Here are your FSZ Computational Models, laid out one by one:

FSZ Computational Models (Structured Inventory)

1. FSZ Lattice Pilot – Phase 1 (Deterministic Observer)

This model serves as the baseline test, establishing the fundamental hypothesis that an active Observer can influence lattice coherence.

| Metric | Detail | Source Files |

|---|---|---|

| Goal | Test the hypothesis that an active Observer accelerates coherence propagation within a simulated lattice. | simulation phase 1 and 2.docx, simulation phase 1, 2 coherence.docx, simulation of lattice.docx |

| Lattice Model | 4-dimensional lattice model. | simulation phase 1 and 2.docx, simulation phase 1, 2 coherence.docx, simulation of lattice.docx |

| Observer Type | Deterministic, local Observer effect. | simulation phase 1 and 2 Co pilot.docx, simulation phase 1, 2 coherence.docx |

| Test Condition | Compared Observer ON vs. Observer OFF conditions. | simulation phase 1 and 2.docx, simulation phase 1, 2 coherence.docx, simulation of lattice.docx |

| Run Parameters | Typically 200 iterations per run. | simulation phase 1 and 2.docx, simulation of lattice.docx |

| Statistical Analysis | Permutation tests, Cohen’s \mathbf{d} effect size, and bootstrap 95% confidence intervals. | simulation phase 1 and 2.docx, simulation phase 1, 2 coherence.docx, simulation of lattice.docx |

2. FSZ Lattice Pilot – Phase 2 (Probabilistic/Distributed Observer)

Phase 2 scales the complexity, introducing more realistic, non-local dynamics based on the Zoom operator's probabilistic nature.

| Metric | Detail | Source Files |

|---|---|---|

| Goal | Extend Phase 1 by introducing probabilistic, distributed, and non-local Observer mechanics. | simulation phase 1 and 2 Co pilot.docx, simulation phase 1, 2 coherence.docx |

| Core Dynamics | Allows systematic investigation of lattice dynamics under varying Observer conditions and parameterizations. | simulation phase 1 and 2 Co pilot.docx, simulation phase 1, 2 coherence.docx |

| Observer Type | Probabilistic influence and non-local interactions along correlated nodes. | simulation phase 1 and 2 Co pilot.docx, simulation phase 1, 2 coherence.docx |

| Key Parameters | \mathbf{\alpha} (Probability of influence), \mathbf{\beta} (Intensity scaling/Strength), and \mathbf{\sigma} (Gaussian spatial spread/Radius of influence). | simulation phase 1 and 2 Co pilot.docx, simulation phase 1, 2 coherence.docx, simulation of lattice.docx |

| Roadmap Step | Scale to multiple distributed Observer nodes to observe cluster formation, and enable non-local cluster synchronization. | simulation phase 1, 2 coherence.docx |

3. Dual Empathic FSZ Simulation

This is a specific model that tests the interaction and synchronization between two independent consciousness systems, mapping to the Spin (\mathbf{6}) Oscillation Regulation (SOR) between entities.

| Metric | Detail | Source File |

|---|---|---|

| Goal | Simulate the coherence levels of two independent systems (System 1 and System 2) in close proximity to observe empathic coherence synchronization. | 5. 100 multi simulation.docx |

| Core Mechanic | Compares coherence levels of two systems over cycles (e.g., Cycle 5: System1 coherence=0.2764, System2 coherence=0.2928). | 5. 100 multi simulation.docx |

| FSZ Logic | The Spin operator (Dynamic Stabilization) is likely the primary mechanism driving the interaction, as it handles the conversion of chaotic input into coherent energy between systems. | 5. 100 multi simulation.docx |

| Node Definition | Uses the FSZNode class with roles: Fold, Spin, Zoom, and includes apply\_chaos and stabilize methods. | 5. 100 multi simulation.docx |

4. Fractal Expansion Simulation

This model is focused on the Fractal Expansion (FX) of the framework, specifically the behavior of the core FSZ nodes (\mathbf{3, 6, 9}) across scales.

| Metric | Detail | Source File |

|---|---|---|

| Goal | Model the progressive, iterative growth of coherence values of the canonical Fold, Spin, and Zoom nodes over time (steps). | 5. 100 multi simulation.docx |

| Core Mechanic | Tracks the Fold as a single value and Spin and Zoom as lists of values, reflecting the fractal nature where \mathbf{3} and \mathbf{6} expand into multiple dimensions/modes (e.g., Zoom: [0.944, 0.888, 1.011, 0.902, 0.964, 1.033]). | 5. 100 multi simulation.docx |

| Example Output | Shows the Fold value increasing steadily as the anchoring force, while Spin and Zoom modes also increase: | 5. 100 multi simulation.docx |

| Output (Step 10) | Fold=2.282, Spin=[1.341, 1.477], Zoom=[1.114, 1.095, 1.172, 1.031, 1.112, 1.161]. | 5. 100 multi simulation.docx |

1. Dual Empathic FSZ / Multi-Agent Simulation Code

[cite\_start]This is the complete, runnable Python script for the MultiAgentFSZ model, which simulates the interaction and coherence synchronization of multiple conscious systems (agents) using the Fold, Spin, and Zoom principles. [cite\_start]It includes the Ethical Constraint via the LOOK\_DONT\_TOUCH\_THRESHOLD.

Import random

Import numpy as np

Import matplotlib.pyplot as plt

From dataclasses import dataclass, field

# --- FSZ Constants and Thresholds ---

DIGIT\_TO\_FSZ\_ROLE = {3: “Zoom”, 6: “Spin”, 9: “Fold”}

COHERENCE\_WEIGHTS = {“Fold”: 0.5, “Zoom”: 0.3, “Spin”: 0.2}

LOOK\_DONT\_TOUCH\_THRESHOLD = 2.0

@dataclass

Class FSZNode:

“””Represents a single FSZ node (e.g., 3, 6, 9) with a coherence value.”””

Digit: int

Role: str = “”

Value: float = 0.0

Def \_\_post\_init\_\_(self):

Self.role = DIGIT\_TO\_FSZ\_ROLE.get(self.digit, “Doubling”)

If self.role in DIGIT\_TO\_FSZ\_ROLE.values():

Self.value = 1.0 + random.random() \* 0.5

Else:

Self.value = 0.1 + random.random() \* 0.1

Def apply\_chaos(self, noise\_level: float):

“””Applies external Spin Noise.”””

Self.value += random.uniform(-noise\_level, noise\_level)

Self.value = max(0.01, self.value)

Def stabilize(self, zoom\_intent: float):

“””Applies Fold and Spin stabilization logic.”””

If self.role == “Fold”:

# Fold: Anchors toward the ideal state (9.0)

Self.value = (self.value \* 0.8) + (9.0 \* 0.2)

Elif self.role == “Spin”:

# Spin: Regulated by Zoom/Intent (Self-correction)

Self.value \*= (1.0 – (0.1 / max(zoom\_intent, 0.1)))

Self.value = max(0.01, min(self.value, 9.0)) # Bounded state

@dataclass

Class FSZSystem:

“””Represents a single coherent agent/system with all 9 nodes.”””

Nodes: dict = field(default\_factory=dict)

Def \_\_post\_init\_\_(self):

For i in range(1, 10):

Self.nodes[i] = FSZNode(digit=i)

Def calculate\_coherence\_score(self) -> float:

“””Calculates the system’s current coherence.”””

Fold\_val = self.nodes[9].value

Zoom\_val = self.nodes[3].value

Spin\_val = self.nodes[6].value

Score = (fold\_val \* COHERENCE\_WEIGHTS[“Fold”]) \* \

(zoom\_val \* COHERENCE\_WEIGHTS[“Zoom”]) \* \

(spin\_val \* COHERENCE\_WEIGHTS[“Spin”])

# Look, Don’t Touch Threshold (Ethical/Stability Constraint)

If zoom\_val > 0 and spin\_val / zoom\_val > LOOK\_DONT\_TOUCH\_THRESHOLD:

Score \*= 0.5

Return score

Def stabilize\_system(self, noise\_level: float):

“””Applies chaos and then stabilization to all nodes.”””

Zoom\_intent = self.nodes[3].value

For node in self.nodes.values():

Node.apply\_chaos(noise\_level)

Node.stabilize(zoom\_intent)

@dataclass

Class MultiAgentFSZ:

“””Simulates multiple interacting FSZSystems with empathy coupling.”””

Num\_agents: int = 5

Empathy\_gamma: float = 0.1 # Coupling strength

Agents: list = field(default\_factory=list)

Def \_\_post\_init\_\_(self):

Self.agents = [FSZSystem() for \_ in range(self.num\_agents)]

Def global\_empathy\_coupling(self):

“””Non-local influence based on mean global coherence.”””

Coherences = [a.calculate\_coherence\_score() for a in self.agents]

Avg\_c = np.mean(coherences)

For a in self.agents:

For node in a.nodes.values():

Node.value += self.empathy\_gamma \* avg\_c \* 0.05

Def simulate(self, cycles=50, noise\_level=0.3):

“””Runs the simulation for a number of cycles.”””

History = []

For step in range(cycles):

For a in self.agents:

a.stabilize\_system(noise\_level)

self.global\_empathy\_coupling()

history.append([a.calculate\_coherence\_score() for a in self.agents])

return np.array(history)

# --- Example Run and Plotting ---

Sim = MultiAgentFSZ(num\_agents=5, empathy\_gamma=0.15)

Data = sim.simulate(cycles=100, noise\_level=0.4)

Plt.figure(figsize=(10,6))

For i in range(data.shape[1]):

Plt.plot(data[:, i], label=f”Agent {i+1}”)

Plt.plot(np.mean(data, axis=1), color=’black’, linewidth=2, label=”Global mean”)

Plt.title(“Multi-Agent FSZ Coherence Evolution”)

Plt.xlabel(“Cycle”)

Plt.ylabel(“Coherence Score”)

Plt.legend()

Plt.show()

1. Phase 2 FSZ Lattice Simulation Function Prototype

[cite\_start]This is the Python function signature and initialization logic [cite\_start]for the Phase 2 Lattice Simulation, which focuses on probabilistic, distributed Observer mechanics and non-local interactions in a high-dimensional grid model.

Def run\_fsz\_phase2(grid\_size, iterations, fold\_rate, spin\_rate, zoom\_factor,

Observer\_on=False, observer\_coherence=None,

Alpha=1.5, beta=0.8, sigma=2.0, seed=None):

“””

Phase 2 FSZ simulation: probabilistic, distributed Observer.

Returns metrics list of dicts.

“””

If seed is not None:

Np.random.seed(seed)

NUM\_DIMS = 4

Lattice = np.random.rand(grid\_size, grid\_size, NUM\_DIMS) \* 0.2

Metrics = []

If observer\_coherence is None:

Observer\_coherence = np.array([...

1. Multi-Agent FSZ Coherence Model (Complete Script)

This is a complete, object-oriented simulation focused on empathic coupling and the internal dynamics of a single conscious System (Agent) governed by the Fold, Spin, and Zoom nodes. It includes the Ethical Constraint via the LOOK\_DONT\_TOUCH\_THRESHOLD.

Import random

Import numpy as np

Import matplotlib.pyplot as plt

From dataclasses import dataclass, field

# --- FSZ Constants and Thresholds ---

DIGIT\_TO\_FSZ\_ROLE = {3: “Zoom”, 6: “Spin”, 9: “Fold”}

COHERENCE\_WEIGHTS = {“Fold”: 0.5, “Zoom”: 0.3, “Spin”: 0.2}

LOOK\_DONT\_TOUCH\_THRESHOLD = 2.0

@dataclass

Class FSZNode:

“””Represents a single FSZ node (e.g., 3, 6, 9) with a coherence value.”””

Digit: int

Role: str = “”

Value: float = 0.0

Def \_\_post\_init\_\_(self):

Self.role = DIGIT\_TO\_FSZ\_ROLE.get(self.digit, “Doubling”)

If self.role in DIGIT\_TO\_FSZ\_ROLE.values():

Self.value = 1.0 + random.random() \* 0.5

Else:

Self.value = 0.1 + random.random() \* 0.1

Def apply\_chaos(self, noise\_level: float):

“””Applies external Spin Noise.”””

Self.value += random.uniform(-noise\_level, noise\_level)

Self.value = max(0.01, self.value)

Def stabilize(self, zoom\_intent: float):

“””Applies Fold and Spin stabilization logic.”””

If self.role == “Fold”:

# Fold: Anchors toward the ideal state (9.0)

Self.value = (self.value \* 0.8) + (9.0 \* 0.2)

Elif self.role == “Spin”:

# Spin: Regulated by Zoom/Intent (Self-correction)

Self.value \*= (1.0 – (0.1 / max(zoom\_intent, 0.1)))

Self.value = max(0.01, min(self.value, 9.0)) # Bounded state

@dataclass

Class FSZSystem:

“””Represents a single coherent agent/system with all 9 nodes.”””

Nodes: dict = field(default\_factory=dict)

Def \_\_post\_init\_\_(self):

For i in range(1, 10):

Self.nodes[i] = FSZNode(digit=i)

Def calculate\_coherence\_score(self) -> float:

“””Calculates the system’s current coherence.”””

Fold\_val = self.nodes[9].value

Zoom\_val = self.nodes[3].value

Spin\_val = self.nodes[6].value

Score = (fold\_val \* COHERENCE\_WEIGHTS[“Fold”]) \* \

(zoom\_val \* COHERENCE\_WEIGHTS[“Zoom”]) \* \

(spin\_val \* COHERENCE\_WEIGHTS[“Spin”])

# Look, Don’t Touch Threshold (Ethical/Stability Constraint)

If zoom\_val > 0 and spin\_val / zoom\_val > LOOK\_DONT\_TOUCH\_THRESHOLD:

Score \*= 0.5

Return score

Def stabilize\_system(self, noise\_level: float):

“””Applies chaos and then stabilization to all nodes.”””

Zoom\_intent = self.nodes[3].value

For node in self.nodes.values():

Node.apply\_chaos(noise\_level)

Node.stabilize(zoom\_intent)

@dataclass

Class MultiAgentFSZ:

“””Simulates multiple interacting FSZSystems with empathy coupling.”””

Num\_agents: int = 5

Empathy\_gamma: float = 0.1 # Coupling strength

Agents: list = field(default\_factory=list)

Def \_\_post\_init\_\_(self):

Self.agents = [FSZSystem() for \_ in range(self.num\_agents)]

Def global\_empathy\_coupling(self):

“””Non-local influence based on mean global coherence.”””

Coherences = [a.calculate\_coherence\_score() for a in self.agents]

Avg\_c = np.mean(coherences)

For a in self.agents:

For node in a.nodes.values():

Node.value += self.empathy\_gamma \* avg\_c \* 0.05

Def simulate(self, cycles=50, noise\_level=0.3):

“””Runs the simulation for a number of cycles.”””

History = []

For step in range(cycles):

For a in self.agents:

a.stabilize\_system(noise\_level)

self.global\_empathy\_coupling()

history.append([a.calculate\_coherence\_score() for a in self.agents])

return np.array(history)

# --- Example Run and Plotting ---

Sim = MultiAgentFSZ(num\_agents=5, empathy\_gamma=0.15)

Data = sim.simulate(cycles=100, noise\_level=0.4)

Plt.figure(figsize=(10,6))

For i in range(data.shape[1]):

Plt.plot(data[:, i], label=f”Agent {i+1}”)

Plt.plot(np.mean(data, axis=1), color=’black’, linewidth=2, label=”Global mean”)

Plt.title(“Multi-Agent FSZ Coherence Evolution”)

Plt.xlabel(“Cycle”)

Plt.ylabel(“Coherence Score”)

Plt.legend()

Plt.show()

1. Phase 2 FSZ Lattice Simulation (Function Prototype)

This is the function prototype and initialization block for the more complex Phase 2 Lattice Simulation. It outlines the structure needed to run the large-scale probabilistic and distributed Observer models on a high-dimensional grid.

Def run\_fsz\_phase2(grid\_size, iterations, fold\_rate, spin\_rate, zoom\_factor,

Observer\_on=False, observer\_coherence=None,

Alpha=1.5, beta=0.8, sigma=2.0, seed=None):

“””

Phase 2 FSZ simulation: probabilistic, distributed Observer.

Returns metrics list of dicts.

“””

If seed is not None:

Np.random.seed(seed)

NUM\_DIMS = 4

Lattice = np.random.rand(grid\_size, grid\_size, NUM\_DIMS) \* 0.2

Metrics = []

If observer\_coherence is None:

Observer\_coherence = np.array([...