


# Planar: A software for exact decoding quantum error correction codes with planar structure

Dongyang Feng(冯东阳), Hanyan Cao(曹涵彦), and Pan Zhang(张潘)

**Citation:**Chin. Phys. B, 2025, 34 (**5**): 050311. DOI: 10.1088/1674-1056/adcb26
Journal homepage: http://cpb.iphy.ac.cn; http://iopscience.iop.org/cpb

**What follows is a list of articles you may be interested in**

---

## Distributed quantum circuit partitioning and optimization based on combined spectral clustering and search tree strategies

Zilu Chen(陈子禄), Zhijin Guan(管致锦), Shuxian Zhao(赵书娴), and Xueyun Cheng(程学云)
Chin. Phys. B, 2025, 34 (**5**): 050305. DOI: 10.1088/1674-1056/adbada

## All-microwave CZ gate based on fixed-frequency driven coupler

Wanpeng Gao(高万鹏), Xiaoliang He(何潇梁), Zhengqi Niu(牛铮琦), Daqiang Bao(包大强), Kuang Liu(刘匡), Junfeng Chen(陈俊锋), Zhen Wang(王镇), and Z. R. Lin(林志荣)
Chin. Phys. B, 2025, 34 (**4**): 040304. DOI: 10.1088/1674-1056/adb68a

## Global receptive field transformer decoder method on quantum surface code data and syndrome error correction

Ao-Qing Li(李熬庆), Ce-Wen Tian(田策文), Xiao-Xuan Xu(徐晓璇), Hong-Yang Ma(马鸿洋), and Jun-Qing Liang(梁俊卿)
Chin. Phys. B, 2025, 34 (**3**): 030306. DOI: 10.1088/1674-1056/adab63

## Quantum decoder design for subsystem surface code based on multi-head graph attention and edge weighting

Nai-Hua Ji(纪乃华), Hui-Qian Sun(孙汇倩), Bo Xiao(肖博), Ping-Li Song(宋平俐), and Hong-Yang Ma(马鸿洋)
Chin. Phys. B, 2025, 34 (**2**): 020309. DOI: 10.1088/1674-1056/ada1c7

## Correcting on-chip distortion of control pulses with silicon spin qubits

Ming Ni(倪铭), Rong-Long Ma(马荣龙), Zhen-Zhen Kong(孔真真), Ning Chu(楚凝), Wei-Zhu Liao(廖伟筑), Sheng-Kai Zhu(祝圣凯), Chu Wang(王儲), Gang Luo(罗刚), Di Liu(刘頓), Gang Cao(曹刚), Gui-Lei Wang(王桂磊), Hai-Ou Li(李海欧), and Guo-Ping Guo(郭国平)
Chin. Phys. B, 2025, 34 (**1**): 010308. DOI: 10.1088/1674-1056/ad8db1

---

**SPECIAL TOPIC — Computational programs in complex systems**

# PLANAR: A software for exact decoding quantum error correction codes with planar structure

Dongyang Feng(冯东阳)[1,2,3,†], Hanyan Cao(曹涵彦)[2,3,†], and Pan Zhang(张潘)[1,2,‡]

[1] *School of Fundamental Physics and Mathematical Sciences, Hangzhou Institute for Advanced Study, UCAS, Hangzhou* 310024, *China*
[2] *CAS Key Laboratory for Theoretical Physics, Institute of Theoretical Physics, Chinese Academy of Sciences, Beijing* 100190, *China*
[3] *School of Physical Sciences, University of Chinese Academy of Sciences, Beijing* 100049, *China*

(Received 14 March 2025; revised manuscript received 9 April 2025; accepted manuscript online 10 April 2025)

Quantum error correction is essential for realizing fault-tolerant quantum computing, where both the efficiency and accuracy of the decoding algorithms play critical roles. In this work, we introduce the implementation of the PLANAR algorithm, a software framework designed for fast and exact decoding of quantum codes with a planar structure. The algorithm first converts the optimal decoding of quantum codes into a partition function computation problem of an Ising spin glass model. Then it utilizes the exact Kac–Ward formula to solve it. In this way, PLANAR offers the exact maximum likelihood decoding in polynomial complexity for quantum codes with a planar structure, including the surface code with independent code-capacity noise and the quantum repetition code with circuit-level noise. Unlike traditional minimum-weight decoders such as minimum-weight perfect matching (MWPM), PLANAR achieves theoretically optimal performance while maintaining polynomial-time efficiency. In addition, to demonstrate its capabilities, we exemplify the implementation using the rotated surface code, a commonly used quantum error correction code with a planar structure, and show that PLANAR achieves a threshold of approximately $p_{\rm uc} \approx 0.109$ under the depolarizing error model, with a time complexity scaling of $O(N^{0.69})$, where $N$ is the number of spins in the Ising model.

**Keywords:** quantum computing, quantum error correction, planar Ising model

**PACS:** 03.67.Pp, 02.50.–r, 05.10.–a, 05.50.+q

**DOI:** 10.1088/1674-1056/adcb26 **CSTR:** 32038.14.CPB.adcb26

## 1. Introduction

The realization of fault-tolerant quantum computing (FTQC) marks a pivotal milestone in the advancement of quantum technologies, and significant progress has been achieved in recent years.[1–9] To achieve FTQC, the development of quantum error correction codes (QECCs) with robust properties, as well as fast and accurate decoders, is of paramount importance.[10–15] A quantum code, denoted as $[n,k,d]$, encodes $k$ logical qubits with minimum distance $d$ using $n$ physical qubits. The encoded states are fixed points (i.e., as eigenvectors with eigenvalue 1) by operators generated by $m$ generators, which are known as stabilizers $\mathcal{S}$.[16] Measuring the ancilla qubits corresponding to the stabilizers yields a result of 1 without affecting the encoded states. When a Pauli error $O \in \{I,X,Y,Z\}^n$ occurs and anti-commutes with the stabilizers, the stabilizer measurements produce $-1$, resulting in syndromes $\gamma \in \{1,-1\}^m$. This tells us that there were errors and we need to identify the error and then correct it to recover the encoded states. Identifying the possible errors given syndrome $\gamma$ is the task of decoding.

There are two kinds of decoding approaches, the first one finds the most probable error with

$$\hat{O} = \arg\max_{O} P(O|\gamma), \tag{1}$$

where $P(O|\gamma)$ is the conditional probability of finding the error operator $O$ given syndrome $\gamma$ in the error model. This kind is known as minimum weight decoding (MWD),[17–19] where the weight can be understood as the negative log probability. The most commonly used MWD algorithm is the minimum-weight perfect matching (MWPM) algorithm.[17] In the case of quantum Calderbank–Shor–Steane (CSS) code, where $X$ and $Z$ sectors can be decoded independently, the MWD of these sectors can be reformulated as matching problems. Efficient implementations of MWPM, such as Blossom[20] and Union-Find,[21] can rapidly resolve the MWD problem for specific codes. The advantage of the minimum-weight decoding is the efficiency, as it turns the problem into an optimization task that can be effectively addressed across various codes and noise models.

However, in quantum codes, a degeneracy issue arises, where multiple errors produce identical effects on the encoded logical space. The effect is presented by the logical sectors $L$ characterized using the commutation relation between the er-

[†] These authors contributed equally to this work.
[‡] Corresponding author. E-mail: panzhang@itp.ac.cn

© 2025 Chinese Physical Society and IOP Publishing Ltd. All rights, including for text and data mining, AI training, and similar technologies, are reserved.

ror operator $O$ and logical operators $I, L_X, L_Y$, and $L_Z$. In other words, the optimal decoding problem is to estimate the logical sector $L$ given the syndrome $\gamma$. This is actually the principle of the maximum likelihood decoding (MLD),[22,23] which maximizes the conditional probability of the logical sector given syndrome

$$\hat{L} = \arg\max_L P(L|\gamma) = \arg\max_L \sum_{S\in\mathcal{S}} P[L,S|\gamma], \tag{2}$$

where $L$ ranges over all possible logical sectors and the summation extends over all elements of the stabilizer group. Since MLD accounts for the properties of the stabilizer group and incorporates quantum degeneracy, its performance is generally superior to MWD. However, MLD is more computationally demanding, as it requires the evaluation of $4^k$ coset probabilities, each of which belongs to the #P-hard complexity class and lacks a polynomial-time exact solution.

For specific quantum codes with lattice-like structures, the maximum likelihood decoding can be effectively performed using tensor network algorithms, especially the boundary matrix product state (bMPS) algorithms.[24] The performance of such algorithms can closely approach the theoretical optimum as they rigorously adhere to the MLD decoding formula. Additionally, the approximation error in these algorithms can be accurately controlled using the canonical form of MPS. However, the primary limitation of this method is its scaling, which is $O(n\chi^3)$, where $\chi$ is the bond dimension of MPS. As the bond dimension typically increases with code size, the tensor network decoding method becomes less efficient for larger codes. Recently, an exact maximum likelihood encoding approach[25] has been established for the repetition code with circuit-level noise. We refer to it as PLANAR. The approach has a polynomial time complexity to the number of qubits. Regarding decoding accuracy, PLANAR achieves optimal performance by strictly adhering to the MLD criterion. For decoding speed, it scales polynomially with the number of stabilizers. This scaling is comparable to the performance of the MWPM algorithm and significantly faster than MLD algorithms based on tensor network contraction. Additionally, it outperforms the minimum weight perfect matching algorithm on various noise models in repetition codes.[1] In this work, we extend the study of the PLANAR to a broad class of quantum error-correcting codes with planar structures, using an example of rotated surface code with independent code-capacity noise and explain the details of the implementation.[26]

In the following sections, we first present the statistical mechanical mapping of quantum error correction code problems and the principles of PLANAR decoding. This includes examining the planar Ising model mapping for specific QECCs and introducing a polynomial-time solver for the planar Ising model partition function. Subsequently, we detail the mapping procedure using Python code for rotated surface codes of arbitrary distances. We then provide numerical results to demonstrate the performance of PLANAR. Finally, we conclude with discussions and outline potential future research directions.

## 2. Principles of the PLANAR decoder

### 2.1. The maximum likelihood decoding

PLANAR belongs to the maximum likelihood decoding, which aims to identify the most likely logical sector by computing the probability $P(L)$ for each logical sector $L$. Here, $P(L)$ represents the total probability of all error configurations that are consistent with the syndrome measurements and have the same commutation relation with the logical operator. This is achieved by considering all possible error configurations through the decomposition of each Pauli error $O$ into three components: $O = S\cdot L\cdot E$, where $S$ represents the stabilizer group element, $L$ denotes the logical operator, and $E$ corresponds to the pure error operator. Each component comprises Pauli operators acting on all encoded qubits under the given stabilizer code, with each operator parametrized by a binary configuration $\{\alpha,\beta,\gamma\}$ indicating the activation state of each constituent operator. We will abuse the notation of syndrome $\gamma\in\{+1,-1\}^m$ and pure error configuration $\gamma\in\{0,1\}^m$. Consequently, the pure error configuration $\gamma\in\{0,1\}^m$ is fixed by the syndrome, requiring summation over all stabilizer group elements (parametrized by $\alpha\in\{0,1\}^m$) to determine the optimal logical operator. For the details of the ELS decomposition, we refer to Ref. [27]. Then the probability of the error belonging to the logical sector $L$ is given by

$$P(L) = \sum_{S\in\mathcal{S}} P[O(S,L,E)], \tag{3}$$

and the most probable logical operator can be selected using

$$L_{\text{optimal}} = \arg\max_L \sum_{S\in\mathcal{S}} P[O(S,L,E)]. \tag{4}$$

The challenge of the MLD as shown in the last equation is performing the summation over all possible stabilizer operators $S\in\mathcal{S}$, which requires enumerating all possible binary configurations $\alpha\in\{0,1\}^m$. This problem belongs to the #P complexity class, and no general polynomial-time algorithms exist for solving it exactly. However, in Ref. [25], it has been shown that for quantum codes with planar structures, such as the repetition codes with circuit-level noise, there exist exact MLD algorithms with polynomial complexity.

### 2.2. Mapping to planar Ising model and exact solution via Kac–Ward formula

To achieve a polynomial-complexity exact algorithm for quantum codes with a planar structure, PLANAR first maps the decoding problem to the partition function computation of an Ising spin glass model on a planar graph. The mapping allows the error correction problem to be expressed as

a statistical mechanics problem, where the partition function of the Ising model corresponds to the probability of the sum of all error configurations consistent with the syndrome measurements and belongs to the same logical sector. During the mapping, stabilizer generators serve as spins while data qubits correspond to (hyper)edges. The Hamiltonian is formulated as[28]

$$H_O(\boldsymbol{s}) = -\sum_{\sigma}\sum_{i=1}^{n}\langle\sigma, O_i\rangle \cdot J(\sigma)_i \prod_{\{k|\langle\sigma,(g_k)_i\rangle=-1\}} s_k, \tag{5}$$

where $O_i$ denotes the component of Pauli error operator $O$ acting on the $i$-th qubit. $(g_k)_i$ is the $i$-th component of the $k$-th stabilizer. Such representation holds under the Nishimori condition[29]

$$J(\sigma) = \frac{1}{4}\sum_{\rho}\langle\sigma,\rho\rangle \cdot \log p(\rho), \quad \sigma,\rho \in \{I,x,y,z\}, \tag{6}$$

where $\langle\ \rangle$ represents the scalar commutator which equals 1 if the two inputs commute and $-1$ otherwise, $p(\rho)$ represents the probability of error $\rho$ occurring on a specific qubit. We can easily check the probability of an error

$$\begin{aligned}
P(O) &= \mathrm{e}^{-H_O(s=1)} \\
&= \exp\left\{\sum_{i=1}^{n}\sum_{\sigma}\langle\sigma, O_i\rangle \cdot J(\sigma)_i\right\} \\
&= \exp\left\{\sum_{i=1}^{n}\sum_{\sigma}\langle\sigma, O_i\rangle \cdot \frac{1}{4}\sum_{\rho}\langle\sigma,\rho\rangle \cdot \log p(\rho)\right\} \\
&= \exp\left\{\sum_{i=1}^{n}\frac{1}{4}\sum_{\sigma}\sum_{\rho}\langle\sigma, O_i\cdot\rho\rangle \cdot \log p(\rho)\right\} \\
&\qquad (\langle A,B\rangle\cdot\langle A,C\rangle = \langle A,BC\rangle) \\
&= \exp\left\{\sum_{i=1}^{n}\sum_{\rho}\delta_{\rho,O_i} \cdot \log p(\rho)\right\} \\
&\qquad \left(\frac{1}{4}\sum_{\sigma}\langle\sigma,\rho\rangle = \delta_{\rho,I}\right) \\
&= \prod_{i=1}^{n} p(O_i).
\end{aligned}$$

Non-trivial configurations of spins means there are stabilizers acting on the error $O$ and causing a new error $O'$ which belongs to the same equivalence class as $O$. Therefore, summing all elements which are equivalent to $O$ is equal to summing all spin configurations, i.e., the coset probability of a quantum code equals the partition function of the corresponding Ising model,

$$P(L) = \mathcal{Z}_L.$$

The planar nature of the Ising model ensures that the problem can be solved exactly using polynomial-time algorithms such as the Kac–Ward formula,[30] as discussed in Subsection 3.3.3. This approach selects the most likely logical section, eliminating traditional methods' suboptimality. Moreover, the polynomial complexity of the Kac–Ward solution renders it highly efficient for large-scale quantum codes.

## 3. Implementation of PLANAR demonstrated under the rotated surface code

PLANAR is designed with modularity in mind, separating graph construction, probability calculation, and decoding logic into distinct components. This modular design enables straightforward extension to diverse code families and error models. The software handles both $X$ and $Z$ errors independently, ensuring optimal performance for all error types. In this section, we will take the standard rotated surface code as an illustrative example to explain the implementation of the PLANAR decoder. Under the uncorrelated code-capacity noise model, the decoding graph of the rotated surface code exhibits a planar structure, making it a simple and ideal candidate for decoding with PLANAR.

### 3.1. The rotated surface code

We first introduce the rotated surface code, a commonly used quantum error correction code on a two-dimensional lattice. It is constructed from a two-dimensional lattice, where each plaquette represents a stabilizer generator, and each vertex represents a qubit. To convert the decoding problem into a partition-function computation problem of a spin glass, we first index every qubit in the rotated surface lattice, which corresponds to the edges in Fig. 1(a). Then, we represent each stabilizer generator, pure error generator, and logical operator by a parity check matrix (PCM). In general, the logical generator can be generated by calculating the kernel of the PCM of the stabilizer generator. For the rotated surface code, we can easily find it using the symmetry of the code. This is achieved by the function `gen_rotated_surface_code` in `experiments/codes.py` in our implementation.[26] Then, we construct the PCM of the pure error generator by using the commutation relation between the stabilizer generator and the pure error generator. More specifically, we calculate the kernel of the PCM of the stabilizer generator and check if it commutes with the logical operator. For the detailed mathematics, we refer to Ref. [25]. In our implementation, it is implemented in the function `cal_pure_error_basis` in `planar/planar_decoder.py` in Ref. [26].

### 3.2. Generation of the decoding graph

In this section, we introduce how to convert the rotated surface code into a planar Ising model on a *decoding graph*. Here we consider the independent code-capacity noise, where the $X$ stabilizers and $Z$ stabilizers belong to two separate graphs. We first construct the planar decoding graph from the PCM for both $X$ and $Z$ stabilizer generators. The process generates dangling edges, as illustrated in Fig. 1(b), necessitating the introduction of a virtual node to connect these edges for each decoding graph, as shown in Fig. 1(c). The Kac–Ward solution is then applied, implemented by the function `construct_kac_ward_solution` in `planar/planar_decoder.py` as detailed in Ref. [26].

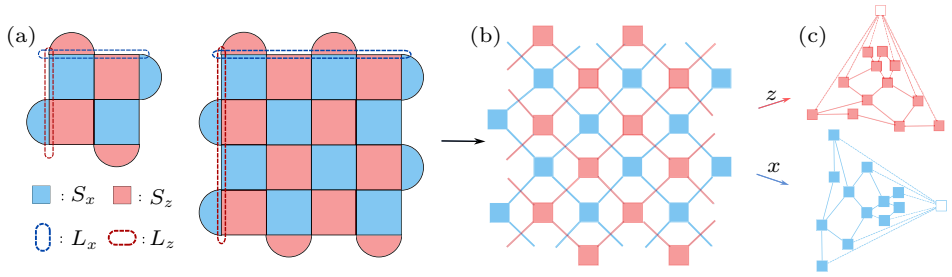



**Fig. 1.** (a) Rotated surface code lattice, the left side is a distance-3 rotated surface code lattice, while the right side is a distance-5 one. Each blue/red plaquette represents an $X/Z$-type stabilizer, and each vertex represents a qubit. The blue/red dashed line indicates the $X/Z$-type logical operator. (b) Separately converting the rotated surface code into a planar Ising model, the red part denotes the $Z$ type while the blue part denotes the $X$ type. Each square corresponds to a different spin. (c) A decoding graph is used for the Kac–Ward solution by adding an auxiliary spin connecting each dangling bond.

```python
def construct_kac_ward_solution(pcm):
    n = pcm.shape[0]
    edges_dict = {}
    for m in range(pcm.shape[1]):
        edge = pcm[:, m].nonzero()[0]
        assert len(edge) <= 2, print('non-planar graph')
        if len(edge) == 1:
            if (edge[0], n) not in edges_dict.keys():
                edges_dict[(edge[0], n)] = [m]
            else:
                edges_dict[(edge[0], n)].append(m)
        else:
            assert edge[0] < edge[1]
            if (edge[0], edge[1]) not in edges_dict.keys():
                edges_dict[(edge[0], edge[1])] = [m]
            else:
                edges_dict[(edge[0], edge[1])].append(m)
    g = nx.from_edgelist(edges_dict.keys())
    kwsolution = KacWardSolution(nx.adjacency_matrix(g, nodelist=np.arange(n+1)))
    return kwsolution, edges_dict
```

### 3.3. Coset probability calculation

This step calculates the partition function of the Ising model generated in step 2. To apply the Kac–Ward formula, a non-backtracking matrix is first constructed to encode the spatial relationships between edges in the graph. This matrix is constructed during the initialization of the class `KacWardSolution`. Specifically, each entry in the matrix represents the angle between connected edges. The graph layout, including node positions, is represented using `networkx` to calculate these angles, which are then expressed as complex numbers in the non-backtracking matrix.

```python
class KacWardSolution:
    def __init__(self, adj_matrix) -> None:
        adj = scipy.sparse.csr_matrix(adj_matrix)
        graph = nx.from_scipy_sparse_array(adj)
        is_planar, _ = nx.check_planarity(graph)
        if not is_planar:
            raise ValueError("The graph is not planar, the KacWard solution does not apply.")
        pos = nx.planar_layout(graph)
        self.n, self.m = graph.number_of_nodes(), graph.number_of_edges()
        di_edge_list = sum([[(i,j), (j,i)] for (i,j) in graph.edges()], start=[])
        self.di_edge_dict = {edge:idx for idx, edge in enumerate(di_edge_list)}
        self.nonbacktrack = scipy.sparse.lil_matrix(
            (2*self.m, 2*self.m), dtype=np.complex128)
        for i, j in di_edge_list:
            for k in graph.neighbors(j):
                if k != i:
                    self.nonbacktrack[
                        self.di_edge_dict[(i,j)],
                        self.di_edge_dict[(j,k)]
                    ] = np.exp(
                        1J/2. * compute_angle(
                            pos[i], pos[j], pos[k])
        pass
```

The partition function of the Ising model is then computed using the Kac–Ward formula

$$Z = 2^n \left( \prod_{\langle ij \rangle} \cosh \theta_{ij} \right) \det(I - W)^{\frac{1}{2}}. \tag{7}$$

The non-backtracking matrix $A$ and the matrix for computing $\det(I-W)$ are represented in sparse format, where $W=AD$. Here, $D$ is a diagonal matrix with $D_{ij,ij}=J_{ij}$, corresponding to the edge weights, as defined by the formula

$$A_{ij,kl}=\begin{cases}\exp(\frac{i}{2}\phi_{ijl}), & j=k \text{ and } i\neq l,\\ 0, & \text{otherwise},\end{cases} \tag{8}$$

where $\phi_{ijl}$ denotes the angle between the edges $ij$ and $kl$, which corresponds to the angle formed by the nodes $i$, $j$, and $l$.

The $LU$ decomposition of matrix $A$ is computed using `scipy.sparse.linalg.splu(A)`. The real part of the sum of the logarithms of the diagonal elements of $U$ yields the logarithmic partition function. This is implemented in the function `logZ` within the `KacWardSolution` class.

```
    def logZ(self, weights):
        D = scipy.sparse.spdiags(np.tanh(weights), 0, len(weights), len(weights))
        logZ = self.n * np.log(2.0)
        logZ += logcosh(weights).sum() / 2
        A = scipy.sparse.eye(2*self.m, 2*self.m, dtype=np.complex128) - self.nonbacktrack @ D
        A = scipy.sparse.csc_matrix(A)
        LU = scipy.sparse.linalg.splu(A)
        logdet = np.log(LU.U.diagonal()).sum()
        comp_part = np.imag(logdet)/pi
        logdet = np.real(logdet)
        logZ += 0.5*logdet
        return logZ
```

### 3.4. Using Kac–Ward solution to determine the most likely logical recovery operator

Following the Kac–Ward solution for the partition function of the planar Ising model, the probabilities of different logical sectors (cosets) are computed to identify the most probable logical recovery operator. This is implemented in the `log_coset_p` and `decode` functions within the PLANAR decoder.[26]

#### 3.4.1. Coset probability calculation

The function accepts the pure error operator, with or without the corresponding logical operator ($X/Z$ or $I$, representing distinct logical sectors), the error rate, the Kac–Ward solution, and a dictionary mapping graph edges to indices consistent with the Kac–Ward solution. It computes edge weights based on the error rate, which are then used to determine the logarithm of the partition function ($\log Z$) via the Kac–Ward solution. The output is the log probability for the specified logical error configuration, enabling comparison across logical sectors.

#### 3.4.2. Decoding process

The decoding process `decode` begins by determining the pure error corresponding to the syndrome, which represents an error configuration consistent with the syndrome measurements. For both $X$ and $Z$ components of the decoding graph, the `log_coset_p` function calculates the probabilities of different logical cosets (i.e., the pure error combined with or without the logical operator). The most probable logical operator is selected by comparing these probabilities. The recovery operation is then constructed by combining the pure error with the chosen logical operator.

```
def log_coset_p(operator, error_rate, solution:KacWardSolution, edges_dict):
    if len(operator.shape) >= 2:
        assert operator.shape[0] == 1, print(operator.shape)
    operator = operator.reshape(-1)
    weight = np.array([
        0.5*np.log((1-error_rate)/error_rate),
        0.5*np.log(error_rate/(1-error_rate)),
    ])
    weights = weight[operator]
    weights_nonbacktrack = np.array([
        sum(weights[edges_dict[minmax(*edge)]]) for edge in solution.di_edge_dict
    ])
    logZ = solution.logZ(weights_nonbacktrack)
    return logZ
```

This approach ensures the decoder selects the most likely error configuration, achieving maximum likelihood decoding. The Kac–Ward solution guarantees exact probability computations, eliminating the suboptimality associated with minimum weight methods such as MWPM.

```python
def decode(self, syndrome, error_prob):
    pure_error = self.pure_error_basis[
    syndrome.astype(bool)
    ].sum(0) % 2
    assert np.allclose(syndrome, (self.stabilizers @ pure_error) % 2)
    pure_error_x, pure_error_z = pure_error[:self.n], pure_error[self.n:]
    error_rate_x = error_prob[1] + error_prob[3]
    error_rate_z = error_prob[2] + error_prob[3]
    probs_x = [
        log_coset_p(pure_error_x, error_rate_x, self.kwx, self.edges_dict_x),
        log_coset_p((pure_error_x + self.lx) % 2, error_rate_x, self.kwx, self.edges_dict_x)
    ]
    ind_x = np.argmax(probs_x)
    probs_z = [
        log_coset_p(pure_error_z, error_rate_z, self.kwz, self.edges_dict_z),
        log_coset_p((pure_error_z + self.lz) % 2, error_rate_z, self.kwz, self.edges_dict_z)
    ]
    ind_z = np.argmax(probs_z)
    recovery_op = pure_error
    if ind_x: recovery_op = (recovery_op + self.logicals[0]) % 2
    if ind_z: recovery_op = (recovery_op + self.logicals[1]) % 2
    return recovery_op
```

## 4. Numerical results

We first employ the rotated surface code family to evaluate the performance of PLANAR. To determine the threshold of the rotated surface code, we adopt the standard assumption of uncorrelated noise. In this model, the error probabilities for each qubit are defined as follows:

$$\begin{aligned} p(I) &= (1-p_x)(1-p_z), \\ p(X) &= p_x(1-p_z), \\ p(Z) &= p_z(1-p_x), \\ p(Y) &= p_x p_z. \end{aligned} \tag{9}$$

Under the uncorrelated noise assumption, there are no correlations between $X$ and $Z$ types of stabilizers, implying that $J(Y) = 0$ in the system's Hamiltonian as shown in Eq. (5). This simplification allows the Hamiltonian to be decomposed into two independent terms corresponding to $X$ and $Z$ errors:

$$\begin{aligned} H_{uc} = &-\sum_e J_X \langle X, O_e \rangle s_i^z s_j^z - \sum_e J_Z \langle Z, O_e \rangle s_k^x s_l^x, \\ &i,j,k,l \in \partial e, \end{aligned} \tag{10}$$

where $e$ denotes edges, and $i$, $j$ and $k$, $l$ represent spins connected by these edges in the $X$ or $Z$ subgraph. This formulation demonstrates that the subsystems associated with $X$ and $Z$ errors are decoupled. Thus, the partition function of the entire system can be expressed as the product of the partition functions of the two subsystems, i.e., $\mathcal{Z} = \mathcal{Z}_x \cdot \mathcal{Z}_z$. This independence extends to the logical error probabilities $P(L_x)$ and $P(L_z)$, such that $P(L_x, L_z) = P(L_x) \cdot P(L_z)$. Consequently, the original Ising model graph can be partitioned into $X$ and $Z$ subgraphs, each forming a planar lattice that retains the planar structure. This decoupling simplifies the analysis and improves the computational efficiency of the decoding process.

We first evaluated PLANAR on the rotated surface code, testing its performance for code distances ranging from 9 to 25. The decoder achieves optimal maximum-likelihood decoding results for this error model, with a threshold of approximately $p_{\text{uc}} \approx 0.109$, as depicted in the main graph of Fig. 2(a). Additionally, as observed from the subgraph in the upper left corner, the decoding time for each trial scales polynomially with the increase in the number of spins, which corresponds to the nodes in Fig 1(c). This polynomial scaling behavior indicates that the algorithm remains computationally efficient even as the system size grows. The algorithm's time complexity is quantified as $O(N^{0.69})$, where $N$ represents the number of spins in the Ising model. This scaling value further underscores the efficiency of the algorithm, making it suitable for practical applications in large-scale quantum error correction tasks. The observed scaling behavior aligns with theoretical expectations and demonstrates the algorithm's capability to handle increasingly complex systems without significant computational overhead.

It is worth noting that, under this error model, PLANAR provides a theoretically optimal solution, eliminating the need for comparison with other decoders. This optimality is crucial for practical quantum error correction, as it ensures the highest possible accuracy in error detection and correction.

This theoretical guarantee is particularly significant in the context of fault-tolerant quantum computing, where even small error rates can compromise the integrity of quantum computations. By providing an optimal solution, PLANAR not only enhances the reliability of quantum error correction but also sets a benchmark against which other decoding algorithms can be evaluated. Furthermore, the elimination of the need for comparative analysis with other decoders simplifies the implementation and validation process, allowing researchers and practitioners to focus on optimizing other aspects of quantum error correction protocols.

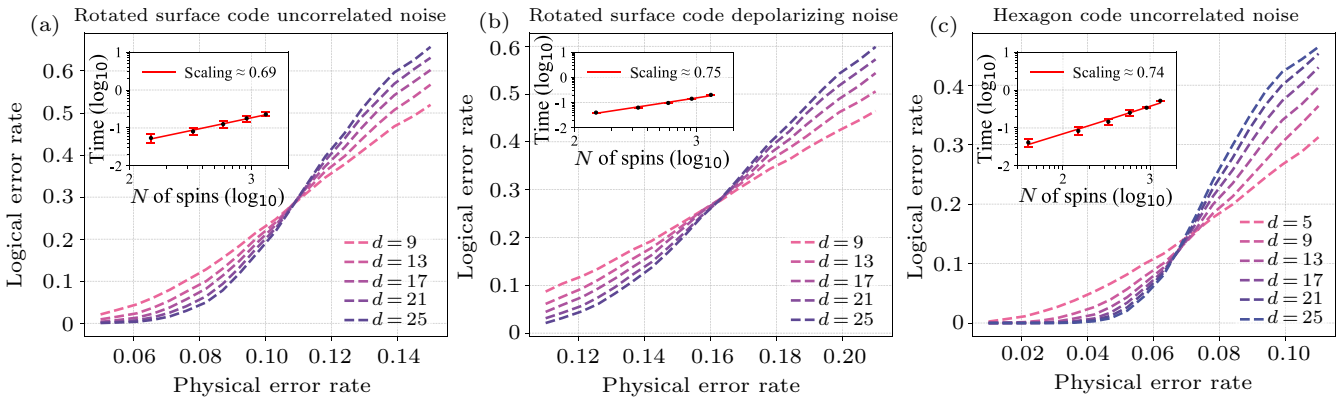



**Fig. 2.** Numerical results of simulations are presented. (a) The decoding performance of the rotated surface code under the uncorrelated noise model is shown, where each colored line represents a different code distance ranging from 9 to 25. The threshold $p_{\text{uc}}$ is observed at a physical error rate of approximately 0.109, where the lines converge. The subgraph in the upper left corner illustrates the time scaling behavior with respect to the increasing number of spins, which corresponds to the number of nodes in Fig. 1(c). (b) The decoding performance of the rotated surface code under the depolarizing noise model with an uncorrelated approximation is depicted, yielding a threshold $p_{\text{uc}}^{\text{dep}}$ at a physical error rate near 0.164. This result aligns with the theoretical prediction of $p_{\text{uc}}^{\text{dep}} = \frac{3}{2}p_{\text{uc}} \approx 0.164$.[29] (c) The decoding performance of the hexagon code under the uncorrelated noise model is displayed, with code distances ranging from 5 to 25. The threshold is observed around 0.007, and the time complexity scales as $O(N^{0.74})$.

Furthermore, we also tested PLANAR on the commonly used depolarizing error model, where the error probabilities are defined as

$$\begin{aligned} p(X) = p(Y) = p(Z) = p/3, \\ p(I) = 1 - p. \end{aligned} \tag{11}$$

Consequently, the system's Hamiltonian includes an additional term compared to Eq. (10), corresponding to the contribution of $J(Y)$:

$$\begin{aligned} H_{\text{dep}} = &-\sum_e J_X \langle X, O_e \rangle s_i^z s_j^z - \sum_e J_Z \langle Z, O_e \rangle s_k^x s_l^x \\ &-\sum_e J_Y \langle Y, O_e \rangle s_i^x s_j^x s_k^z s_l^z, \quad i,j,k,l \in \partial e. \end{aligned} \tag{12}$$

To apply PLANAR to this error model, we approximate the contribution of $J(Y)$ by separating it into its $X$ and $Z$ components, eliminating the correlation caused by the $Y$ component, redefining the error probabilities for each qubit as follows:

$$\begin{aligned} p(X) = p(Z) = \frac{2}{3}p, \\ p(I) = 1 - \frac{2}{3}p. \end{aligned} \tag{13}$$

This approximation is also utilized by several well-known minimum-weight decoders, such as MWPM[17] and BP-OSD.[31] However, these decoders consistently exhibit inferior decoding performance compared to PLANAR, even when employing the same uncorrelated approximation. This discrepancy arises because minimum-weight decoders introduce additional approximations during their decoding process, as they aim to identify the most probable error (i.e., the error with the minimum weight). In contrast, PLANAR rigorously seeks to determine the maximum likelihood logical sector without such approximations. The ability of PLANAR to directly target the maximum likelihood solution, rather than relying on intermediate approximations, underscores its theoretical and practical advantages over conventional minimum-weight decoders.

Under this uncorrelated approximation, PLANAR still yields remarkable results for the rotated surface code with depolarizing noise model, as shown in Fig. 2(b), with a threshold of approximately $p_{\text{uc}}^{\text{dep}} \approx 0.164$ and time complexity scaling as $O(N^{0.75})$. This result of threshold matches the theoretical prediction of $p_{\text{uc}}^{\text{dep}} = \frac{3}{2}p_{\text{uc}} \approx 0.164$,[29] and also demonstrates the robustness and adaptability of PLANAR in handling different error models, which is essential for its application in diverse quantum computing environments.

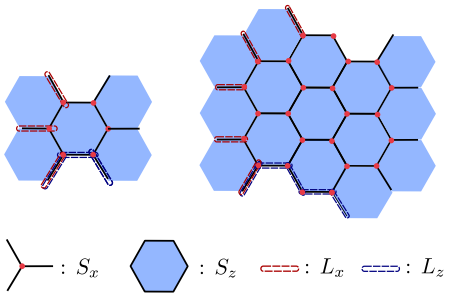



**Fig. 3.** Hexagon code lattice. The left side shows a distance-3 hexagon code lattice, while the right side shows a distance-5 lattice. Each vertex/plaquette represents an $X/Z$-type stabilizer, and each edge represents a qubit. The red/blue dashed lines indicate $X/Z$-type logical operators.

To further demonstrate the generality of PLANAR, we also tested it on the hexagon code, a novel quantum error correction code with a planar structure, as shown in Fig. 3. The hexagon code is constructed from a hexagonal lattice, where each vertex and plaquette represents a stabilizer generator, and each edge represents a qubit. The planar structure of the hexagon code ensures that the decoding problem can be mapped to a planar Ising model, enabling efficient and exact decoding with PLANAR. For the hexagon code, we tested PLANAR on code distances ranging from 5 to 25. The decoder achieves opti-

mal maximum-likelihood decoding results under the uncorrelated noise model as well, with a threshold of approximately $p_{\mathrm{uc}}^{\mathrm{hex}} \approx 0.07$ and time complexity scaling as $O(N^{0.74})$, as shown in Fig. 2(c). These results further validate the versatility and efficiency of PLANAR for decoding quantum codes with planar structures. The ability of PLANAR to handle different code geometries and error models underscores its potential as a universal decoding tool in quantum error correction.

## 5. Conclusion

In this work, we presented the implementation details of PLANAR, a software framework designed for the exact decoding of quantum codes with planar structures. By leveraging the Kac–Ward formula, PLANAR achieves maximum likelihood decoding with polynomial time complexity. The introduction of the rotated surface code demonstrates the practical applicability of PLANAR, showcasing its ability to efficiently handle complex quantum error correction tasks. We also presented a new quantum error correction code, the hexagon code, to demonstrate the generality of PLANAR.

Our numerical results indicate that the rotated surface code using PLANAR achieves a threshold of approximately $p_{\mathrm{uc}} \approx 0.109$ under the uncorrelated error model, with a time complexity scaling of $O(N^{0.69})$. For hexagon codes, these values would be $p_{\mathrm{uc}}^{\mathrm{hex}} \approx 0.07$ and $O(N^{0.74})$. This performance highlights the potential of PLANAR for real-world quantum error correction applications, particularly in large-scale quantum systems where efficiency and accuracy are paramount. Future work will focus on extending PLANAR to other code families that have planar structures and optimizing its implementation for faster or even real-time decoding in experimental quantum hardware. The modular design of PLANAR ensures its adaptability to various quantum error correction scenarios, making it a versatile tool for advancing fault-tolerant quantum computing.

## Program availability

The source code is openly available on GitHub at https://github.com/CHY-i/planar.

## Acknowledgements

This work is supported by the National Natural Science Foundation of China (Grant Nos. 12325501, 12047503, and 12247104) and the Chinese Academy of Sciences (Grant No. ZDRW-XX-2022-3-02). P. Z. is partially supported by the Innovation Program for Quantum Science and Technology (Grant No. 2021ZD0301900).

## References

[1] Google Quantum AI 2021 *Nature* **595** 383
[2] Google Quantum AI and Collaborators 2024 *Nature* **638** 920
[3] Google Quantum AI 2023 *Nature* **614** 676
[4] Bluvstein D, Evered S J, Geim A A, Li S H, Zhou H, Manovitz T, Ebadi S, Cain M, Kalinowski M, Hangleiter D, *et al*. 2024 *Nature* **626** 58
[5] Honciuc Menendez D, Ray A and Vasmer M 2024 *Phys. Rev. A* **109** 062438
[6] Hong Y, Durso-Sabina E, Hayes D and Lucas A 2024 *Phys. Rev. Lett.* **133** 180601
[7] Da Silva M, Ryan-Anderson C, Bello-Rivas J, Chernoguzov A, Dreiling J, Foltz C, Gaebler J, Gatterman T, Hayes D, Hewitt N, *et al*. 2024 arXiv preprint arXiv: 2404.02280
[8] Mayer K, Ryan-Anderson C, Brown N, Durso-Sabina E, Baldwin C H, Hayes D, Dreiling J M, Foltz C, Gaebler J P, Gatterman T M, *et al*. 2024 arXiv preprint arXiv: 2404.08616
[9] Ryan-Anderson C, Bohnet J G, Lee K, Gresh D, Hankin A, Gaebler J, Francois D, Chernoguzov A, Lucchetti D, Brown N C, *et al*. 2021 *Phys. Rev. X* **11** 041058
[10] Fowler A G, Mariantoni M, Martinis J M and Cleland A N 2012 *Phys. Rev. A* **86** 032324
[11] Campbell E T 2019 *Quantum Science and Technology* **4** 025006
[12] Bonilla Ataides J P, Tuckett D K, Bartlett S D, Flammia S T and Brown B J 2021 *Nat. Commun.* **12** 2172
[13] Hastings M B and Haah J 2021 *Quantum* **5** 564
[14] Fowler A G 2011 *Phys. Rev. A* **83** 042310
[15] Bravyi S, Cross A W, Gambetta J M, Maslov D, Rall P and Yoder T J 2024 *Nature* **627** 778
[16] Gottesman D 1997 *Stabilizer codes and quantum error correction* (California Institute of Technology)
[17] Higgott O 2022 *ACM Transactions on Quantum Computing* **3** 1
[18] Fowler A G 2013 arXiv preprint arXiv: 1310.0863
[19] Higgott O, Bohdanowicz T C, Kubica A, Flammia S T and Campbell E T 2023 *Phys. Rev. X* **13** 031007
[20] Kolmogorov V 2009 *Mathematical Programming Computation* **1** 43
[21] Liyanage N, Wu Y, Deters A and Zhong L 2023 *Scalable quantum error correction for surface codes using fpga* 2023 *IEEE International Conference on Quantum Computing and Engineering (QCE)* Vol. 01 pp. 916–927
[22] Bohdanowicz T C, Crosson E, Nirkhe C and Yuen H 2019 *Tensor networks for exact and approximate maximum likelihood decoding Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing* (Phoenix AZ USA: ACM) pp. 481–490
[23] Piveteau C, Chubb C T and Renes J M 2024 *PRX Quantum* **5** 040303
[24] Bravyi S, Suchara M and Vargo A 2014 *Phys. Rev. A* **90** 032326
[25] Cao H, Zhao S, Feng D, Shen Z, Yan H, Su T, Sun W, Xu H, Pan F, Yu H, *et al*. 2025 arXiv preprint arXiv: 2501.03582
[26] https://github.com/CHY-i/planar
[27] Cao H, Pan F, Wang Y and Zhang P 2023 arXiv preprint arXiv: 2307.09025
[28] Chubb C T and Flammia S T 2021 *Annales de l'Institut Henri Poincaré D* **8** 269
[29] Bombin H, Andrist R S, Ohzeki M, Katzgraber H G and Martin-Delgado M A 2012 *Phys. Rev. X* **2** 021004
[30] Kac M and Ward J C 1952 *Phys. Rev.* **88** 1332
[31] Roffe J, White D R, Burton S and Campbell E 2020 *Phys. Rev. Res.* **2** 043423