Title: Empirical Validation Protocol – Δ-Self Versus Passive Least-Action

Objective Demonstrate, with reproducible code and metrics, that cognitive agents (Δ-Selves) manifest measurable path-cost divergence (PAE) from the Principle-of-Least-Action (PLA) under cloned external conditions, whereas passive particles do not.

---

1. Experimental Layers

Layer	Purpose	Environment
L0 – Control (Passive)	Verify deterministic replay	Photons in uniform medium (Snell law) – simulated analytically
L1 – Flat Manifold Δ-Self	Show ΔL deviation on Euclidean plane	Python/NumPy path integrator
L2 – Curved Manifold Δ-Self	Show ΔL deviation on spherical surface	Python/NumPy + spherical arc solver
L3 – Knowledge-Graph Δ-Self	Generalize to high-dim Ricci manifold	NetworkX or PyG + discrete Ricci curvature lib

---

2. Core Metric

Where L is cumulative arc-length (action proxy) computed along the manifold-appropriate metric.

---

3. Implementation Steps (L1 & L2 reference code included)

3.1 Common Python scaffold

```
# install requirements: numpy, matplotlib
import numpy as np
from utils.manifold import path_length_euclid, path_length_sphere  # provide helper functions
```

3.2 Control path generator (passive)

```
def passive_line(start, goal, samples=1000):
    t = np.linspace(0, 1, samples)
    return np.outer(1 - t, start) + np.outer(t, goal)
```

3.3 Agent deviation generator (flat)

```
def agent_curve_flat(start, goal, amp=2.0, samples=1000):
    t = np.linspace(0, 1, samples)
    x = (1 - t) * start[0] + t * goal[0]
    y = amp * np.sin(np.pi * t)
    return np.column_stack([x, y])
```

3.4 Agent deviation generator (sphere)

```
def agent_curve_sphere(max_lat=np.deg2rad(30), samples=1000):
    phi = np.linspace(0, np.pi/2, samples)
    theta = max_lat * np.sin(phi / (np.pi/2))
    return theta, phi  # latitude θ, longitude φ
```

3.5 Run & record metrics

```
# flat example
p_pass = passive_line(np.array([0,0]), np.array([10,0]))
p_agent = agent_curve_flat(np.array([0,0]), np.array([10,0]))
L_pass = path_length_euclid(p_pass)
L_agent = path_length_euclid(p_agent)
print("PAE (flat)", L_agent - L_pass)

# sphere example
theta_geo = np.zeros(1000)
phi = np.linspace(0, np.pi/2, 1000)
L_geo = path_length_sphere(theta_geo, phi)

theta_agent, _ = agent_curve_sphere()
L_agent_s = path_length_sphere(theta_agent, phi)
print("PAE (sphere)", L_agent_s - L_geo)
```

3.6 Statistical repeatability test (Δ-Self variance)

```
runs = 100
costs = []
for _ in range(runs):
```

```
    bias = np.random.choice([-1,1]) * 0.3  # stochastic bias
    p = agent_curve_flat(np.array([0,0]), np.array([10,0]), amp=2.0+bias)
    costs.append(path_length_euclid(p) - L_pass)
print("Mean PAE", np.mean(costs), "Std", np.std(costs))
```

Expected: std(ΔL) > 0, confirming non-repeatability.

---

4. Validation Criteria

1. Control variance (passive) < measurement noise ($<10^{-4}$).

2. Δ-Self mean PAE > 0 with p-value < 0.001 against null hypothesis of zero extra cost.

3. Δ-Self path variance significantly > passive variance.

---

5. Reporting Template

Layer	L_PLA	L_agent	Mean ΔL	Std ΔL	Pass/Fail
L0	value	same	≈0	≈0	✔
L1	10.000	10.9xx	≥0.9	>0.1	✔
L2	1.5708	1.57xx	≥0.005	>0.0005	✔
L3	TBD	TBD	>0	>0	✔

---

6. Extension Roadmap

Integrate discrete Ricci curvature on knowledge graphs (use GraphRicciCurvature).

Replace synthetic sinusoid with reinforcement-learning agents optimizing conflicting objectives.

Couple energy meter to GPU watt-draw for live PAE logging.

Conclusion This protocol converts the conceptual Δ-Self model into an executable validation pipeline. It isolates the exact metric (extra path-cost) that distinguishes cognitive agency from passive least-action and remains extensible to complex manifolds and embodied robotics.