Robust Topological Photonics using embedded Qubits & Holographic Qudits Under Realistic Noise

Sayan Sarkar

Under Guidance of : Dr. Supriyo De

Abstract

A simulation framework for topological photonic quantum computing is presented. It builds an SSH Hamiltonian in an expanded space, where each site is modeled as a two-level system, and incorporates realistic noise through amplitude damping and dephasing. Disorder is measured using the inverse participation ratio, quantifying state localization. Qubit operations are embedded in a protected subspace defined by edge states, and high-dimensional qudit encoding is achieved by redistributing orbital angular momentum modes with a discrete Fourier transform and nonlinear interactions. The framework also models full-lattice noise via the Lindblad master equation and analyzes a two-dimensional photonic Chern insulator by mapping Berry curvature and calculating the Chern number.

Contents

1 Introduction

Photon-based quantum computing is promising due to its fast processing and low decoherence. Topological photonics further protects information via edge states that are immune to local perturbations. However, realistic noise—such as amplitude damping and dephasing—remains a challenge. In this work, we improve upon previous models by:

- Correcting the amplitude damping noise channel via proper lowering operators in an extended Hilbert space.
- Quantifying disorder in the SSH model using the inverse participation ratio (IPR).
- Embedding qubit operations in the topologically protected subspace.
- Implementing holographic OAM qudit encoding with a discrete Fourier transform (DFT) and adding a Kerr-like nonlinear interaction.
- Modeling full-lattice noise using the Lindblad master equation and extending analysis to a 2D photonic Chern insulator.

The remainder of the paper describes the theoretical framework, detailed simulation code, output analysis, and a comparison with existing approaches.

2 Theoretical Framework

2.1 SSH Model and Topological Edge States

The Su-Schrieffer-Heeger (SSH) model describes a 1D lattice with alternating hopping amplitudes:

$$H_{\mathrm{SSH}} = \sum_{i=0}^{N-2} t_i \left(a_i^\dagger a_{i+1} + a_{i+1}^\dagger a_i \right), \tag{1}$$

with

$$t_i = \begin{cases} t_1, & \text{if } i \text{ is even,} \\ t_2, & \text{if } i \text{ is odd.} \end{cases}$$

Disorder is introduced by perturbing t_i randomly (up to 30% variation). The topologically nontrivial phase supports edge states that are localized at the boundaries. To accurately simulate photon loss, we extend the Hilbert space so that each lattice site is modeled as a two-level system.

2.2 Embedded Qubit Operations

Given two edge states $|\psi_{\mathrm{edge}}\rangle$ and $|\psi_{\mathrm{orth}}\rangle$, a qubit operator A is embedded via:

$$A_{\mathrm{embed}} = U\, A\, U^\dagger, \tag{2}$$

where

$$U = \left(|\psi_{\mathrm{edge}}\rangle \quad |\psi_{\mathrm{orth}}\rangle \right).$$

This confines operations to the topologically protected subspace.

2.3 Holographic OAM Qudit Encoding and Nonlinear Interaction

High-dimensional encoding is achieved using OAM states $\{|l\rangle\}$, $l = -L, \ldots, L$. Holographic mode conversion is realized via a DFT:

$$U_{\text{DFT}} = \frac{1}{\sqrt{d}} \left[\omega^{ij}\right]_{i,j=0}^{d-1}, \quad \omega = e^{2\pi i/d}. \tag{3}$$

A Kerr-like nonlinear phase evolution

$$H_{\text{nl}} = \chi \sum_{l} l^2 |l\rangle\langle l|$$

is combined with the DFT to capture nonlinear interactions.

2.4 Full-Lattice Noise Modeling

Decoherence is modeled via the Lindblad master equation:

$$\frac{d\rho}{dt} = -\frac{i}{\hbar}[H, \rho] + \sum_{k} \left(L_k \rho L_k^{\dagger} - \frac{1}{2}\{L_k^{\dagger} L_k, \rho\} \right), \tag{4}$$

with appropriate Lindblad operators for amplitude damping (using proper annihilation operators) and dephasing.

2.5 2D Photonic Chern Insulator

For a 2D model, the Qi–Wu–Zhang Hamiltonian is used:

$$H(\mathbf{k}) = \sin k_x \, \sigma_x + \sin k_y \, \sigma_y + (m + \cos k_x + \cos k_y) \, \sigma_z, \tag{5}$$

and the Berry curvature is computed over the Brillouin zone to obtain the Chern number:

$$C = \frac{1}{2\pi} \int_{\text{BZ}} F(\mathbf{k}) \, d^2k.$$

3 Simulation Code

Below is the complete Python code used in our simulation. The code is structured into sections for the SSH model, embedded qubit operations, holographic OAM encoding, noise modeling, and 2D Chern insulator analysis.

```python
#!/usr/bin/env python3
"""
Fault-Tolerant Topological Photonic Quantum Computing Simulation
-----------------------------------------------------------------
This script implements:
  - SSH Hamiltonian in an extended Hilbert space.
  - Correct amplitude damping using proper annihilation operators.
  - IPR calculation for disorder quantification.
  - Embedded qubit operations in the topologically protected subspace.
```

```
  - Holographic OAM qudit encoding (DFT) with Kerr-like nonlinearity.
  - Full-lattice noise modeling via Lindblad master equations.
  - 2D Chern insulator analysis via Berry curvature and Chern number.
"""

import numpy as np
import qutip as qt
import matplotlib.pyplot as plt

plt.style.use('ggplot')
np.random.seed(42)

# --- Utility functions for extended Hilbert space (2-level systems per site) ---
def operator_on_site(op, i, N):
    op_list = [qt.qeye(2) for _ in range(N)]
    op_list[i] = op
    return qt.tensor(op_list)

def destroy_site(i, N):
    return operator_on_site(qt.destroy(2), i, N)

def create_site(i, N):
    return operator_on_site(qt.create(2), i, N)

# --- SSH Hamiltonian with disorder ---
def ssh_hamiltonian_extended(N, t1, t2, disorder=0.0):
    H = 0
    for i in range(N-1):
        base_t = t1 if (i % 2 == 0) else t2
        t_val = base_t * (1 + disorder*(np.random.rand()-0.5))
        H += t_val * (create_site(i, N) * destroy_site(i+1, N))
        H += np.conjugate(t_val) * (create_site(i+1, N) * destroy_site(i, N))
    return H

def compute_single_excitation_eigensystem(H, N):
    total_excitation = sum(operator_on_site(qt.num(2), i, N) for i in range(N))
    evals, evecs = H.eigenstates()
    single_ex_evals, single_ex_evecs = [], []
    for e, psi in zip(evals, evecs):
        if abs(qt.expect(total_excitation, psi) - 1.0) < 1e-5:
            single_ex_evals.append(e)
            single_ex_evecs.append(psi)
    return np.array(single_ex_evals), single_ex_evecs

def compute_ipr(psi, N):
    ipr = 0
    for i in range(N):
        proj_i = operator_on_site(qt.basis(2,1)*qt.basis(2,1).dag(), i, N)
        ipr += (qt.expect(proj_i, psi))**2
    return ipr
```

```python
# Parameters
N = 6
t1, t2 = 0.5, 1.0
disorder_strength = 0.3

H_ssh = ssh_hamiltonian_extended(N, t1, t2, disorder=disorder_strength)
evals, evecs = compute_single_excitation_eigensystem(H_ssh, N)

ipr_values = [compute_ipr(psi, N) for psi in evecs]
print("IPR for single-excitation eigenstates:", np.round(ipr_values,4))

# Select two edge states (lowest absolute eigenvalues)
idx_sort = np.argsort(np.abs(evals))
edge_state_1 = evecs[idx_sort[0]]
edge_state_2 = evecs[idx_sort[1]]

# Plot edge state distributions
fig, axes = plt.subplots(1, 2, figsize=(12,4))
for i, psi_edge in enumerate([edge_state_1, edge_state_2]):
    probs = [qt.expect(operator_on_site(qt.basis(2,1)*qt.basis(2,1).dag(), s, N), psi_edge)
             for s in range(N)]
    axes[i].bar(range(N), probs, color='royalblue')
    axes[i].set_title("Edge State " + str(i+1))
    axes[i].set_xlabel("Site Index")
    axes[i].set_ylabel("Excitation Probability")
plt.tight_layout()
plt.show()

# --- Embedded Qubit Operations ---
class EmbeddedQubit:
    def __init__(self, psi_edge1, psi_edge2, N):
        self.psi_edge1 = psi_edge1
        self.psi_edge2 = psi_edge2
        self.N = N
        self.op_dims = [[2]*N, [2]*N]
        col1 = psi_edge1.full().ravel()
        col2 = psi_edge2.full().ravel()
        self.U = np.column_stack((col1, col2))
    def embed_operator(self, A_2x2):
        A2 = A_2x2.full()
        A_emb = self.U @ A2 @ self.U.conj().T
        return qt.Qobj(A_emb, dims=self.op_dims)
    def measurement_operators(self, basis='z'):
        if basis.lower() == 'z':
            P0 = 0.5*(qt.qeye(2)+qt.sigmaz())
            P1 = 0.5*(qt.qeye(2)-qt.sigmaz())
        elif basis.lower() == 'x':
            P0 = 0.5*(qt.qeye(2)+qt.sigmax())
            P1 = 0.5*(qt.qeye(2)-qt.sigmax())
        else:
            raise ValueError("Unsupported basis.")
```

```
        return self.embed_operator(P0), self.embed_operator(P1)

def measure_in_subspace(rho, P_plus, P_minus):
    p_plus = (P_plus*rho).tr().real
    p_minus = (P_minus*rho).tr().real
    if (p_plus+p_minus)<1e-14:
        return 0, rho
    if np.random.rand() < p_plus/(p_plus+p_minus):
        post = (P_plus*rho*P_plus)/p_plus
        return +1, post
    else:
        post = (P_minus*rho*P_minus)/p_minus
        return -1, post

def repeated_measurement(rho, P_plus, P_minus, num_trials=1000):
    outcomes = [measure_in_subspace(rho, P_plus, P_minus)[0] for _ in range(num_trials)]
    plus = sum(1 for o in outcomes if o==+1)
    minus = num_trials - plus
    return plus, minus

qubit = EmbeddedQubit(edge_state_1, edge_state_2, N)
rho = edge_state_1 * edge_state_1.dag()
X_gate = qt.sigmax()
X_embedded = qubit.embed_operator(X_gate)
rho = X_embedded @ rho @ X_embedded.dag()

Pz_plus, Pz_minus = qubit.measurement_operators('z')
plus_z, minus_z = repeated_measurement(rho, Pz_plus, Pz_minus)
print("Z-basis: +1 =>", plus_z, ", -1 =>", minus_z)

Px_plus, Px_minus = qubit.measurement_operators('x')
plus_x, minus_x = repeated_measurement(rho, Px_plus, Px_minus)
print("X-basis: +1 =>", plus_x, ", -1 =>", minus_x)

# --- Holographic OAM Qudit Encoding & Nonlinear Interaction ---
def dft_operator(dim):
    omega = np.exp(2j*np.pi/dim)
    F = np.array([[omega**(i*j) for j in range(dim)] for i in range(dim)])/np.sqrt(dim)
    return qt.Qobj(F, dims=[[dim],[dim]])

def holographic_oam_gate(state, dim):
    U_DFT = dft_operator(dim)
    return U_DFT * state

def combined_nonlinear_interaction(dim, chi):
    l_vals = np.arange(-dim//2, dim//2)
    phases = np.exp(1j*chi*(l_vals**2))
    H_nl = qt.Qobj(np.diag(phases), dims=[[dim],[dim]])
    U_DFT = dft_operator(dim)
    return U_DFT * H_nl * U_DFT.dag()

```

```python
dim_OAM = 8
psi_oam = qt.rand_ket(dim_OAM)
probs_original = np.abs(psi_oam.full().flatten())**2
print("\nOriginal OAM probabilities:\n", np.round(probs_original,4))
psi_oam_dft = holographic_oam_gate(psi_oam, dim_OAM)
probs_dft = np.abs(psi_oam_dft.full().flatten())**2
print("\nDFT-transformed OAM probabilities:\n", np.round(probs_dft,4))

fig, axs = plt.subplots(1,2, figsize=(12,4))
axs[0].bar(range(dim_OAM), probs_original, color='royalblue')
axs[0].set_title("Original OAM")
axs[0].set_xlabel("OAM Mode")
axs[0].set_ylabel("Probability")
axs[1].bar(range(dim_OAM), probs_dft, color='seagreen')
axs[1].set_title("After DFT")
axs[1].set_xlabel("OAM Mode")
plt.tight_layout()
plt.show()

chi = 0.1
H_nl_holo = combined_nonlinear_interaction(dim_OAM, chi)
psi_oam_nl = H_nl_holo * psi_oam
probs_nl = np.abs(psi_oam_nl.full().flatten())**2
print("\nNonlinear + DFT OAM probabilities:\n", np.round(probs_nl,4))
plt.figure(figsize=(8,5))
plt.bar(range(dim_OAM), probs_nl, color='mediumpurple')
plt.title("OAM after Nonlinear + DFT")
plt.xlabel("OAM Mode")
plt.ylabel("Probability")
plt.show()

def analyze_distribution(probs, label):
    mean = np.sum(np.arange(len(probs))*probs)
    var = np.sum((np.arange(len(probs))-mean)**2 * probs)
    print(f"{label} => Mean: {mean:.3f}, Var: {var:.3f}")

analyze_distribution(probs_original, "Original")
analyze_distribution(probs_dft, "DFT-Transformed")
analyze_distribution(probs_nl, "Nonlinear+DFT")

# --- Full-Lattice Noise Modeling ---
def create_lindblad_operators_extended(num_sites, gamma_damp, gamma_dephase):
    L_ops = []
    for i in range(num_sites):
        L_damp = destroy_site(i, num_sites)
        L_ops.append(np.sqrt(gamma_damp)*L_damp)
        L_dephase = operator_on_site(qt.num(2), i, num_sites)
        L_ops.append(np.sqrt(gamma_dephase)*L_dephase)
    return L_ops

gamma_damp = 0.05
```

```python
gamma_dephase = 0.05
L_ops = create_lindblad_operators_extended(N, gamma_damp, gamma_dephase)
t_list = np.linspace(0, 10, 100)
result = qt.mesolve(H_ssh, rho, t_list, L_ops, [])
exp_vals = [qt.expect(X_embedded, st) for st in result.states]
plt.figure(figsize=(8,5))
plt.plot(t_list, exp_vals, label=r"$\langle X_{\rm embedded} \rangle$")
plt.xlabel("Time")
plt.ylabel("Expectation Value")
plt.title("Noise Evolution (Amplitude Damping + Dephasing)")
plt.legend()
plt.show()

# --- 2D Photonic Chern Insulator & Topological Phase Analysis ---
def hamiltonian_2D(kx, ky, m):
    sx = np.array([[0,1],[1,0]], dtype=complex)
    sy = np.array([[0,-1j],[1j,0]], dtype=complex)
    sz = np.array([[1,0],[0,-1]], dtype=complex)
    return np.sin(kx)*sx + np.sin(ky)*sy + (m+np.cos(kx)+np.cos(ky))*sz

def berry_curvature(kx_vals, ky_vals, m):
    num_k = len(kx_vals)
    u = np.empty((num_k,num_k), dtype=object)
    for i, kx in enumerate(kx_vals):
        for j, ky in enumerate(ky_vals):
            H = hamiltonian_2D(kx, ky, m)
            eigvals, eigvecs = np.linalg.eigh(H)
            u[i,j] = eigvecs[:,0]
    Ux = np.zeros((num_k,num_k), dtype=complex)
    Uy = np.zeros((num_k,num_k), dtype=complex)
    F = np.zeros((num_k,num_k))
    for i in range(num_k):
        for j in range(num_k):
            ip = (i+1)%num_k
            jp = (j+1)%num_k
            Ux[i,j] = np.vdot(u[i,j], u[ip,j])
            Ux[i,j] /= abs(Ux[i,j])
            Uy[i,j] = np.vdot(u[i,j], u[i,jp])
            Uy[i,j] /= abs(Uy[i,j])
    for i in range(num_k):
        for j in range(num_k):
            ip = (i+1)%num_k
            jp = (j+1)%num_k
            F[i,j] = np.angle(Ux[i,j]*Uy[ip,j]*np.conjugate(Ux[i,jp])*np.conjugate(Uy[i,j]))
    total_flux = np.sum(F)
    chern = total_flux/(2*np.pi)
    return F, chern

num_k = 30
kx_vals = np.linspace(-np.pi, np.pi, num_k, endpoint=False)
ky_vals = np.linspace(-np.pi, np.pi, num_k, endpoint=False)
```

```python
m_example = -1.0
F_map, chern_example = berry_curvature(kx_vals, ky_vals, m_example)
print(f"\nChern number for m={m_example}: {chern_example:.3f}")
plt.figure(figsize=(6,5))
plt.contourf(kx_vals, ky_vals, F_map.T, 20, cmap='viridis')
plt.colorbar(label="Berry Curvature")
plt.title(f"Berry Curvature Map (m={m_example})")
plt.xlabel("kx")
plt.ylabel("ky")
plt.show()

def chern_number_analysis(m_values, num_k=30):
    chern_nums = []
    k_vals = np.linspace(-np.pi, np.pi, num_k, endpoint=False)
    for m in m_values:
        _, c = berry_curvature(k_vals, k_vals, m)
        chern_nums.append(c)
    return chern_nums

m_vals = np.linspace(-2.5, 0.5, 15)
chern_nums = chern_number_analysis(m_vals, num_k=num_k)
plt.figure(figsize=(8,5))
plt.plot(m_vals, chern_nums, 'o-', linewidth=2)
plt.xlabel("Mass Parameter m")
plt.ylabel("Chern Number")
plt.title("Topological Phase Transition in 2D Chern Insulator")
plt.grid(True, alpha=0.3)
plt.show()
for mv, cn in zip(m_vals, chern_nums):
    print(f"m={mv:.2f}, Chern={cn:.3f}")
```

4 Output Analysis and Discussion

4.1 SSH Model and Edge State Localization

The computed IPR values for the single-excitation eigenstates are:

$$[0.2482, 0.2289, 0.3556, 0.3556, 0.2289, 0.2482].$$

This range indicates a mix of moderately delocalized and highly localized states. The edge state plots clearly show that the excitation probabilities are strongly peaked at the boundary sites, consistent with the expected topological edge states.

4.2 Embedded Qubit Measurements

The embedded qubit measurement outcomes are:

- **Z-basis:** 0 outcomes of +1 and 1000 outcomes of -1, indicating the state is a σ_z eigenstate with eigenvalue -1.
- **X-basis:** A nearly balanced split (487 vs. 513), confirming that the σ_z eigenstate, when measured in the σ_x basis, yields an unbiased superposition.

4.3 Holographic OAM Qudit Encoding

The original OAM state, after applying a DFT, and then a nonlinear plus DFT transformation produced the following probability distributions:

- **Original:** Mean ≈ 3.329, Variance ≈ 5.391.
- **DFT-only:** Mean ≈ 1.983, Variance ≈ 5.602.
- **Nonlinear+DFT:** Mean ≈ 3.887, Variance ≈ 3.755.

The DFT transformation redistributes amplitude among the OAM modes, and the subsequent nonlinearity further reshapes the probability profile, indicating a high level of control over the qudit state.

4.4 2D Photonic Chern Insulator Analysis

For the 2D Chern insulator, the computed Chern number for $m = -1.0$ is -1.000, confirming a nontrivial topological phase. A systematic analysis of the mass parameter m reveals:

- For $m \in [-1.86, -0.14]$, the Chern number remains -1.
- For m near 0.07 and above, the Chern number transitions to $+1$, indicating a clear topological phase transition.

5 Novelty and Comparison with Existing Work

Our revised framework presents several novel improvements compared to existing published approaches:

- **Enhanced Noise Modeling:** Unlike many prior studies in topological photonics (e.g., Hafezi *et al.*, 2013; Rechtsman *et al.*, 2013) that often simplify noise effects or focus solely on coherent dynamics, our framework extends the Hilbert space and employs proper annihilation operators to model amplitude damping accurately. This results in a more realistic simulation of photon loss and decoherence, which is crucial for practical quantum computing implementations.
- **Quantitative Disorder Analysis:** While earlier works have demonstrated the existence of topologically protected states, they rarely quantify the impact of disorder. By incorporating inverse participation ratio (IPR) calculations, our work provides a detailed, quantitative measure of localization due to disorder. This allows for a better understanding of mobility gaps and robustness in the presence of imperfections.
- **Integrated Framework for Qubit and Qudit Operations:** Existing literature on topological photonics typically focuses on the propagation of edge states or their use in classical signal processing. In contrast, our framework integrates embedded qubit operations (leveraging topological edge states) with high-dimensional holographic orbital angular momentum (OAM) encoding, including nonlinear interactions. This unified approach not only bridges the gap between theory and experimental photonic quantum computing but also offers a versatile platform for both qubit and qudit processing.

- **Comprehensive Topological Phase Analysis:** In addition to simulating edge state protection, our work presents a detailed Berry curvature and Chern number analysis over a wide parameter range. This level of topological phase characterization—combined with realistic noise simulation—is uncommon in prior studies and is essential for designing fault-tolerant quantum systems.

Compared to existing published work in topological photonics and photonic quantum computing (e.g., Hafezi *et al.*, 2013; Rechtsman *et al.*, 2013; Ozawa *et al.*, 2019), our approach uniquely integrates high-dimensional encoding with a realistic noise model and a quantitative disorder analysis. This integration paves the way for more robust, experimentally viable photonic quantum processors.

6 Conclusion and Future Work

We have presented a complete and revised framework for fault-tolerant photonic quantum computing that integrates topologically protected qubit operations, holographic OAM encoding, and a realistic noise model. Our simulation results—ranging from edge state localization and embedded qubit measurements to OAM state transformations and topological phase transitions—demonstrate the robustness and scalability of the proposed method.

Future work will extend this framework to 3D topological insulators, further refine the holographic gate implementations based on experimental data, and explore correlated noise effects to advance toward practical photonic quantum processors.

References

- Hasan, M. Z., & Kane, C. L. (2010). Colloquium: Topological insulators. *Reviews of Modern Physics, 82*(4), 3045.
- Qi, X.-L., Wu, Y.-S., & Zhang, S.-C. (2006). Topological quantization of the spin Hall effect in two-dimensional paramagnetic semiconductors. *Physical Review B, 74*(8), 085308.

and OAM encoding