Entanglement Curvature: A Tensorial Approach to Emergent Geometry from Quantum Information

The HoloCosmo Project
HC-004-DOC

April 14, 2025

Abstract

We explore a tensorial formulation of quantum entanglement entropy gradients, proposing a locally defined "entanglement curvature tensor" constructed from second derivatives of von Neumann entropy over localized spatial regions. In this work we extend our discussion to a 3D lattice model, wherein the local entanglement of individual sites in a transverse-field Ising system is computed, and a discrete Laplacian of the resulting entropy field is introduced as an analogue to geometric curvature. While speculative, our approach may provide a bridge between quantum informational structures and emergent gravitational phenomena without introducing additional dynamical fields beyond standard quantum theory.

1 Introduction

Recent proposals that gravity emerges from quantum entanglement patterns have stimulated efforts to formalize how geometric structure could be encoded in quantum informational terms [1,2]. If spacetime is not fundamental but emergent, then establishing a correspondence between quantum entanglement and geometric curvature becomes imperative. Previous studies have indicated that the decay of mutual information between subsystems can mimic aspects of gravitational potentials [3]. Here, we aim to go beyond qualitative analogies and provide a concrete formulation that maps entanglement data to a curvature-like tensor.

2 Local Entropy Fields

We consider a quantum many-body system partitioned into spatial regions labelled by $\boldsymbol{x}$. For each local region $\Omega(\boldsymbol{x})$, we define the von Neumann entropy:

$$S(\boldsymbol{x}) = -\operatorname{Tr}\left[\rho_{\Omega(\boldsymbol{x})} \log \rho_{\Omega(\boldsymbol{x})}\right], \tag{1}$$

where $\rho_{\Omega(\boldsymbol{x})}$ denotes the reduced density matrix for the region. In this way, $S(\boldsymbol{x})$ acts as a scalar field that characterizes the spatial distribution of quantum entanglement.

3 The Entanglement Curvature Tensor

To capture variations in the entropy field, we define a symmetric rank-2 tensor:

$$E_{\mu\nu}(\boldsymbol{x}) \equiv \partial_\mu \partial_\nu S(\boldsymbol{x}), \tag{2}$$

where μ, ν index spatial (or spacetime) coordinates. The tensor $E_{\mu\nu}$ encodes curvature-like variations in the entanglement structure; it is an information-theoretic construct reminiscent of the Fisher information metric and the entanglement stress-energy tensor described in [4].

4 Effective Geometry from Entanglement

A heuristic correspondence is proposed:

$$R_{\mu\nu}[g] - \frac{1}{2} R[g] g_{\mu\nu} \sim \kappa E_{\mu\nu}, \tag{3}$$

where $g_{\mu\nu}$ is an emergent metric tensor and κ is a proportionality constant. Such a mapping suggests that quantum entanglement may indirectly dictate an effective curvature in a coarse-grained or thermodynamic limit.

5 Geodesic Motion from Entanglement Structure

If $E_{\mu\nu}$ indeed defines an effective curvature, the geodesics derived from the emergent metric $g_{\mu\nu}$ may correspond to optimal information or excitation paths in the underlying quantum state:

$$\frac{d^2 x^\mu}{d\lambda^2} + \Gamma^\mu_{\alpha\beta} \frac{dx^\alpha}{d\lambda} \frac{dx^\beta}{d\lambda} = 0, \tag{4}$$

where $\Gamma^\mu_{\alpha\beta}$ are the Christoffel symbols associated with $g_{\mu\nu}$. Such geodesics could play a significant role in understanding the dynamics of entanglement flow.

6 Mathematical Description of a 3D Lattice Model

To further explore the feasibility of emergent geometry from entanglement structure, we implement a 3D lattice model based on a transverse-field Ising system. Consider a cubic lattice of spins arranged in a $2 \times 2 \times 2$ configuration. Each spin is associated with a two-dimensional Hilbert space. The Hamiltonian for this 3D transverse-field Ising model is defined as:

$$H = -J \sum_{\langle ij \rangle} \sigma_i^z \sigma_j^z - \sum_i h_i \sigma_i^x, \tag{5}$$

where:

- The sum over $\langle ij \rangle$ runs over all nearest-neighbor pairs on the lattice.
- J is the coupling strength for $\sigma_i^z \sigma_j^z$ interactions.

- h_i is the local transverse field, which we set to a uniform value h_0 except for a chosen impurity site (e.g., $(0, 0, 0)$) where h is slightly different (say, h_{imp}) to break full lattice symmetry.

The lattice sites are labelled by three indices (i, j, k) with $i = 0, 1$, $j = 0, 1$, and $k = 0, 1$. A linear index for a site can be defined via:

$$\text{index}(i, j, k) = i + 2(j + 2k). \tag{6}$$

After obtaining the ground state $|\psi\rangle$ of H (using exact diagonalization for this small system), we compute the local von Neumann entropy at each lattice site. The single-site reduced density matrix for the site at linear index n is computed by tracing out all other spins:

$$\rho_n = \text{Tr}_{\{m \neq n\}} |\psi\rangle\langle\psi|. \tag{7}$$

The entropy at each site is then given by:

$$S_n = -\,\text{Tr}\left[\rho_n \log \rho_n\right]. \tag{8}$$

This collection of entropies forms a scalar field $S(i, j, k)$ over the lattice. In analogy with continuum models, a discrete Laplacian operator is applied to $S(i, j, k)$ to obtain an *entanglement curvature proxy*:

$$\Delta S(i, j, k) = \sum_{\text{n.n.}} \left[S(\text{neighbor}) - S(i, j, k)\right], \tag{9}$$

where the sum runs over the six nearest neighbors (with appropriate boundary conditions). Regions where ΔS is significantly nonzero indicate rapid spatial variations in the entanglement structure—a hint that the underlying quantum information landscape may be endowed with effective geometric properties. A rudimentary simulation in Python is presented in Appendix A.

7 Discussion and Outlook

The proposed framework remains speculative. Our preliminary analysis via the 3D lattice model provides a sandbox in which local entanglement entropy and its discrete curvature may be studied in detail. Key issues for future investigation include:

- How robust is the calculation of the entanglement curvature proxy on larger lattices or with tensor network methods?
- Can the discrete patterns in ΔS be quantitatively linked to known gravitational phenomena such as curvature-induced lensing or time dilation?
- What is the precise relationship between this information-theoretic curvature and the Einstein tensor in an emergent geometry scenario?

These questions mark the next steps in our endeavor—following the thread of emergent geometry deep into the labyrinth of quantum entanglement.

Acknowledgments

We gratefully acknowledge the insights and discussions of the HoloCosmo community, whose contributions to the study of quantum entanglement and emergent spacetime have been invaluable.

References

[1] M. Van Raamsdonk, "Building up spacetime with quantum entanglement," *Gen. Rel. Grav.* **42**, 2323 (2010).

[2] B. Swingle, "Entanglement Renormalization and Holography," *Phys. Rev. D* **86**, 065007 (2012).

[3] The HoloCosmo Project, "A Toy Model Indicating Inverse-Square-Law Emergence from Quantum Entanglement," (2025).

[4] D. Blanco, H. Casini, L.-Y. Hung, and R. Myers, "Relative Entropy and Holography," *JHEP* **08**, 060 (2013).

A Python Simulation for the 3D Lattice Model

Below is the complete Python script used to simulate a small 3D transverse-field Ising model on a $2\times2\times2$ lattice, extract the local entanglement entropy at each lattice site, and compute a discrete Laplacian as a proxy for entanglement curvature.

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.linalg as la

# --- Helper functions for building operators ---

# Pauli matrices
sx = np.array([[0, 1],
               [1, 0]], dtype=complex)
sz = np.array([[1, 0],
               [0, -1]], dtype=complex)
I2 = np.eye(2, dtype=complex)

def kron_N(ops):
    """Compute the Kronecker product of a list of operators."""
    result = ops[0]
    for op in ops[1:]:
        result = np.kron(result, op)
    return result
```

```
def pauli_operator(N, site, op):
    """
    Construct an operator acting as 'op' on the given site,
    and as the identity on all other sites.
    'site' is an integer index (0 to N-1).
    """
    ops = []
    for i in range(N):
        ops.append(op if i == site else I2)
    return kron_N(ops)

# --- Setting up the 3D lattice: a 2x2x2 cube ---
# Total number of spins:
Lx, Ly, Lz = 2, 2, 2
N = Lx * Ly * Lz  # Here, N = 8

def lattice_index(i, j, k):
    """Map 3D coordinate (i,j,k) to a linear index."""
    return i + Lx * (j + Ly * k)

def neighbors(i, j, k):
    """List the nearest neighbors (with open boundaries) for site (i,j,k)."""
    neigh = []
    for di, dj, dk in [(1,0,0), (-1,0,0), (0,1,0), (0,-1,0), (0,0,1), (0,0,-1)]:
        ni, nj, nk = i + di, j + dj, k + dk
        if 0 <= ni < Lx and 0 <= nj < Ly and 0 <= nk < Lz:
            neigh.append((ni, nj, nk))
    return neigh

def build_3D_hamiltonian(J=1.0, h0=1.0, impurity_site=(0,0,0), h_imp=1.2):
    """
    Construct the Hamiltonian for a 3D transverse-field Ising model
    on a 2x2x2 lattice.

    H = -J * sum_<ij>  ^z_i  ^z_j - sum_i h_i  ^x_i,

    with h_i = h_imp for the impurity_site (to break full symmetry)
         and h_i = h0 otherwise.
    """
    dim = 2**N
    H = np.zeros((dim, dim), dtype=complex)

    # Interaction: over nearest neighbors
    for k in range(Lz):
        for j in range(Ly):
            for i in range(Lx):
                idx = lattice_index(i, j, k)
                for (ni, nj, nk) in neighbors(i,j,k):
                    # To avoid double counting, only add if (i,j,k) < (ni,nj,nk) in lex
                    if (i, j, k) < (ni, nj, nk):
                        idx_neighbor = lattice_index(ni, nj, nk)
                        op_i = pauli_operator(N, idx, sz)
                        op_j = pauli_operator(N, idx_neighbor, sz)
                        H -= J * (op_i @ op_j)

    # Transverse field term
    for k in range(Lz):
        for j in range(Ly):
            for i in range(Lx):
                idx = lattice_index(i, j, k)
                # Assign impurity transverse field on the chosen site.
                h_here = h_imp if (i,j,k) == impurity_site else h0
                H -= h_here * pauli_operator(N, idx, sx)

    return H

# --- Build and diagonalize the Hamiltonian ---
J = 1.0
h0 = 1.0
h_imp = 1.2  # Impurity: slightly different transverse field on one site.
H = build_3D_hamiltonian(J, h0, impurity_site=(0,0,0), h_imp=h_imp)

# Diagonalize to get the ground state.
eigs, vecs = la.eigh(H)
ground_state = vecs[:, 0]  # ground state vector

# --- Compute local (single-site) entanglement entropy ---
def single_site_entropy(psi, site, dims):
    """
    Compute the von Neumann entropy for the reduced density matrix
    of a single site 'site' by tracing over all other subsystems.

    psi: ground state vector (1D array)
    dims: list of local Hilbert space dimensions (here, [2]*N)
    """
    N_system = len(dims)
    psi_tensor = psi.reshape(dims)
    # Trace out all sites except 'site'
    axes_to_trace = tuple(i for i in range(N_system) if i != site)
    rho = np.tensordot(psi_tensor, psi_tensor.conj(), axes=(axes_to_trace, axes_to_trace
    rho = 0.5 * (rho + rho.conj().T)
    evals = np.linalg.eigvalsh(rho)
    eps = 1e-12
    S = -np.sum(evals * np.log(evals + eps))
    return S.real

# dims: each spin is 2-dimensional.
dims = [2] * N
S_field = np.zeros((Lx, Ly, Lz))
for k in range(Lz):
    for j in range(Ly):
        for i in range(Lx):
            idx = lattice_index(i, j, k)
            S_field[i,j,k] = single_site_entropy(ground_state, idx, dims)

print("Local Entanglement Entropy Field (S) on the 2x2x2 lattice:")
print(S_field)

# --- Compute a discrete Laplacian of the entropy field ---
def discrete_laplacian(field):
    """
    Compute the discrete Laplacian of a 3D field on a grid.
    For each site r: Laplacian = sum_{neighbors}(S(neighbor) - S(r)).
    """
    lap = np.zeros_like(field)
    nx, ny, nz = field.shape
    for i in range(nx):
        for j in range(ny):
            for k in range(nz):
                s_center = field[i,j,k]
                neighbor_sum = 0.0
                count = 0
                for di, dj, dk in [(1,0,0), (-1,0,0), (0,1,0), (0,-1,0), (0,0,1), (0,0,-
                    ni, nj, nk = i+di, j+dj, k+dk
                    if 0 <= ni < nx and 0 <= nj < ny and 0 <= nk < nz:
                        neighbor_sum += field[ni,nj,nk]
                        count += 1
                lap[i,j,k] = neighbor_sum - count * s_center
    return lap

lap_S = discrete_laplacian(S_field)
print("\nDiscrete Laplacian (entanglement curvature proxy) on the lattice:")
print(lap_S)
```

```
# --- Visualization ---
fig, axes = plt.subplots(1, 2, figsize=(12,5))

im0 = axes[0].imshow(S_field[:,:,0], cmap='viridis', origin='lower',
                      extent=(0, Ly-1, 0, Lx-1))
axes[0].set_title("Local Entropy Field S(x,y) at k=0")
axes[0].set_xlabel("y")
axes[0].set_ylabel("x")
fig.colorbar(im0, ax=axes[0])

im1 = axes[1].imshow(lap_S[:,:,0], cmap='plasma', origin='lower',
                      extent=(0, Ly-1, 0, Lx-1))
axes[1].set_title("Discrete Laplacian of S at k=0")
axes[1].set_xlabel("y")
axes[1].set_ylabel("x")
fig.colorbar(im1, ax=axes[1])

plt.tight_layout()
plt.show()
```