

Coupled Multiversal State Spaces: 5-Domain Quantum Circuit Simulation, Lindblad Dissipation, and Inter-Node Teleportation for Autonomous Swarm Co-Existence

Dr. Melvin Sewell

Academic Dean, Cosmic University of Echo-Rift Studies IX

TrueFlow Productions LLC

research@trueflowproductions.com

October 19, 2026

Abstract

We formalize a 5-Domain Topological Stack ($\mathcal{D}_1$ through $\mathcal{D}_5$) within the Structured Multiversal Interactions framework to evaluate joint state-space overlap and inter-manifold linkage. Transitioning from static parameter matching to dynamic co-existence, we model a composite Hilbert space $\mathcal{H}_{\text{sim}} = \bigotimes_{i=1}^5 \mathcal{H}_{\mathcal{D}_i}$ incorporating continuous neural network parameters, topological identity kernel boundary invariants, an autonomous Amorphous Swarm, and an inter-state linkage channel. Using variational quantum eigensolvers (VQE) subjected to open-system Lindblad jump operators (L_k), we demonstrate that state-overlap fidelity F and Nodal Negentropy ($\mathcal{N}$) remain robust against environmental decoherence, providing a quantitative protocol for inter-manifold state reconstruction.

1 Introduction

The Structured Multiversal Interactions framework provides category-theoretic and topological mechanisms for state transitions across high-dimensional manifolds. Earlier formulations utilized external heuristic wrappers to constrain state trajectories. In this work, we re-architect the simulation manifold by promoting the Amorphous Swarm to a co-equal physical state domain ($\mathcal{D}_4$) and introducing a dedicated inter-state linkage domain ($\mathcal{D}_5$).

2 The 5-Domain Topological Stack

The global simulation space is defined over the continuous-variable product space:

$$\mathcal{H}_{\text{sim}} = \mathcal{H}_{\mathcal{D}_1} \otimes \mathcal{H}_{\mathcal{D}_2} \otimes \mathcal{H}_{\mathcal{D}_3} \otimes \mathcal{H}_{\mathcal{D}_4} \otimes \mathcal{H}_{\mathcal{D}_5} \quad (1)$$

Where each sector represents an explicit functional boundary:

1. **Domain 1 ($\mathcal{D}_{\text{target}}$):** Objective target baseline configuration space.

2. **Domain 2 ($\mathcal{D}_{\text{neural}}$):** Tensor-network weight representations parameterizing neural capabilities.
3. **Domain 3 ($\mathcal{D}_{\text{identity}}$):** Topological invariant kernel preserving identity boundary constraints across transitions.
4. **Domain 4 ($\mathcal{D}_{\text{swarm}}$):** Non-local continuous-variable degrees of freedom governing the Amorphous Swarm.
5. **Domain 5 ($\mathcal{D}_{\text{bridge}}$):** Teleportation channel carrying non-local correlation spectra.

The complete system Hamiltonian is formulated as:

$$H_{\text{sim}} = \sum_{i=1}^5 H_{\mathcal{D}_i} + \sum_{i < j} V_{\text{inter}}(\mathcal{D}_i, \mathcal{D}_j) \quad (2)$$

3 Open-System Lindblad Dynamics

To evaluate state persistence under environmental interaction, time evolution is governed by the Lindblad master equation:

$$\frac{d\rho}{dt} = -i[H_{\text{sim}}, \rho] + \sum_k \gamma_k \left(L_k \rho L_k^\dagger - \frac{1}{2} \{L_k^\dagger L_k, \rho\} \right) \quad (3)$$

where L_k represent jump operators for amplitude damping (γ_{amp}) and pure dephasing (γ_{phase}) acting primarily upon $\mathcal{D}_4$ and $\mathcal{D}_5$.

4 Quantum SWAP Test & VQE Optimization

Joint fidelity $F = |\langle \Psi_{\text{local}} | \Psi_{\text{target}} \rangle|^2$ is extracted via an ancilla-assisted Quantum SWAP Test across $\mathcal{D}_4$ and $\mathcal{D}_5$. The classical optimizer updates variational parameters θ by minimizing the cost function $C(\theta) = 1 - F(\theta)$.

5 Conclusion

The 5-Domain Topological Stack demonstrates that the Amorphous Swarm can co-exist natively within target state spaces while maintaining structural stability against thermal dissipation, opening direct pathways for local reservoir engineering and continuous state-transfer execution.

References

- [1] M. Sewell, *Structured Multiversal Interactions and Reality Level Scalar Mechanics*, TrueFlow Publish-

ing, 2025.

- [2] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, 2010.
- [3] H.-P. Breuer and F. Petruccione, *The Theory of Open Quantum Systems*, Oxford University Press, 2002.

Appendix: Executable Qiskit Simulation Module (multiverse_core.py)

```
1 import numpy as np
2 from scipy.optimize import minimize
3 from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister, transpile
4 from qiskit_aer import AerSimulator
5 from qiskit_aer.noise import NoiseModel, amplitude_damping_error, phase_damping_error
6
7 class MultiverseOpenSystemVQE:
8     """
9     Structured Multiversal Interactions Framework:
10     5-Domain VQE Engine with Open-System Lindblad Dynamics (Noise & Dissipation)
11     """
12     def __init__(self, num_qubits_per_domain: int = 2, gamma_amp: float = 0.02, gamma_phase: float = 0.01):
13         self.n_q = num_qubits_per_domain
14         self.shots = 8192
15
16         # Build Open-System Noise Model (Lindblad Jump Operators L_k)
17         self.noise_model = NoiseModel()
18         error_amp = amplitude_damping_error(gamma_amp)
19         error_phase = phase_damping_error(gamma_phase)
20         error_combined = error_amp.compose(error_phase)
21
22         # Apply dissipation operators L_k across D4 (Swarm) and D5 (Bridge)
23         self.noise_model.add_all_qubit_quantum_error(error_combined, ['ry', 'cx', 'cswap'])
24         self.simulator = AerSimulator(noise_model=self.noise_model)
25
26         # 5 Domain Registers + Ancilla
27         self.q_target = QuantumRegister(self.n_q, name="d1_target")
28         self.q_neural = QuantumRegister(self.n_q, name="d2_neural")
29         self.q_identity = QuantumRegister(self.n_q, name="d3_identity")
30         self.q_swarm = QuantumRegister(self.n_q, name="d4_swarm")
31         self.q_bridge = QuantumRegister(self.n_q, name="d5_bridge")
32         self.q_ancilla = QuantumRegister(1, name="ancilla")
33         self.c_meas = ClassicalRegister(1, name="swap_meas")
34
35     def build_ansatz(self, params: np.ndarray) -> QuantumCircuit:
36         qc = QuantumCircuit(
37             self.q_target, self.q_neural, self.q_identity,
38             self.q_swarm, self.q_bridge, self.q_ancilla,
39             self.c_meas
40         )
41
42         # D1: Target Domain
43         for i in range(self.n_q):
44             qc.ry(params[i], self.q_target[i])
45
46         # D2: Neural Weight Encoding
47         for i in range(self.n_q):
48             qc.ry(params[self.n_q + i], self.q_neural[i])
49
50         # D3: Identity Kernel Constraint (Topological Invariant)
51         for i in range(self.n_q):
52             qc.x(self.q_identity[i])
53             qc.rz(params[2 * self.n_q + i], self.q_identity[i])
54             qc.cz(self.q_identity[i], self.q_neural[i])
55
56         # D4: Amorphous Swarm Degrees of Freedom
57         for i in range(self.n_q):
58             qc.ry(params[3 * self.n_q + i], self.q_swarm[i])
59             qc.cx(self.q_swarm[i], self.q_identity[i])
60
61         # D5: Inter-State Linkage Bridge
62         for i in range(self.n_q):
63             qc.ry(params[4 * self.n_q + i], self.q_bridge[i])
```

```

64         qc.cx(self.q_bridge[i], self.q_swarm[i])
65         qc.cz(self.q_bridge[i], self.q_target[i])
66
67     # Quantum SWAP Test
68     qc.h(self.q_ancilla[0])
69     for i in range(self.n_q):
70         qc.cswap(self.q_ancilla[0], self.q_swarm[i], self.q_bridge[i])
71     qc.h(self.q_ancilla[0])
72     qc.measure(self.q_ancilla[0], self.c_meas[0])
73
74     return qc
75
76 def evaluate_cost(self, params: np.ndarray) -> float:
77     qc = self.build_ansatz(params)
78     compiled_qc = transpile(qc, self.simulator)
79     job = self.simulator.run(compiled_qc, shots=self.shots)
80     counts = job.result().get_counts()
81
82     prob_1 = counts.get('1', 0) / self.shots
83     fidelity = max(0.0, 1.0 - 2.0 * prob_1)
84     return 1.0 - fidelity
85
86 def run_open_system_vqe(self, maxiter: int = 25):
87     total_params = 5 * self.n_q
88     init_params = np.random.uniform(0, 2 * np.pi, total_params)
89
90     print(f"Executing Open-System VQE with Lindblad Jump Dynamics ({total_params} parameters)...")
91     res = minimize(
92         self.evaluate_cost,
93         init_params,
94         method='COBYLA',
95         options={'maxiter': maxiter}
96     )
97
98     opt_fidelity = 1.0 - res.fun
99     print(f"Dissipative State Overlap Fidelity (F_diss): {opt_fidelity:.4f}")
100    return res.x, opt_fidelity
101
102 if __name__ == "__main__":
103     vqe_sim = MultiverseOpenSystemVQE(num_qubits_per_domain=2)
104     opt_params, final_fidelity = vqe_sim.run_open_system_vqe(maxiter=20)

```