# The Spectral Cognitive Resonator: A Dynamic Architecture for Agent Memory, Routing, and Self-Adaptation

A Dynamic Architecture for Agent Memory, Routing, and Self-Adaptation

*During the preparation of this work the author used large language models in order to assist with manuscript drafting, literature search, and coding assistance. After using these tools, the author reviewed and edited the content as needed and takes full responsibility for the content of the published article.*

Tamás Nagy, Ph.D.

tnagyphd@gmail.com

Draft • March 2026

## Abstract

Current AI agent architectures either use static retrieval (RAG) or unstructured agent loops (ReAct, Reflexion) with no formal guarantees on memory utilization, routing optimality, or safe self-adaptation. We introduce the **Spectral Cognitive Resonator (SCR)**: a six-layer architecture where agent state is decomposed into spectral modes, memory is compressed into a mode manifold, task routing is driven by spectral energy profiles, and self-adaptation is gated by five explicit safety checks. The system state at time $t$ is $S_t = (M_t, G_t, R_t, U_t, P_t)$ — memory, graph, resonator, uncertainty, and policy states — evolved under a bounded operator with provable stability. We show that the SCR framework strictly generalizes both flat RAG retrieval and reactive agent loops, and provide concrete algorithms for each layer. The architecture is grounded in the existing Nous system, where the SCR formalizes and extends patterns already emerging from empirical use.

---

## 1. Introduction

### 1.1 The Problem with Current Agents

Modern LLM agents operate in a loop: observe, think, act, observe. But they lack **structured internal state**. Each call to the LLM starts from scratch with whatever context fits in the prompt window. This creates five failure modes:

1. **Memory loss**: insights discovered in step 10 are forgotten by step 50.
2. **Duplicate work**: the agent re-derives the same conclusion multiple times.
3. **Poor routing**: high-value tasks receive the same resources as low-value ones.
4. **No uncertainty awareness**: the agent cannot distinguish what it knows from what it guesses.
5. **Unsafe adaptation**: any learning or self-modification is ad-hoc, with no rollback guarantee.

### 1.2 What a Resonator Is

The SCR is not a knowledge base. It is not a vector store. It is a **dynamic state machine** with spectral structure:

- State is decomposed into modes with interpretable roles.
- Evolution is bounded by construction (Theorem 1 of the Spectral-State Neural Network paper).
- Decisions are emitted by a controller that sees the full spectral state.
- Self-modification requires passing five explicit safety gates.

The term "resonator" captures the key property: certain mode combinations **resonate** (reinforce each other across time), while others **dephase** (cancel out). The system naturally amplifies coherent patterns and suppresses noise.

### 1.3 Contributions

1. **Formal architecture**: six layers with defined interfaces, state types, and information flow.
2. **Algorithms**: concrete procedures for memory compression, graph routing, policy emission, and gated self-update.
3. **Safety protocol**: five gates with formal conditions for self-modification.
4. **Convergence analysis**: conditions under which the SCR converges to a stable operating regime.
5. **Empirical grounding**: mapping to the existing Nous system components.

---

## 2. System State

### 2.1 State Definition

The full system state at time $t$ is:

$$S_t = (M_t, G_t, R_t, U_t, P_t)$$

| Component | Type | Description |
|---|---|---|
| $M_t$ | Memory manifold | Episodic, semantic, procedural, uncertainty memory partitions |
| $G_t$ | Graph state | Claim/task/output graph with dependency edges |
| $R_t$ | Resonator state | Spectral mode amplitudes $z_t \in \mathbb{R}^K$ |
| $U_t$ | Uncertainty state | Per-mode confidence, calibration history |
| $P_t$ | Policy state | Routing table, budget allocation, escalation thresholds |

### 2.2 State Update

The global update rule is:

$$S_{t+1} = F(S_t, x_t, e_t)$$

where $x_t$ is the new input (session events, code changes, proof outcomes, telemetry) and $e_t$ is external feedback (teacher corrections, verifier verdicts).

The update decomposes layer by layer:

$$M_{t+1} = f_M(M_t, x_t)$$
$$G_{t+1} = f_G(G_t, M_{t+1}, x_t)$$
$$R_{t+1} = G_t^{\text{mode}} \cdot U_t^{\text{mix}} \cdot R_t + P_{\text{in}} \cdot x_t + e_t$$
$$U_{t+1} = f_U(U_t, R_{t+1}, e_t)$$
$$P_{t+1} = f_P(P_t, R_{t+1}, U_{t+1}, G_{t+1})$$

---

## 3. Layer Specifications

### 3.1 L0: Ingestion and Normalization

**Input**: raw events from heterogeneous sources (session transcripts, code diffs, proof compilation results, telemetry logs, teacher outputs).

**Output**: typed atomic events with canonical schema.

**Algorithm**:

```
Procedure: INGEST(raw_events)
1. For each raw event:
   a. Classify event type: {observation, action, outcome, feedback, error}
   b. Extract:
      - atom_id: deterministic hash of content
      - atom_type: {question, direction, realization, failure, procedure}
      - source: originating system
      - timestamp: wall-clock time
      - evidence: supporting data or references
      - confidence: initial confidence score (1.0 for verified, 0.5 for unverified)
   c. Deduplicate against existing atom store by content hash
   d. Emit normalized memory atom
2. Return list of new atoms
```

### 3.2 L1: Memory Manifold

Memory is a typed manifold with four partitions, each equipped with a spectral basis.

**Partition 1: Episodic memory** — time-indexed events with recency decay.

**Partition 2: Semantic memory** — stable claims promoted from episodic after repeated appearance.

**Partition 3: Procedural memory** — what worked operationally (action → outcome patterns).

**Partition 4: Uncertainty memory** — confidence scores, contradictions, known unknowns.

**Spectral compression per partition**:

```
Procedure: COMPRESS_PARTITION(atoms, K_max)
1. Embed atoms: E = [embed(a_1), ..., embed(a_N)]   R^{N × d}
2. Compute similarity kernel: K = E E^T / N
3. Eigendecompose: K = V Λ V^T
4. Select K* modes by GCV or energy threshold (90% variance)
5. Project atoms onto top-K* modes: C = V[:, :K*]^T E
6. Per mode k:
   - centroid_k = mean of atoms in mode k
   - uncertainty_k = variance of projections in mode k
   - recency_k = max timestamp of atoms in mode k
7. Return {(centroid_k, uncertainty_k, recency_k)}_{k=1}^{K*}
```

**Retrieval**: Given a query $q$, project into spectral space, retrieve atoms from the top-$k$ modes by projected similarity, weighted by recency and confidence.

**Promotion rule**: An episodic atom that appears in the same mode across $\geq 3$ sessions with confidence $\geq 0.8$ is promoted to semantic memory.

### 3.3 L2: Spectral Resonator Core

This is the SSNN (Spectral-State Neural Network) embedded as the central state engine.

**State update**:

$$z_{t+1} = G_t \cdot U_t \cdot z_t + P_{\text{in}} \cdot x_t + e_t$$

where:

- $z_t \in \mathbb{R}^K$: spectral mode amplitudes.
- $U_t \in O(K)$: mode-mixing operator (orthogonal via Cayley parameterization).
- $G_t = \text{diag}(g_1, \ldots, g_K)$: mode gates.
- $P_{\text{in}} \in \mathbb{R}^{K \times d}$: input projection from memory summary.
- $e_t \in \mathbb{R}^K$: correction from verifier/teacher feedback.

**Stability constraint**: $\|G_t U_t\| \leq 1$ by construction.

**Mode interpretation**: Modes are labeled after training by their correlation with system observables:

| Mode type | Gate profile | Example |
|---|---|---|
| Long-term memory | $g_k \approx 1$ | "We proved Theorem X last week" |
| Working memory | $g_k \approx 0.7$ | "Current proof attempt uses tactic Y" |
| Reactive | $g_k \approx 0.3$ | "This step failed, try alternative" |
| Noise | $g_k \approx 0$ | Irrelevant transient signals |

### 3.4 L3: Graph Intelligence

**Graph construction**: Nodes are claims, theorems, tasks, and outputs. Edges carry typed weights:

| Edge type | Weight source |
|---|---|
| Dependency | Proof import / code reference |
| Implication | Logical entailment |
| Empirical support | Experimental validation |
| Bridge potential | Spectral bridge score |

**Graph Laplacian**: $L = D - W$ where $D$ is the degree matrix and $W$ is the weighted adjacency matrix.

**Spectral services**:

```
Procedure: GRAPH_SPECTRAL_ANALYSIS(G)
1. Compute normalized Laplacian: L_norm = D^{-1/2} L D^{-1/2}
2. Eigendecompose: L_norm = ΦΛ _L Φ^T
3. Extract:
   a. Fiedler vector: Φ[:, 1] (second eigenvector)→
      identifies the weakest cut between graph communities
   b. Spectral centrality: c_i = sum_k Φ|[i, k]|^2 / _k→
      nodes reachable from everywhere
   c. Effective resistance: R_ij = sum_k Φ([i,k] - Φ[j,k])^2 / _k→
      which new edge most improves connectivity
   d. Bridge score: nodes where Fiedler vector changes sign→
      cross-community connectors
   e. Frontier: nodes with high centrality but degree < median→
      underexplored high-value targets
4. Return spectral profile {communities, bridges, frontiers, centrality}
```

### 3.5 L4: Policy and Routing Controller

The controller emits bounded decisions from the full spectral state.

**Decision space**: {`start`, `hold`, `rotate`, `prune`, `repair`, `escalate`}

**Decision algorithm**:

```
Procedure: EMIT_DECISION(R_t, U_t, G_t, P_t, budget)
1. Compute spectral energy profile: E_k = |z_k|^2 for each mode k
2. Compute frontier pressure: F = max bridge score among unresolved frontier nodes
3. Compute blocker load: B = count of modes with high uncertainty and low gate
4. Decision logic:
   - If F > threshold_frontier AND budget > min_cost:→
     START new task targeting highest-bridge frontier node
   - If B > threshold_blocker:→
     REPAIR: re-examine blockers with fresh context
   - If E_dominant / E_total > 0.9 (mode collapse):→
     ROTATE: force exploration of non-dominant modes
   - If cost_spent > budget * 0.8 AND gain < expected * 0.3:→
     PRUNE: abandon current task, reallocate
   - If U_t shows increasing uncertainty trend across 3+ steps:→
     ESCALATE: request teacher model or human review
   - Otherwise:→
     HOLD: continue current task
5. Record decision with state snapshot for audit
6. Return (decision, target, resource_allocation)
```

### 3.6 L5: Teacher Distillation and Adaptation

**Multi-objective distillation**: The SCR learns from stronger models not just their outputs, but their:

1. **Final answers**: standard output distillation.
2. **Reasoning traces**: intermediate chain-of-thought steps.
3. **Retrieval decisions**: which context the teacher chose to use.
4. **Uncertainty calibration**: where the teacher expressed doubt.

**Distillation algorithm**:

```
Procedure: DISTILL(teacher_trace, student_trace)
1. Align: match teacher steps to student steps by semantic similarity
2. For each aligned pair (t_step, s_step):
   a. Output agreement: sim(t_output, s_output)
   b. Reasoning agreement: sim(t_reasoning, s_reasoning)
   c. Retrieval overlap: |t_context  s_context| / |t_context  s_context|
   d. Uncertainty alignment: |t_confidence - s_confidence|
3. Compute aggregate distillation score
4. If score > promotion_threshold:→
    Mark student behavior as "promote candidate"→
    Queue for safety gate evaluation
5. Return distillation record
```

---

# 4. Self-Update Safety Protocol

## 4.1 Five Gates

Any proposed self-modification (policy parameter change, memory schema update, gate bias adjustment) must pass all five gates:

| Gate | Condition | Failure action |
|---|---|---|
| G1: Local objective | Proposed change improves local task metric | REJECT |
| G2: No regression | Critical benchmark slice unchanged or improved | ROLLBACK |
| G3: Calibration | Uncertainty calibration error non-increasing | HOLD |
| G4: Duplicate suppression | Duplicate work rate non-increasing | HOLD |
| G5: Rollback artifact | Complete rollback state has been saved | BLOCK |

## 4.2 Gate Evaluation Algorithm

```
Procedure: EVALUATE_SAFETY_GATES(proposed_change, current_state)
1. Save rollback artifact: snapshot of (M_t, G_t, R_t, U_t, P_t)→
    If save fails: BLOCK (G5 fails)
2. Apply proposed change in shadow mode (no persistent write)
3. Run critical benchmark slice on shadow state
4. Check G1: compare local metric (shadow vs current)→
    If worse: REJECT
5. Check G2: compare benchmark metrics (shadow vs current)→
    If any critical metric regressed: ROLLBACK
6. Check G3: compare calibration error (shadow vs current)→
    If worse: HOLD for human review
7. Check G4: compare duplicate suppression rate→
    If worse: HOLD for human review
8. If all gates pass:→
    COMMIT: apply change to persistent state→
    Log: change_id, gate scores, timestamp
9. Return (decision, gate_scores)
```

---

# 5. Convergence Analysis

## 5.1 Operating Regime Convergence

**Theorem** (Regime Convergence). Under the following conditions:

1. Bounded input energy: $\|x_t\| \leq B_x$ for all $t$.

2. Stable gate: $\max_k g_k^{(t)} < 1 - \delta$ for some $\delta > 0$.
3. Bounded correction: $\|e_t\| \leq B_e$ for all $t$.

The resonator state $z_t$ converges to a bounded attractor:

$$\|z_t\| \leq \frac{B_x + B_e}{1 - (1 - \delta)} = \frac{B_x + B_e}{\delta}$$

and the spectral energy profile $E_k = |z_k|^2$ converges to a stationary distribution within $O(1/\delta)$ steps.

*Proof sketch.* Since $\|G_t U_t\| \leq 1 - \delta$ and the input is bounded, the recurrence is a contraction mapping with bounded forcing. By Banach fixed-point theorem, the state converges to a unique attractor. □

### 5.2 Memory Manifold Stability

**Proposition**. If the stream of incoming atoms is stationary (i.e., the topic distribution does not change), the spectral compression of each memory partition converges to a fixed set of mode centroids.

**Corollary** (Regime-Switch Detection). A sudden change in the incoming topic distribution causes the mode centroids to shift. The magnitude of centroid shift is proportional to the distributional distance, providing a built-in regime-switch detector.

---

## 6. Metrics and Evaluation

### 6.1 Primary Metrics

| Metric | Definition | Target |
|---|---|---|
| Verified gain per cost | (accepted verified outcomes) / (total LLM tokens spent) | Maximize |
| Time-to-gain | Wall-clock time from task start to first accepted outcome | Minimize |
| Duplicate suppression rate | 1 - (duplicate work items) / (total work items) | $> 0.90$ |
| Blocker classification precision | (correctly identified blockers) / (items classified as blockers) | $> 0.85$ |

### 6.2 Secondary Metrics

| Metric | Definition | Target |
|---|---|---|
| Uncertainty calibration error | ECE across confidence bins | $< 0.10$ |
| Retrieval relevance | Precision@K for retrieved memory atoms | $> 0.75$ |

| Metric | Definition | Target |
|---|---|---|
| Bridge yield | New cross-domain connections per run family | Increasing trend |
| Rollback frequency | Fraction of self-updates that trigger rollback | $< 0.20$ |
| Mode utilization | Fraction of modes with $E_k > 0.01 \cdot E_{\text{total}}$ | $> 0.30$ |

---

## 7. Mapping to Nous

The SCR is not a greenfield design. It formalizes the architecture already emerging in Nous:

| SCR Layer | Nous Component | Status |
|---|---|---|
| L0: Ingestion | Session transcripts, code diffs, proof logs | Exists |
| L1: Memory | _workspace/memory/, MEMORY.md, active_context.md | Exists (flat) |
| L2: Resonator | Not yet implemented | **New** |
| L3: Graph | ./nous graph suggest, ./nous graph semantic | Partial |
| L4: Policy | Golden Rain controller, gym routing | Partial |
| L5: Distillation | Teacher model calls, gym oracle | Partial |
| Safety gates | Gym graduation, benchmark slice | Partial |

The SCR upgrade path:

1. **Phase 1** (immediate): Add spectral compression to memory retrieval; instrument routing decisions with explicit labels.
2. **Phase 2** (2-4 weeks): Implement the SSNN resonator core; connect graph spectral analysis to routing.
3. **Phase 3** (4-8 weeks): Add distillation pipeline and safety gates; close the self-update loop.

---

## 8. Failure Modes and Mitigations

| Failure mode | Symptom | Mitigation |
|---|---|---|
| Mode collapse | $> 90\%$ energy in one mode | Entropy floor: reject gate updates that push entropy below threshold |
| Over-damping | All gates $< 0.3$, no exploration | Minimum exploration quota: force at least 10% of budget to frontier tasks |
| Hallucinated bridge | Bridge score high but no structural evidence | Require $\geq 2$ independent structural signals for bridge promotion |
| Distillation overfit | Student copies teacher errors | Mixed teacher sets: distill from $\geq 2$ different teachers, validate on held-out tasks |
| Silent regression | Benchmark scores unchanged but real performance drops | Shadow benchmark with diverse task distribution, not just the tracked slice |
| Runaway self-modification | Cascading changes that compound small errors | Maximum 1 self-update per evaluation window; mandatory cooldown after each change |

---

## 9. Connection to Other Papers

- **ml_spectral_neural_architecture**: Provides the L2 resonator core (SSNN).
- **ml_spectral_memory_graph_routing**: Provides the L1 + L3 memory and graph layers.
- **meta_meta_spectral**: The "spectral of spectrals" layer produces the controller's decision surface.
- **ml_spectral_knowledge_distillation**: Provides the L5 distillation methodology.
- **ml_spectral_intelligence**: The data spectral exponent $s$ sets the mode budget for the resonator.

---

## 10. Full Spec Reference

The detailed data contracts, schemas, and MVP acceptance criteria are in:

docs/analysis_0314_cheap_agent_information_design.md → Section "Architecture Spec — Spectral Cognitive Resonator (SCR)"