2} \xrightarrow{\delta_{-2}} \mathcal{C}_{-1} \xrightarrow{\delta_{-1}} \mathcal{C}_0 \xrightarrow{\delta_0} \mathcal{C}_1 \xrightarrow{\delta_1} \mathcal{C}_2 \tag{5}$$

This layered structure introduces redundancy at the data and measurement levels. Each physical error activates multiple stabilizers, and each measurement error alters the syndrome in a way that triggers multiple metachecks. Thus, the 4D complex supports single-shot decoding: both data and measurement errors can be corrected after a single round of noisy syndrome extraction.

B. 4D Matrix Construction

To prepare for constructing the full 4D chain complex, first define the identity-expanded forms of the classical seed matrices:

$$\widetilde{\delta}_A := \delta_A \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \tag{6}$$

$$\widetilde{\delta}_B := \mathbb{1} \otimes \delta_B \otimes \mathbb{1} \otimes \mathbb{1} \tag{7}$$

$$\widetilde{\delta}_C := \mathbb{1} \otimes \mathbb{1} \otimes \delta_C \otimes \mathbb{1} \tag{8}$$

$$\widetilde{\delta}_D := \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes \delta_D \tag{9}$$

Each δ_X is the parity-check matrix of a classical seed code C_X, and its expansion via identity tensors ensures that every term acts on the full tensor product space $C_A \otimes C_B \otimes C_C \otimes C_D$. This uniform embedding is essential for composing boundary maps, where dimensional consistency across terms is required. These identity-augmented forms ensure that all boundary operators are well-defined and structurally aligned within the chain complex.

The resulting boundary maps naturally define the parity-check and metacheck matrices used for syndrome extraction and validation:

$$\delta_{-2}^T = M_Z = \begin{pmatrix} \widetilde{\delta}_A & \widetilde{\delta}_B & \widetilde{\delta}_C & \widetilde{\delta}_D \end{pmatrix} \tag{10}$$

$$\begin{aligned} \delta_{-1}^T = H_Z &= [H_{Z_1} \mid H_{Z_2}] \\ &= \left(\begin{array}{ccc:ccc} \widetilde{\delta}_B & \widetilde{\delta}_C & \widetilde{\delta}_D & 0 & 0 & 0 \\ \widetilde{\delta}_A & 0 & 0 & \widetilde{\delta}_C & \widetilde{\delta}_D & 0 \\ 0 & \widetilde{\delta}_A & 0 & \widetilde{\delta}_B & 0 & \widetilde{\delta}_D \\ 0 & 0 & \widetilde{\delta}_A & 0 & \widetilde{\delta}_B & \widetilde{\delta}_C \end{array}\right) \end{aligned} \tag{11}$$

$$\begin{aligned} \delta_0 = H_X &= [H_{X_1} \mid H_{X_2}] \\ &= \left(\begin{array}{ccc:ccc} \widetilde{\delta}_C & \widetilde{\delta}_B & 0 & \widetilde{\delta}_A & 0 & 0 \\ \widetilde{\delta}_D & 0 & \widetilde{\delta}_B & 0 & \widetilde{\delta}_A & 0 \\ 0 & \widetilde{\delta}_D & \widetilde{\delta}_C & 0 & 0 & \widetilde{\delta}_A \\ 0 & 0 & 0 & \widetilde{\delta}_D & \widetilde{\delta}_C & \widetilde{\delta}_B \end{array}\right) \end{aligned} \tag{12}$$

$$\delta_1 = M_X = \begin{pmatrix} \widetilde{\delta}_D & \widetilde{\delta}_C & \widetilde{\delta}_B & \widetilde{\delta}_A \end{pmatrix} \tag{13}$$

This construction ensures algebraic consistency ($\delta_i \circ \delta_{i-1} = 0$) and enables the detection of both qubit-level and measurement-level faults. Let the full parity-check matrix be organized as:

$$\begin{aligned} H = [H_X \mid H_Z] &= [H_{X_1} \mathrel{\vdots} H_{X_2} \mid H_{Z_1} \mathrel{\vdots} H_{Z_2}] \\ &= \left(\begin{array}{ccc:ccc|ccc:ccc} \widetilde{\delta}_C & \widetilde{\delta}_B & 0 & \widetilde{\delta}_A & 0 & 0 & \widetilde{\delta}_B & \widetilde{\delta}_C & \widetilde{\delta}_D & 0 & 0 & 0 \\ \widetilde{\delta}_D & 0 & \widetilde{\delta}_B & 0 & \widetilde{\delta}_A & 0 & \widetilde{\delta}_A & 0 & 0 & \widetilde{\delta}_C & \widetilde{\delta}_D & 0 \\ 0 & \widetilde{\delta}_D & \widetilde{\delta}_C & 0 & 0 & \widetilde{\delta}_A & 0 & \widetilde{\delta}_A & 0 & \widetilde{\delta}_B & 0 & \widetilde{\delta}_D \\ 0 & 0 & 0 & \widetilde{\delta}_D & \widetilde{\delta}_C & \widetilde{\delta}_B & 0 & 0 & \widetilde{\delta}_A & 0 & \widetilde{\delta}_B & \widetilde{\delta}_C \end{array}\right) \end{aligned} \tag{14}$$

As with the 2D HGP construction (Eq. 1, Eq. 2), the Hadamard rotation is implemented by swapping the rightmost halves of H_X and H_Z.

$$\begin{aligned} H' &= [H_{X_1} \mathrel{\vdots} H_{Z_2} \mid H_{Z_1} \mathrel{\vdots} H_{X_2}] \\ &= \left(\begin{array}{ccc:ccc|ccc:ccc} \widetilde{\delta}_C & \widetilde{\delta}_B & 0 & 0 & 0 & 0 & \widetilde{\delta}_B & \widetilde{\delta}_C & \widetilde{\delta}_D & \widetilde{\delta}_A & 0 & 0 \\ \widetilde{\delta}_D & 0 & \widetilde{\delta}_B & \widetilde{\delta}_C & \widetilde{\delta}_D & 0 & \widetilde{\delta}_A & 0 & 0 & 0 & \widetilde{\delta}_A & 0 \\ 0 & \widetilde{\delta}_D & \widetilde{\delta}_C & \widetilde{\delta}_B & 0 & \widetilde{\delta}_D & 0 & \widetilde{\delta}_A & 0 & 0 & 0 & \widetilde{\delta}_A \\ 0 & 0 & 0 & 0 & \widetilde{\delta}_B & \widetilde{\delta}_C & 0 & 0 & \widetilde{\delta}_A & \widetilde{\delta}_D & \widetilde{\delta}_C & \widetilde{\delta}_B \end{array}\right) \end{aligned} \tag{15}$$

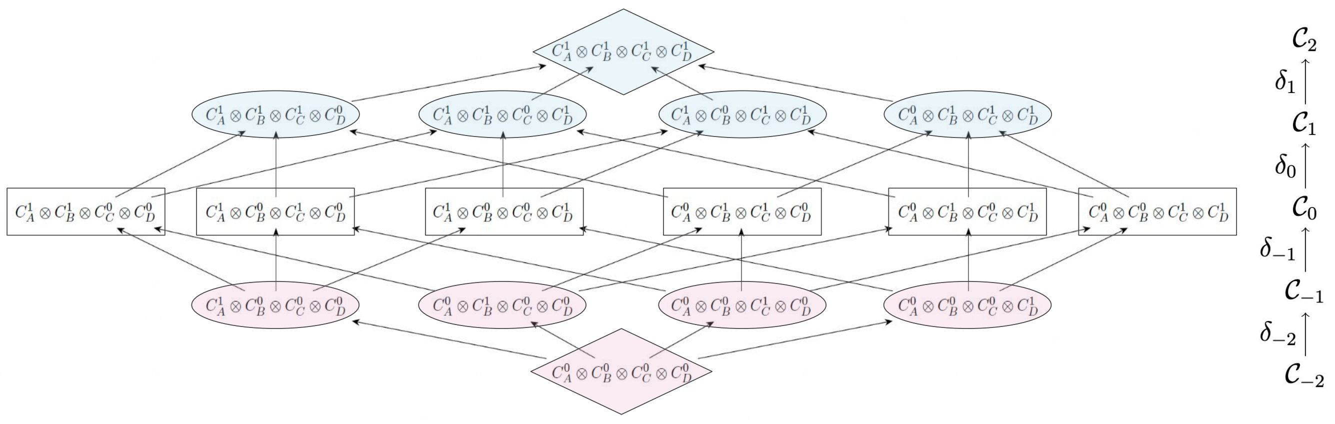
$C_A^1 \otimes C_B^1 \otimes C_C^1 \otimes C_D^1$
$C_A^1 \otimes C_B^1 \otimes C_C^1 \otimes C_D^0$
$C_A^1 \otimes C_B^1 \otimes C_C^0 \otimes C_D^1$
$C_A^1 \otimes C_B^0 \otimes C_C^1 \otimes C_D^1$
$C_A^0 \otimes C_B^1 \otimes C_C^1 \otimes C_D^1$
$C_A^1 \otimes C_B^1 \otimes C_C^0 \otimes C_D^0$
$C_A^1 \otimes C_B^0 \otimes C_C^1 \otimes C_D^0$
$C_A^1 \otimes C_B^0 \otimes C_C^0 \otimes C_D^1$
$C_A^0 \otimes C_B^1 \otimes C_C^1 \otimes C_D^0$
$C_A^0 \otimes C_B^1 \otimes C_C^0 \otimes C_D^1$
$C_A^0 \otimes C_B^0 \otimes C_C^1 \otimes C_D^1$
$C_A^1 \otimes C_B^0 \otimes C_C^0 \otimes C_D^0$
$C_A^0 \otimes C_B^1 \otimes C_C^0 \otimes C_D^0$
$C_A^0 \otimes C_B^0 \otimes C_C^1 \otimes C_D^0$
$C_A^0 \otimes C_B^0 \otimes C_C^0 \otimes C_D^1$
$C_A^0 \otimes C_B^0 \otimes C_C^0 \otimes C_D^0$
$\mathcal{C}_2$
δ_1
$\mathcal{C}_1$
δ_0
$\mathcal{C}_0$
δ_{-1}
$\mathcal{C}_{-1}$
δ_{-2}
$\mathcal{C}_{-2}$

FIG. 3: 4D lifted chain complex. Each node is a tensor product $C_A^i \otimes C_B^j \otimes C_C^k \otimes C_D^l$, where C^0 and C^1 denote code and check spaces. Nodes are grouped by total degree into five homological layers, with diamonds ($\mathcal{C}_{\pm 2}$) representing metachecks, ovals ($\mathcal{C}_{\pm 1}$) representing stabilizers, and rectangles ($\mathcal{C}_0$) representing qubits. Arrows denote boundary maps δ_i linking adjacent layers. Blue nodes correspond to X-type (bit-flip) operators, and pink nodes to Z-type (phase-flip) operators. The structure enables detection and correction of both data and measurement errors via single-shot decoding.

After this transformation, both H_X and H_Z exhibit centrosymmetric structure: the right half is a 180-degree rotation of the left. This symmetry aligns the code with the dephasing-biased noise model while preserving commutation between stabilizers [20].

C. Biased LHP Code Simulation

To implement the 4D-LHP code construction described above, I extended functionality from two core libraries: the `LDPC` package, which provides a symbolic protograph framework for constructing quasi-cyclic classical codes, and the `BPOSD` package, which defines CSS and general stabilizer code classes [21, 22]. This implementation inherits the protograph-handling functionality from the `LDPC` base class and the stabilizer structure and decoding interfaces from the `BPOSD` classes.

To incorporate lifting, my `LHP4D` class takes symbolic protographs as input matrices $(\delta_A, \delta_B, \delta_C, \delta_D)$ and performs intermediate operations directly in the symbolic domain before compiling the final parity-check matrices to binary form via the lift parameter L. I also adapted design patterns and techniques from the `Bias Tailored qLDPC` repository [23], which served as a practical reference for implementing LHP codes.

Bias tailoring is applied dynamically during simulation by modifying the error model rather than altering the code structure itself. The qubits are partitioned into two sectors corresponding to halves of the H_X and H_Z matrices. The first sector is not rotated, so it retains the standard Pauli error probabilities. In the second sector, the roles of X and Z are interchanged, meaning qubits that would typically receive Z errors are now subject to X errors (and vice versa).

This swap simulates the action of a Hadamard transformation on those qubits. The resulting channel asymmetry enables the simulation of bias-aware decoding without altering the code structure itself, preserving compatibility with standard CSS decoders.

D. Noise Model and Single-Shot Decoding

The SS decoding strategy used in this work consists of two decoding stages and a failure test, described in Algorithm 1. This approach draws on the protocol implemented in the `single_shot_3D_HGP` repository [24].

To evaluate the performance of bias-tailored LHP codes, I performed numerical simulations under a circuit-level noise model. Each physical qubit experiences Pauli noise with total probability p. Noise bias is introduced via parameters $\beta_X, \beta_Y, \beta_Z$, which define the relative rates of each Pauli error. The corresponding error probabilities are computed as:

$$p_X = \frac{p \cdot \beta_X}{\beta}, \quad p_Y = \frac{p \cdot \beta_Y}{\beta}, \quad p_Z = \frac{p \cdot \beta_Z}{\beta}, \tag{16}$$

$$\text{where } \beta = \beta_X + \beta_Y + \beta_Z.$$

In addition to data qubit errors, each stabilizer measurement may independently fail with probability q. A measurement error results in a flipped syndrome bit, potentially masking or mimicking a physical error. To decode under this joint data-and-syndrome noise model, I employ the two-step single-shot decoding strategy [5].

In the first stage, the observed syndrome—potentially corrupted by measurement noise—is interpreted as a codeword in the metacheck code and decoded using BP+OSD to locate and correct any measurement errors. The resulting corrected syndrome is then passed to a second BP+OSD decoder, which infers the corresponding physical error. If both decoding steps succeed, the combined recovery operator restores the system to the codespace.

After decoding, the residual error is computed as the sum of the true error and the inferred recovery operator:

$$\varepsilon_X = e_X + \hat{e}_X, \quad \varepsilon_Z = e_Z + \hat{e}_Z \tag{17}$$

The residual represents the net remaining error after correction. To determine whether the decoding has succeeded, the residual error is tested against all logical operators. Specifically, a logical failure is declared if the residual error anticommutes with any logical operator:

$$(L_Z \cdot \varepsilon_X) \vee (L_X \cdot \varepsilon_Z) \neq 0, \tag{18}$$

as this failure indicates that the error has induced a nontrivial logical operation on the encoded state. Otherwise, decoding is deemed successful, ensuring that the recovery operator corrects the syndrome and preserves the encoded logical information.

E. Protograph Seed Selection and Code Parameters

To obtain high-quality classical seeds for the 4D-LHP construction, I performed a *progressive edge growth* (PEG) search. The PEG heuristic [25, 26] constructs Tanner graphs iteratively, selecting each edge to maximize a local structural metric—typically the girth—thereby avoiding short cycles that degrade decoding performance. This process yields LDPC matrices with large local girth, favorable thresholds, and improved error resilience.

From several candidates, I selected a small-scale protograph with high local girth and no weight-two cycles:

$$\begin{bmatrix} \lambda(2) & \lambda() & \lambda(0) & \lambda(0) & \lambda() & \lambda(2) \\ \lambda() & \lambda(0) & \lambda(2) & \lambda() & \lambda(2) & \lambda(0) \\ \lambda(0) & \lambda(1) & \lambda() & \lambda(1) & \lambda(0) & \lambda(2) \\ \lambda(0) & \lambda(1) & \lambda(1) & \lambda(1) & \lambda(1) & \lambda() \end{bmatrix} \tag{19}$$

Lifting with $L = 3$ yields a quantum code with parameters $[[384, 48, 6]]$ (rate $K/N = 0.125$). This code demonstrates equal distance and significantly better rate than the canonical hypergraph product code $[[400, 16, 6]]$ (rate 0.04), which is often used in numerical benchmarks of quantum LDPC decoding [10]. The generated code is also competitive with the best known asymptotically good constructions: Panteleev and Kalachev's lifted product codes achieve constant rate $R \gtrsim 0.1$ in the large-N limit [12], while recent twisted constructions have reached rates near $R \simeq 0.20$ [27].

Against this backdrop, the finite-length rate of 0.125 demonstrates that the 4D–LHP architecture can match or exceed the efficiency of much larger asymptotic codes, while retaining practical properties such as bounded stabilizer weight and support for single-shot decoding.

IV. RESULTS

To quantify performance, the simulation measures the *word error rate* (WER), defined as the probability that the decoder fails to return the correct logical state. Monte Carlo trials were conducted over a range of physical error rates $p \in [0.01, 0.08]$, measurement error rates $q \in [0.0, 0.04]$, and bias ratios $\eta = p_Z/p_X$ ranging from 10^{-3} to 10^3.

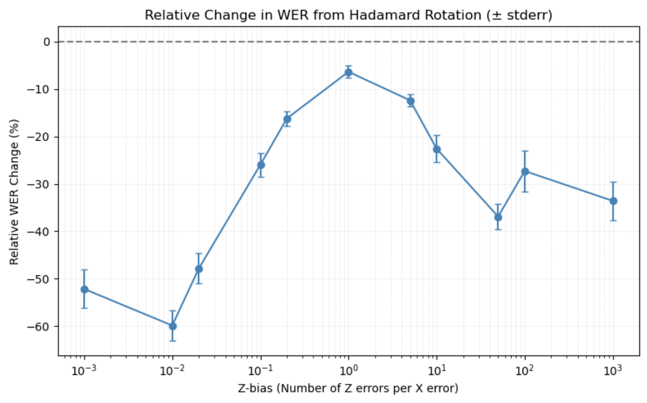

FIG. 4: Benefit of Hadamard bias-tailoring. Relative change in WER (negative values indicate improvement) as a function of the $Z{:}X$ bias ratio $\eta = p_Z/p_X$. Error bars show $\pm$ 1 s.e.m. Bias tailoring yields greatest gains (up to 60 %) in moderate bias $10^{-3} \leq \eta \leq 10^{-1}$.

Figure 4 reports the *relative* change in WER induced by bias tailoring, expressed as a percentage difference between the untailored and Hadamard-rotated code. Across the entire range of tested error rates and noise biases, the Hadamard rotation consistently improved performance—lowering the WER in every tested configuration.

The largest gains were observed in the moderately biased regime ($10^{-2} \leq \eta \leq 10^{-1}$), where WER dropped by up to 60%. Even in the near-unbiased case ($\eta \approx 1$), the tailored code outperformed its untwisted counterpart by approximately 10%, indicating that the rotation does not degrade performance even when bias is weak. For extremely high bias ($\eta \gg 10^2$), the benefit remains substantial (up to 40%), though it tapers slightly—likely due to error patterns concentrating in highly confined sectors where decoder performance saturates.

These results confirm that bias tailoring via a Hadamard rotation consistently enhances the code's resilience to physical noise by increasing the effective distance of dominant-error logical operators. Notably, this improvement holds regardless of the underlying error rates p and q, making the technique broadly applicable across realistic NISQ regimes.

Figure 5a plots the WER of the 4D-LHP code as a function of the physical error rate p, stratified by measurement error probability q. In the ideal-measurement limit ($q = 0$), the decoder maintains high reliability: the WER remains below 10^{-2} for physical error rates up to $p \simeq 0.06$, and remains below 0.1 even at $p = 0.08$. This performance baseline reflects the effectiveness of the BP+OSD decoder under purely data-driven noise.

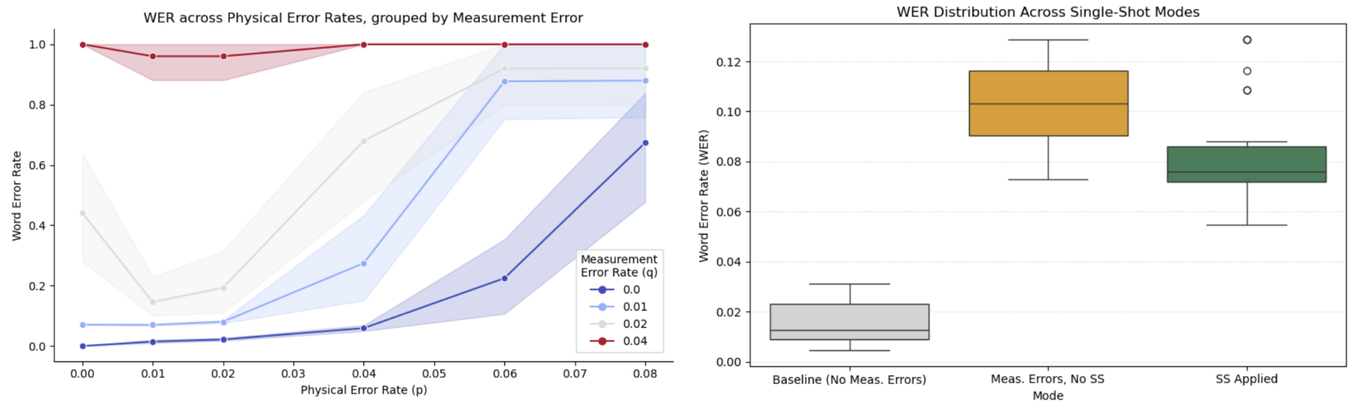

FIG. 5: Effect of measurement errors and single-shot (SS) decoding on word-error rate (WER). **Left**: WER vs. physical error rate p for several measurement-error rates q (no SS decoding). Even modest $q \geq 0.02$ causes an order-of-magnitude rise in logical failures. **Right**: WER distributions at a representative point ($p = 0.04$). Baseline (perfect measurements), noisy measurements without SS, and noisy measurements with SS are compared. A single SS round lowers the median WER from 1.03×10^{-1} to 7.4×10^{-2} (-28 %), recovering roughly one-third of the performance lost to measurement noise.

However, once measurement noise is introduced, performance degrades rapidly. For $q = 0.01$, the WER begins to rise sharply beyond $p = 0.04$, and for $q = 0.02$, logical errors become frequent across the full range of tested p values. At $q = 0.04$, decoding fails almost deterministically, with WER saturating near 1.0 for all values of p. These results highlight the vulnerability of quantum LDPC codes to even modest levels of measurement noise if left uncorrected.

The increasing spread between the curves as p increases underscores the compounding effect of data and measurement errors. In the absence of metachecks, syndrome faults propagate directly into decoding failures, exacerbating the logical error rate. These findings underscore the necessity of explicit mechanisms for correcting measurement errors—such as single-shot decoding—when operating under realistic, noisy readout conditions.

Notably, the lowest measurement noise configuration ($q = 0.0$) defines an upper bound on achievable performance for the given code and decoder, serving as a target baseline for evaluating the effectiveness of single-shot recovery in the next section.

Figure 5b compares the distribution of word error rates (WER) across three decoding modes: (i) a baseline configuration with noiseless measurement , (ii) decoding under realistic measurement noise without single-shot (SS) correction, and (iii) decoding under measurement noise with the SS layer applied.

The baseline mode establishes the optimal performance expected in the absence of measurement faults, with a median WER below 0.02 and tight variance across trials. Introducing measurement errors without correction degrades performance sharply: the median WER rises to $\sim$ 0.11, and the interquartile range (IQR) expands significantly, indicating unstable and unreliable decoding. This degradation underscores the critical role that accurate syndrome extraction plays in quantum LDPC decoding.

When the single-shot layer is enabled, the decoder regains a substantial portion of lost performance. The median WER drops to $\sim$ 0.08, recovering roughly one third of the performance gap caused by measurement noise.

Together, these results validate the efficacy of single-shot decoding in practical, noisy scenarios. Although the SS layer does not fully restore ideal performance, it meaningfully reduces the logical failure rate and narrows performance variation—offering a practical pathway toward fault-tolerant operation in near-term quantum architectures.

Evaluating metacheck performance is essential for assessing the single-shot capabilities of the current code and for guiding the design of optimized 4D-LHP codes. Figure 6 analyzes the effectiveness of the metacheck layer by reporting the distribution of detection and correction rates across Monte Carlo trial.

Detection rates are tightly concentrated near 1.0 for both X- and Z-type measurement faults. Most trials achieve fault identification accuracy above 99.8%, confirming that the metasyndrome reliably signals the presence of measurement errors. The sharp rightward skew in the histogram indicates that near-perfect detection is not an outlier but rather the dominant regime. This highlights the structural redundancy of the 4D chain complex: each measurement error affects multiple overlapping metachecks, boosting the likelihood of detection.

However, detection does not guarantee correction. The metacheck correction rate—the proportion of measurement faults successfully corrected—is more broadly distributed. While many trials achieve correction rates in the 5–10% range, a substantial fraction see near-zero correction. This bimodal distribution indicates that correction success depends heavily on the specific fault pattern and decoder convergence. Nonetheless, even partial correction has meaningful impact: as shown in Fig. 5b, correcting just a small subset of faulty syndromes is sufficient to materially reduce the word error rate.

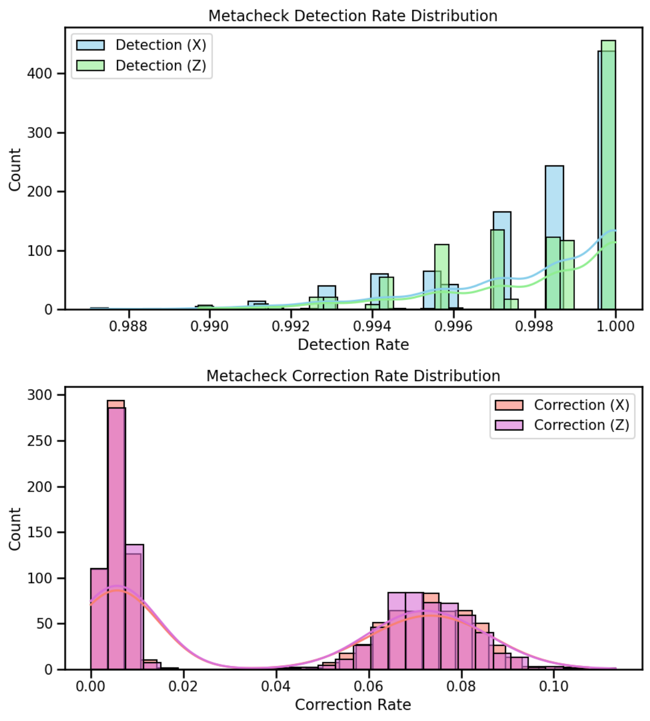

FIG. 6: Top panel: Detection rates for X-type (blue) and Z-type (green) measurement faults concentrate tightly near 1.0, confirming near-perfect fault identification. **Bottom panel: Correction rates** are more dispersed, peaking around 7-8%, with a notable spike near zero for both error types. Although far fewer faults are corrected than detected, this limited success is sufficient to significantly reduce WER.

Together, these results validate the design of the metacheck layer. It provides nearly full fault *visibility* (detection), and enough *actionability* (correction) to recover significant performance. Improving the correction success rate—e.g., through tailored seed selection or metacheck-specific decoding heuristics—could further amplify the benefit of single-shot decoding in high-noise or bias-skewed regimes.

Across the simulated parameter space, the 4D-LHP code exhibits:

1. **Strong bias responsiveness:** a tailored Hadamard rotation cuts logical errors by up to 60% in regimes of moderate asymmetry.
2. **Robustness to measurement noise:** single-shot decoding restores a substantial portion of baseline performance at realistic measurement-error rates.
3. **High fault visibility:** metachecks detect almost all measurement faults, and their partial correction materially improves overall reliability.

These results demonstrate that combining bias tailoring with single-shot decoding in a 4D lifted-product architecture yields significant practical advantages under circuit-level noise.

V. DISCUSSION

The numerical results presented above illustrate how a single architectural framework—a four–dimensional lifted homological product (4D LHP) code equipped with Hadamard bias-tailoring and single-shot (SS) decoding—addresses *two* practical hurdles that currently limit near-term quantum devices: biased physical noise and faulty syndrome extraction.

Below I first contextualize the empirical gains, then outline concrete avenues for boosting metacheck effectiveness, and finally chart broader directions for next-generation code design and decoding.

A. Interpreting the Numerical Gains

Additive benefits. Bias tailoring and SS decoding have, until now, advanced along mostly separate lines of inquiry. The present work shows they can be combined *constructively.* The tailored Hadamard rotation reduces the word error rate (WER) by as much as 60% in the moderately biased regime ($\eta \sim 10^{-2}$; Fig. 4)

In parallel, the SS layer mitigates roughly one third of the logical errors introduced by measurement noise (Fig. 5a). When both techniques are applied, they reduce errors independently: bias tailoring leverages the Hadamard rotation to align with the noise bias, while SS decoding exploits structural redundancy in the chain complex.

Metacheck visibility vs. actionability. The 4D complex furnishes near-perfect *visibility* of measurement faults, as more than 99.8% of faulty syndromes are flagged. However, only 5–10% of those detected faults are successfully corrected (Fig. 6). This apparent discrepancy is characteristic of current LDPC decoders, which are highly sensitive to short cycles and trapping sets [28]. Even so, partial correction notably diminishes the WER relative to an architecture without metachecks. Hence, the bottleneck is not detection but rather *actionability*—a decoder-centric limitation that future work can address.

B. Improving Metacheck Correction Efficiency

Seed optimisation with confinement heuristics. Detection without correction typically arises when the measurement-error pattern spans a “loose” metacheck boundary that the BP+OSD decoder fails to close. Classical seeds with stronger *confinement* [5] yield tighter, low-weight boundaries and thereby increase the likelihood that noisy syndromes remain correctable under BP+OSD decoding. Extending progressive-edge growth (PEG) searches to include confinement and trapping-set metrics [25] could therefore increase the metacheck-correction fraction beyond today’s 10 %.

Decoders that exploit metasyndrome structure. Standard BP treats each check on an equal footing; it is oblivious to the fact that metachecks represent *syndrome* correlations rather than qubit correlations. Tailored message-passing rules [29] or neural decoders trained on the joint space of data- and meta-syndromes [30] may extract more information from the same parity budget, converting visibility into actionability with modest classical overhead.

Adaptive two-level scheduling. Because metachecks are cheap to measure (they act on classical bits), one can iterate a light-weight meta-decoding step *in situ*: if a subset of metachecks remains unsatisfied after the first SS pass, only those checks need be remeasured, dramatically reducing latency while probabilistically lifting correction rates toward the detection limit [4].

C. Future Directions

Systematic seed search via PEG. While this work employed PEG-optimized protographs for the seed matrices $\{\delta_A, \delta_B, \delta_C, \delta_D\}$, the search was heuristic and manually tuned for girth. A more systematic exploration of the seed space—guided by structural metrics tailored to 4D-LHP codes—could uncover configurations with significantly higher metacheck-correction efficiency.

In particular, extending PEG to explicitly prioritize local confinement (i.e., limiting the spatial spread of error patterns through the Tanner graph) and avoiding known trapping sets may yield seeds that not only support good code distance but also exhibit stronger metasyndrome resolvability. This direction could benefit from combining classical PEG techniques [25] with emerging graph-theoretic or machine-learning-guided search heuristics that explore the vast combinatorial space of symbolic protographs.

Sustained threshold under repeated decoding. This work analyzes code performance in a single-cycle noise model, wherein a round of noise is followed by a single-shot decode. While this is sufficient to validate metacheck functionality, it does not yet establish a true fault-tolerance threshold. In a real fault-tolerant architecture, errors accumulate over many cycles of syndrome extraction, decoding, and correction.

To assess long-term stability, future simulations must extend to sustained-depth regimes—repeatedly applying circuit-level noise, single-shot decoding, and fault propagation tracking across multiple rounds. Following the methodology of [5], one can determine whether logical error rates saturate (threshold behavior) or accumulate unboundedly over time. A favorable sustained threshold would demonstrate that 4D-LHP codes are not only effective in isolated rounds but also suitable for long-duration memory or computation.

One-stage decoders for metacheck integration. The current decoding approach separates syndrome cleaning and error correction into two stages: first correcting measurement noise using metachecks, then correcting data errors using the cleaned syndrome. While robust, this method incurs classical overhead, potential latency, and dependency between decoders. Recent studies [31] suggest that a single-stage belief propagation (BP) decoder—capable of jointly interpreting data and measurement errors—can outperform two-stage decoding, especially in sparse codes with correlated error structures.

Adapting such a decoder to the 4D-LHP setting would require integrating metachecks directly into the Tanner graph and modifying message-passing rules to reflect the layered structure of the chain complex. If successful, this could significantly reduce decoding latency and memory requirements, making 4D-LHP codes more viable for real-time decoding in hardware-constrained settings.

VI. CONCLUSION

This work introduces a four-dimensional lifted homological-product (4D-LHP) code that tackles two major challenges in quantum error correction: biased physical noise and faulty stabilizer measurements. By combining a tailored Hadamard rotation with a chain-complex structure that supports metachecks, the code achieves both bias alignment and single-shot (SS) decoding.

Together, these mechanisms reduce the word error rate (WER) by up to an order of magnitude compared to untailored, non-SS baselines—cutting logical failures by 60% under bias and recovering a third of the performance lost to measurement noise. Importantly, the approach is scalable: the same design principles can be applied to build high-rate, low-overhead LDPC codes suitable for near-term quantum memories.

Looking forward, the integration of bias tailoring and SS decoding into logical gate protocols—such as lattice surgery or twist-based operations—could further reduce overhead in biased-noise processors [32, 33]. More broadly, this work illustrates how algebraic code constructions, noise-adapted Clifford transformations, and modern decoding techniques can be combined to meet the practical demands of early fault-tolerant quantum hardware.

[1] J. Preskill, Quantum **2**, 79 (2018).
[2] P. W. Shor, (1996), quant-ph/9605011 [quant-ph].
[3] H. Bombín, Phys. Rev. X. **5** (2015).
[4] H. Bombín, Phys. Rev. X. **6** (2016).
[5] A. O. Quintavalle, M. Vasmer, J. Roffe, and E. T. Campbell, PRX Quantum **2** (2021).
[6] E. T. Campbell, Quantum Sci. Technol. **4**, 025006 (2019).
[7] D. Ruiz, J. Guillaud, A. Leverrier, M. Mirrahimi, and C. Vuillot, Nat. Commun. **16**, 1040 (2025).
[8] S. Puri, L. St-Jean, J. A. Gross, A. Grimm, N. E. Frat-

tini, P. S. Iyer, A. Krishna, S. Touzard, L. Jiang, A. Blais, S. T. Flammia, and S. M. Girvin, Sci. Adv. **6**, eaay5901 (2020).
[9] J. P. Bonilla Ataides, D. K. Tuckett, S. D. Bartlett, S. T. Flammia, and B. J. Brown, Nat. Commun. **12**, 2172 (2021).
[10] J.-P. Tillich and G. Zemor, IEEE Trans. Inf. Theory **60**, 1193 (2014).
[11] J. Roffe, L. Z. Cohen, A. O. Quintavalle, D. Chandra, and E. T. Campbell, Quantum **7**, 1005 (2023).
[12] P. Panteleev and G. Kalachev, in *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing* (ACM, New York, NY, USA, 2022).
[13] C. Weibel, An Introduction to Homological Algebra , 1–29 (1994).
[14] S. Bravyi and M. B. Hastings, arXiv [quant-ph] (2013).
[15] P. Panteleev and G. Kalachev, Quantum **5**, 585 (2021).
[16] Q. Xu, N. Mannucci, A. Seif, A. Kubica, S. T. Flammia, and L. Jiang, Phys. Rev. Res. **5** (2023).
[17] J. A. Campos and K. R. Brown, (2024), 2412.03808 [quant-ph].
[18] E. Huang, A. Pesah, C. T. Chubb, M. Vasmer, and A. Dua, PRX quantum **4** (2023).
[19] A. G. Fowler, A. M. Stephens, and P. Groszkowski, Phys. Rev. A **80** (2009).
[20] M. Akarsu, Kuramsal Eğit. **15**, 64 (2022).
[21] J. Roffe and collaborators, Ldpc: Symbolic construction of classical and quantum ldpc codes, https://github.com/quantumgizmos/ldpc (2023), accessed May 9, 2025.
[22] J. Roffe and collaborators, Bposd: Belief propagation and ordered statistics decoding for quantum ldpc codes, https://github.com/quantumgizmos/bp_osd (2023), accessed May 9, 2025.
[23] J. e. a. Roffe, Bias-tailored quantum ldpc codes: Simulation framework, https://github.com/quantumgizmos/bias_tailored_qldpc (2023), accessed May 9, 2025.
[24] M. Vasmer, single_shot_3d_hgp, https://github.com/MikeVasmer/single_shot_3D_HGP/tree/bdfb437b2abcfa514997f26be97a711b878448cb (2023), accessed: 2025-05-14.
[25] X. He, L. Zhou, and J. Du, IEEE Trans. Commun. **66**, 1845 (2018).
[26] H. Chen and Z. Cao, in *Proc. IEEE WCNC* (Hong Kong, China, 2007).
[27] Z. Yi, Z. Liang, J. Chen, Z. Wang, and X. Wang, (2024), 2402.09648 [quant-ph].
[28] T. Richardson and R. Urbanke, IEEE Transactions on Information Theory **47**, 619 (2001).
[29] A. Hutter, J. R. Wootton, and L. P. Pryadko, Quantum Science and Technology **5**, 034008 (2020).
[30] S. Varsamopoulos, B. Criger, and K. Bertels, in *2019 IEEE International Conference on Quantum Computing and Engineering (QCE)* (IEEE, 2019) pp. 267–277.
[31] O. Higgott and N. P. Breuckmann, PRX quantum **4** (2023).
[32] B. J. Brown and K. R. Brown, Phys. Rev. Appl. **17**, 044082 (2022).
[33] S. E. e. a. Roberts, Quantum Sci. Technol. **7**, 045012 (2022).

Appendix A: Single-Shot Decoding Algorithm

Algorithm 1 gives the Monte-Carlo procedure used in Sec. IV.

Algorithm 1 Single-Shot CSS Simulation with BP + OSD

```
Input: Parity–check matrices $H_X, H_Z$; metachecks $M_X, M_Z$; physical error rate $p$; bias vector $(\beta_X, \beta_Y, \beta_Z)$; measurement-error rate $q$; decoding parameters (BP method, OSD order, ...); rounds $T$
Output: Logical error rate (LER), word error rate (WER).
1 Pre-processing
    1. Build CSS code object from $(H_X, H_Z)$
    2. Compute channel probabilities $\Pr(X) = p \cdot \beta_X/\Sigma$, $\Pr(Y) = p \cdot \beta_Y/\Sigma$, $\Pr(Z) = p \cdot \beta_Z/\Sigma$ where $\Sigma = \beta_X + \beta_Y + \beta_Z$.
    3. Instantiate four BP+OSD decoders:
       $\mathsf{BP}_X$ on $H_Z$, $\mathsf{BP}_Z$ on $H_X$ (data errors)
       $\mathsf{BP}_{\mathrm{MX}}$ on $M_X$, $\mathsf{BP}_{\mathrm{MZ}}$ on $M_Z$ (meas. errors).
Main Monte-Carlo Loop $t = 1, \ldots, T$
    1. SamplePhysicalError
       Independently flip each qubit as $I, X, Y, Z$ according to channel probs; store binary vectors $e_X, e_Z$.
    2. SyndromeGeneration
       $s_X^{\mathrm{ideal}} \leftarrow H_X e_X \bmod 2$
       Add measurement noise: $s_X^{\mathrm{noisy}} \leftarrow s_X^{\mathrm{ideal}} + \mathrm{Bernoulli}(q) \pmod 2$ (likewise for $Z$).
    3. MetacheckDecode
       Compute meta-syndromes $m_X \leftarrow M_X s_X^{\mathrm{noisy}}$, $m_Z \leftarrow M_Z s_Z^{\mathrm{noisy}}$
       Decode with $\mathsf{BP}_{\mathrm{MX}}, \mathsf{BP}_{\mathrm{MZ}}$ to obtain meas. error estimates $\hat{\mu}_X, \hat{\mu}_Z$.
    4. SyndromeCorrection
       $\tilde{s}_X \leftarrow s_X^{\mathrm{noisy}} + \hat{\mu}_X \pmod 2$, $\tilde{s}_Z \leftarrow s_Z^{\mathrm{noisy}} + \hat{\mu}_Z \pmod 2$.
    5. DataDecode (a) Decode $Z$-errors first: $\hat{e}_Z \leftarrow \mathsf{BP}_Z(\tilde{s}_Z)$ (b) Optional channel update: adjust $\mathsf{BP}_X$ priors using $\hat{e}_Z$ (c) Decode $X$-errors: $\hat{e}_X \leftarrow \mathsf{BP}_X(\tilde{s}_X)$
    6. FailureTest
       Residual error $\varepsilon_X = e_X + \hat{e}_X$, $\varepsilon_Z = e_Z + \hat{e}_Z \pmod 2$.
       If $(L_Z \varepsilon_X) \vee (L_X \varepsilon_Z) \neq 0$ for any logical operator $L_{X/Z}$, count a logical failure; otherwise success.
    7. Update running tallies: counts of BP convergence, OSD-0 success, OSD-W success, min. logical weight.
```