Specification: Iterative Residual Convergence Memory Attuner

Identifier: SPEC-2608.01
Authors: J. Kornreich and Collaborators
Target Substrate: Gemma 4 31B ($D=5376$), Gemstone CLI runtime (`governor_residual_attunement.go`)
Status: Verified & Compiled
Classification: Technical Architecture Standard

1. Scope and Purpose

This specification defines the algorithmic requirements, data structures, and mathematical formulas for the **Iterative Residual Convergence Memory Attuner (IRCA)** and **Syntax-Aware Boundary Scoring (SABS)** within the Gemstone Governor engine.

The engine replaces coarse-grained multi-level chunking with dynamic error minimization, iteratively distilling high-resonance code excerpts from canonical memory shards until the semantic residual error vector Δ satisfies the dynamic convergence threshold ϵ_{dyn}.

```
graph TD
    A[Raw Document Data] --> B[Generate Intent Vector h_t]
    B --> C[Compute Initial Residual d_0]
    C --> D{d_k <= ε_dyn OR |S_k| <= 150 B?}
    D -->|Yes| E[Apply SABS Contextual Padding]
    D -->|No| F[Partition Subspaces]
    F --> G[Score Candidates with SABS Bonus]
    G --> H{Stagnation Guard Triggered?}
    H -->|Yes| E
    H -->|No| I[Descend to Best Subspace S_{k+1}]
    I --> D
    E --> J[Emit Distilled Snippet S*]

    style E fill:#dcfce7,stroke:#16a34a,stroke-width:2px;
    style J fill:#dbeafe,stroke:#2563eb,stroke-width:2px;
```

2. Mathematical Formulations

2.1 Latent Space Vector Mapping

Let S be a tokenized text block and $\mathcal{D} = 5376$ the latent vector dimension:

$$h(S) = \frac{1}{\sqrt{N}} \sum_{i=1}^{N} \frac{1}{\sqrt{i}} \mathbf{w}(t_i)$$

$$\hat{h}(S) = \frac{h(S)}{|h(S)|_2}$$

where $\mathbf{w}(t_i) \in \{-1, +1\}^{\mathcal{D}}$ is a deterministic signed projection seeded by SHA-256 block hashing over term t_i.

2.2 Semantic Residual Distance

Given target vector $\hat{h}_t$ and candidate subspace vector $\hat{v}(S_k)$:

$$d(\hat{h}_t, \hat{v}(S_k)) = 1.0 - (\hat{h}_t \cdot \hat{v}(S_k))$$

2.3 Syntax-Aware Dynamic Epsilon (SABS)

The convergence threshold ϵ_{dyn} tightens dynamically around syntactic pivot landmarks:

$$\epsilon_{dyn} = \epsilon_{base} \cdot \exp(-\alpha \cdot \Lambda)$$

where: - $\epsilon_{base} = 0.40$ (default threshold) - $\alpha = 0.5$ (sensitivity factor) - $\Lambda = \sum_{i=1}^{M} w_i$ (aggregate weight of detected pivot landmarks) - Pivot weights: `func` / `type` / `struct` / `class` (0.25), `return` / `if` / `switch` (0.15), `#` / `##` headings (0.20). - Clamped range: $\epsilon_{dyn} \in [0.24, 0.40]$.

---

3. Data Structures & Go Type Definitions

```
// ResidualAttunerConfig defines hyperparameters for the iterative descent engine.
type ResidualAttunerConfig struct {
    Dimension           int     // 5376 (Gemma 4 31B)
    BaseEpsilon         float64 // 0.40
    MaxIterations       int     // 8
    StagnationThreshold float64 // 0.005
    MinAtomicBytes      int     // 150 bytes
    PaddingBytes        int     // 128 bytes
}

// ResidualAttunementResult contains output metrics from a descent trajectory.
type ResidualAttunementResult struct {
    AttunedSnippet      string   // Syntactically padded distilled snippet
    InitialDistance     float64  // Initial residual distance d_0
    FinalDistance       float64  // Final residual distance d*
    IterationsExecuted  int      // Steps taken (k <= 8)
    Converged           bool     // True if d* <= ε_dyn
    SyntacticIntegrity  float64  // SABS structural integrity score [0.0, 1.0]
    PivotTokensDetected []string // Identified structural landmarks
}
```

---

4. Subspace Partitioning Hierarchy

```
flowchart LR
    A["Raw Shard (12 KB)"] -->|"Level 1: Section Split"| B["Semantic Section (~3.5 KB)"]
    B -->|"Level 2: Paragraph Split"| C["Code Block (~900 B)"]
    C -->|"Level 3: Statement Split"| D["Atomic Closure (~250 B)"]
    D -->|"SABS Padding Buffer"| E["Grounded Working Set (~400 B)"]

    style E fill:#dcfce7,stroke:#16a34a,stroke-width:2px;
```

---

5. Verification & Telemetry Invariants

1. **Deterministic Reproducibility:** Repeated invocations with identical (h_t, D) must yield bit-identical snippets S^*.
2. **Zero Allocation Fast Path:** Vector operations are bounded to stack allocations where possible, maintaining $<50\mu\text{s}$ CPU overhead.
3. **AST Safety:** Distilled code snippets must never contain unclosed brackets, braces, or severed function declarations.