SERC as a Relational Equilibrium Framework for Large-Scale AI Systems

From Simplex Geometry and the Zero Point P_0 to Energy-Efficient and Stable Inference

Leszek Papiernik

February 2026

Abstract

We introduce SERC (Structure–Energy–Resonance–Coherence) as a simplex-based relational metric for modeling internal equilibrium and emergent stability in complex computational systems, including large-scale AI models. The framework is defined on a regular 3-simplex with a canonical Gram metric $G = 4I - J$, inducing a coherence functional $Q(Z) = \frac{1}{2} Z^\top G Z$ that measures relational tension between components. Its unique global minimum under the barycentric constraint $S + E + R + C = 1$ is attained at the Zero Point $P_0 = (1/4, 1/4, 1/4, 1/4)$.

Building on this structure, we interpret internal time as relational path length and analyze its vanishing near P_0. We map SERC coordinates to operational quantities in large-scale AI systems and introduce the notion of relational friction as accumulated relational change during inference. Minimal simulations demonstrate that maintaining proximity to P_0 reduces relational tension and internal time, suggesting a potential route toward more stable and energy-efficient inference. We discuss implications for AI inference, hardware efficiency, and future research directions.

1 Introduction

Large-scale artificial intelligence systems operate in high-dimensional internal spaces whose structure is only partially understood. Despite their impressive performance, these systems exhibit instability, drift, and high energy consumption. Current approaches address these issues locally but lack a unifying geometric framework.

We propose that the internal dynamics of AI models can be analyzed through the SERC relational metric. The SERC state

$$Z = (S, E, R, C), \qquad S + E + R + C = 1,$$

is embedded in a regular 3-simplex with Gram matrix $G = 4I - J$. The coherence functional

$$Q(Z) = \tfrac{1}{2} Z^\top G Z$$

measures relational tension and has a unique global minimum at

$$P_0 = (1/4, 1/4, 1/4, 1/4).$$

This paper connects the SERC framework with practical challenges in AI inference, proposing that maintaining proximity to P_0 may reduce computational overhead, stabilize internal dynamics, and improve energy efficiency.

2 The SERC Framework

2.1 State Space and Barycentric Constraint

The SERC state is a four-component relational vector

$$Z = (S, E, R, C) \in \mathbb{R}^4_{\geq 0},$$

subject to the barycentric constraint

$$S + E + R + C = 1.$$

This embeds the state space into a regular 3-simplex.

2.2 Simplex Geometry and Gram Metric

The axes are not orthogonal; the angle between any two satisfies $\cos\theta = -\frac{1}{3}$. The inner product is defined by

$$G = 4I - J,$$

with null eigenvector $(1, 1, 1, 1)$ corresponding to uniform scaling.

2.3 Coherence Functional and Zero Point

The coherence functional is

$$Q(Z) = \tfrac{1}{2} Z^\top G Z = \tfrac{1}{2} \sum_{i,j} (Z_i - Z_j)^2.$$

It vanishes only when all components are equal. Under the barycentric constraint, the unique global minimum is

$$P_0 = (1/4, 1/4, 1/4, 1/4).$$

3 Relational Time and Internal Dynamics

3.1 Relational Evolution

A trajectory $Z(t)$ evolves within the simplex, with t an arbitrary parameter.

3.2 Relational Velocity

Velocity is defined as

$$v(t) = \sqrt{\dot{Z}^\top G \dot{Z}}.$$

3.3 Internal Time

Internal time is the raelational path length:

$$T = \int v(t)\, dt.$$

3.4 Vanishing of Time Near P_0 to Energy-Efficient and Stable Inference

As $Z(t) \to P_0$, we have $\dot{Z} \to 0$ and thus $v(t) \to 0$, making internal time locally irrelevant.

4 Mapping SERC to AI Systems

4.1 Operational Interpretation

We propose:

- S: structural complexity (entropy, sparsity),
- E: activation energy ($\|h\|^2$),
- R: resonance (alignment, correlation),
- C: coherence (smoothness, consistency).

4.2 Relational Friction

Relational friction is cumulative internal time:

$$T = \sum_n \sqrt{(Z_{n+1} - Z_n)^\top G (Z_{n+1} - Z_n)}.$$

5 Minimal Simulations

We simulate discrete dynamics:

$$Z_{n+1} = Z_n + \alpha X_n - \beta G Z_n,$$

with projection onto the simplex.

Two modes:

- **A**: no reset,
- **B**: reset to P_0 after each step.

Mode B yields lower $Q(Z)$, shorter T, and greater stability.

6 Implications for AI Inference and Hardware

Maintaining proximity to P_0 may:

- reduce computational overhead,
- stabilize attention and activations,
- lower energy consumption,
- reduce thermal stress,
- extend hardware lifespan.

7 Discussion and Future Work

Future directions include:

- empirical evaluation on real LLMs,
- controlled resets toward P_0,
- hardware-level integration,
- extension to multi-modal and multi-agent systems.

8 Conclusions

SERC provides a unified relational geometry for analyzing internal dynamics in AI systems. The Zero Point P_0 represents a natural equilibrium configuration. Minimal simulations suggest that maintaining proximity to P_0 may improve stability and efficiency. This motivates empirical testing on real AI models.

Appendix A: Minimal SERC Simulation Code

This appendix provides a reference implementation of the minimal SERC simulation described in Section 5. The code is written in Python and requires only `numpy`. It implements the SERC metric, the coherence functional, barycentric projection, discrete relational dynamics, and two operational modes (with and without reset to P_0).

A.1 Python Reference Implementation

```
import numpy as np

# --- SERC DEFINITIONS ----------------------------------------------------

# Gram matrix G = 4I - J
I = np.eye(4)
J = np.ones((4,4))
G = 4*I - J

# Zero Point P0
P0 = np.array([0.25, 0.25, 0.25, 0.25])

def barycentric_project(Z):
    """Project onto simplex S+E+R+C=1 with non-negativity."""
    Z = np.maximum(Z, 0)
    s = np.sum(Z)
    if s == 0:
        return P0.copy()
    return Z / s

def Q(Z):
    """Coherence functional Q(Z) = 1/2 Z^T G Z."""
    return 0.5 * Z @ G @ Z

def grad_Q(Z):
    """Gradient of Q: nabla Q = G Z."""
    return G @ Z

# --- DYNAMICS ------------------------------------------------------------

def update(Z, input_vector, alpha=0.1, beta=0.05):
    """
    One iteration of relational dynamics:
    - input_vector: external perturbation
    - alpha: input strength
    - beta: relaxation rate (gradient descent on Q)
    """
    # external influence
```

```
    Z = Z + alpha * input_vector

    # relaxation toward P0
    Z = Z - beta * grad_Q(Z)

    # enforce barycentric constraint
    Z = barycentric_project(Z)
    return Z

# --- SIMULATION ------------------------------------------------------------

def simulate(steps=200, reset=False):
    Z = np.random.rand(4)
    Z = barycentric_project(Z)

    trajectory = []
    Q_values = []
    T = 0.0  # relational time

    for _ in range(steps):
        # random input (simulated query)
        inp = np.random.rand(4)
        inp = barycentric_project(inp)

        # update
        Z_new = update(Z, inp)

        # relational velocity
        dZ = Z_new - Z
        v = np.sqrt(dZ @ G @ dZ)
        T += v

        Z = Z_new

        # optional reset to P0
        if reset:
            Z = P0.copy()

        trajectory.append(Z.copy())
        Q_values.append(Q(Z))
```

```
    return np.array(trajectory), np.array(Q_values), T

# --- RUN EXAMPLE -------------------------------------------------------------

traj_A, Q_A, T_A = simulate(reset=False)
traj_B, Q_B, T_B = simulate(reset=True)

print("Mode A (no reset):")
print("  Mean Q =", np.mean(Q_A))
print("  Relational time T =", T_A)

print("\nMode B (reset to P0):")
print("  Mean Q =", np.mean(Q_B))
print("  Relational time T =", T_B)
```

A.2 Parameter Description

- α controls the strength of external perturbations (analogous to input-driven activation changes).
- β controls the relaxation rate toward equilibrium (gradient descent on Q).
- **reset=True** enforces $Z \leftarrow P_0$ after each step.
- **Q(Z)** measures relational tension.
- **T** accumulates relational path length (internal time).

A.3 Reproducing the Results

To reproduce the qualitative results in Section 5:

1. Run the script twice: once with `reset=False`, once with `reset=True`.
2. Compare:
    - mean coherence tension $\langle Q \rangle$,
    - total relational time T,
    - stability of trajectories.
3. Mode B should consistently yield:
    - lower $Q(Z)$,
    - shorter T,
    - more stable trajectories.

A.4 Optional Visualization Code

```
import matplotlib.pyplot as plt

plt.plot(Q_A, label="No reset")
plt.plot(Q_B, label="Reset to P0")
plt.xlabel("Step")
plt.ylabel("Q(Z)")
plt.legend()
plt.show()
```

This visualization highlights the stabilizing effect of resetting to P_0.

A.5 Implementation Notes

- The simulation is intentionally minimal and does not model high-dimensional neural activations.
- The purpose is to illustrate geometric properties of the SERC metric.
- The same relational metrics can be applied to real AI models by mapping activations to (S, E, R, C).

Appendix B: Figures

B.1 Regular 3-Simplex with SERC Axes

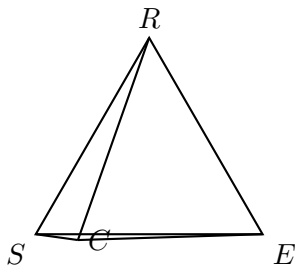

Figure 1: Regular 3-simplex representing the SERC state space. Each vertex corresponds to dominance of a single component.

B.2 Zero Point P_0 as the Barycentric Center

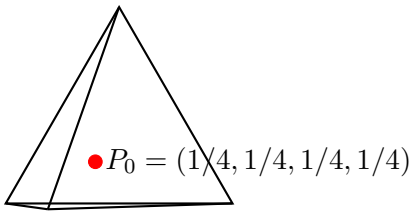

Figure 2: The Zero Point P_0 as the barycentric center of the simplex. It is the unique global minimum of the coherence functional $Q(Z)$.

B.3 Example Relational Trajectory and Relaxation Toward P_0

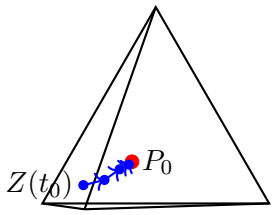

Figure 3: Example relational trajectory $Z(t)$ approaching the Zero Point P_0. The path length corresponds to internal time T.

References

[1] L. Papiernik, *SERC Unified Relational Simplex Metric*, 2026.