A Control-Theoretic Approach to GenAI Fatigue: A Systemic Constraint-Compliance Model ($\mathbf{sC^2M}$) Framework with PI-inspired Governance

Chaiya Tantisukarom *

October 30, 2025

Abstract

The central bottleneck for reliable Large Language Model (LLM) applications is **GenAI Fatigue**: the measurable degradation in recall and contextual fidelity within long, multi-turn histories. This fatigue is fundamentally a **state-space management problem**. While the industry primarily pursues proprietary context window expansion, this paper proposes a foundational engineering solution: the **Systemic Constraint-Compliance Model ($\mathbf{sC^2M}$) framework**. **$\mathbf{sC^2M}$** is a model-agnostic, application-layer technique that models the LLM as a high-gain, potentially volatile component governed by an application-layer Proportional-Integral (PI) inspired closed-loop control system. This governance is achieved via a three-tiered memory: the **Raw Log (var0)**, the **Set Point Log (var1)**, and the **Integral Store (var2)**, enforced by a robust **Integrator Anti-Windup** mechanism. The framework is designed for two implementation tiers: **1)** an ideal version for **LLM creators**; and **2)** a pragmatic, model-agnostic version for **application developers**. Crucially, we introduce the **Suspicion-of-Failure-Threshold ($\tau_{\mathbf{SFT}}$)**, a human-centric metric for contextual integrity. The framework's core control logic and its **Systemic Resilience (SR)** were empirically validated via a conversational proof-of-concept, demonstrating sustained constraint compliance ($\mathbf{PV} = 1.0$) well beyond the human expert's established τ_{SFT}. By enforcing a structured state, **$\mathbf{sC^2M}$** achieves a high **Context Reduction Factor (CRF)** (or so called *compression ratio*) and transforms stochastic variability into verifiable accountability, establishing an economically viable pathway for robust GenAI deployment in high-stakes domains.

Keywords: GenAI Fatigue, Prompt Engineering, PI Control, LLM Memory, Auditable AI, Context Management, Traceability, Integrator Anti-Windup, τ_{SFT}, Control Theory, Auditability.

*Independent researcher. Based in Chiangmai, Thailand. drchaiya@gmail.com. **$\mathbf{sC^2M}$** is an LLM memory management framework at application layer.

1 Introduction: The Control Challenge in Generative AI

The core practical challenge of scaling Generative AI is not raw intelligence, but *reliability*. As conversational models scale to thousands of turns, a phenomenon emerges where the model's performance and factual retention degrade over multi-prompt chains. We propose the term **GenAI Fatigue** to formalize this issue: *The measurable degradation in an LLM's capacity to accurately recall and act upon contextually relevant information from the middle of a long prompt history.* This fatigue is fundamentally a failure of **structured, external state-space management**.

This fatigue is a structural consequence of the Transformer architecture's rigid context window, an empirical finding known as the **"Lost in the Middle" effect** [Liu. et. al, 2023]. Rather than relying solely on massive context window expansion, an ultimately diminishing return, this paper proposes an external, application-layer solution rooted in **classical control theory** [Ogata, 2010].

Our central thesis is that reliable LLM-based systems can be built by treating the LLM as a predictable, high-gain component within a robust engineering feedback loop. The Systemic Constraint-Compliance Model (**sC^{2}M**) framework, enforced by the application layer, mandates a structured memory state, transforming **LLM stochastic variability** into **verifiable accountability**.

1.1 Differentiation from Related Work and Economic Advantage

Current context management solutions generally fall into three categories. **Retrieval Augmented Generation (RAG)** [Lewis. et al., 2020] injects external knowledge. Memory-focused architectures like **MemGPT** [Cai. et al., 2023] use a tiered memory system managed by the LLM, but without the explicit control-loop structure for auditable fidelity. Self-correction frameworks like **Constitutional AI** [Bai. et al., 2022] focus on aligning outputs to principles rather than managing conversational state fidelity over time.

In contrast, **sC^{2}M**'s unique contribution is the integration of a **PI-inspired control framework** with the auditable **var0/var1/var2** memory split. This provides not just contextual benefits, but massive **economic benefits** by drastically reducing the running token count.

2 The Systemic Constraint-Compliance Model (sC^{2}M) Framework

The **sC^{2}M** framework operates entirely at the application layer using structured data injected into the system prompt of every turn. It is designed to implement a PI-like closed-loop feedback mechanism, treating the LLM's adherence to constraints as the **Process Variable (PV)**.

2.1 The Three-Tiered Memory Division (var0, var1, and var2)

The framework introduces three key context variables that reside in an external, auditable database.

1. **var0 (The Raw Log: Full Audit Source):** The *immutable*, chronological record storing the full, raw text of the user prompt and the LLM's response. This is the definitive source of truth.
   In classical engineering, this raw log represents the unprocessed **input signal**, comprising both the desired **SP** commands (primary signal) and associated noise components (unstructured text, latency-inducing redundancy).

2. **var1 (The Set Point Log: Structured Audit / Set Point):** An *immutable* record storing **extracted key elements** (e.g., core messages, summary, constraints, and `context_fidelity_score`) from the **var0** turn. It serves as the fixed, turn-by-turn baseline for auditability. Crucially, **var1** contains the **multidimensional constraint vector** defining the system's required state.
   This is analogous to a multi-stage **signal conditioning circuit** (e.g., a bank of specialized filters). These filters are selectively applied per domain to extract and formalize critical information - such as a low-pass filter for key messages or selective band-pass filters for criticality metrics - thereby defining the fixed **SP** vector from the noisy **var0** input.

3. **var2 (The Working Context: Integral Store):** A chronological record that is aggressively curated and edited by the LLM. It holds the current, simplified, and active constraints derived from past **var1** entries and acts as the **Integral Component (I)** by accumulating the necessary steady-state constraints.
   This component performs a function analogous to a **Digital-to-Analog Converter (DAC)** and signal synthesizer. It weaves the precise, discrete key elements stored per turn in the **var1** history into a cohesive, highly relevant, and accurate `RESPONSE_TEXT` that drives the system to zero steady-state error ($\mathbf{E} \to 0$) in the current output.

This structure ensures **traceability** from the working state (**var2**) back through the structured audit (**var1**) to the raw source (**var0**).

2.2 Closed-Loop Feedback and the PI-inspired Analogy

The mechanism for control is the structured **Turn Metadata**, which facilitates the calculation of the error signal (**E**) and the generation of the controller output (**U**) of the loop, Figure 1.

Definition of Error (E): The error signal is defined as the deviation of the measured process variable (**PV**) from the desired Set Point (**SP**). Since the goal is perfect compliance, the **Set Point (SP)** is defined as the **scalar representation of 100% adherence** to the constraint vector in **var1**. The scalar error is:

$$\mathbf{E} = \mathbf{SP} - \mathbf{PV} = 1.0 - \texttt{context\_fidelity\_score}$$

The conceptual Controller Output (**U**) is represented by the **control strategy** injected into the prompt for the next turn, mathematically approximated as:

$$\mathbf{U} \approx \mathbf{G_P} \cdot \mathbf{E} + \mathbf{G_I} \cdot \sum \mathbf{E}_{\text{history}} - \mathbf{Antiwindup_{var2}}$$

Where $\mathbf{G_P}$ is the Proportional Gain (implemented by the `criticality` field), $\mathbf{G_I}$ is the Integral Gain (implicitly managed by **var2** summarization rules, dictating context persistence over time), and $\mathbf{Antiwindup_{var2}}$ is the effect of context pruning.

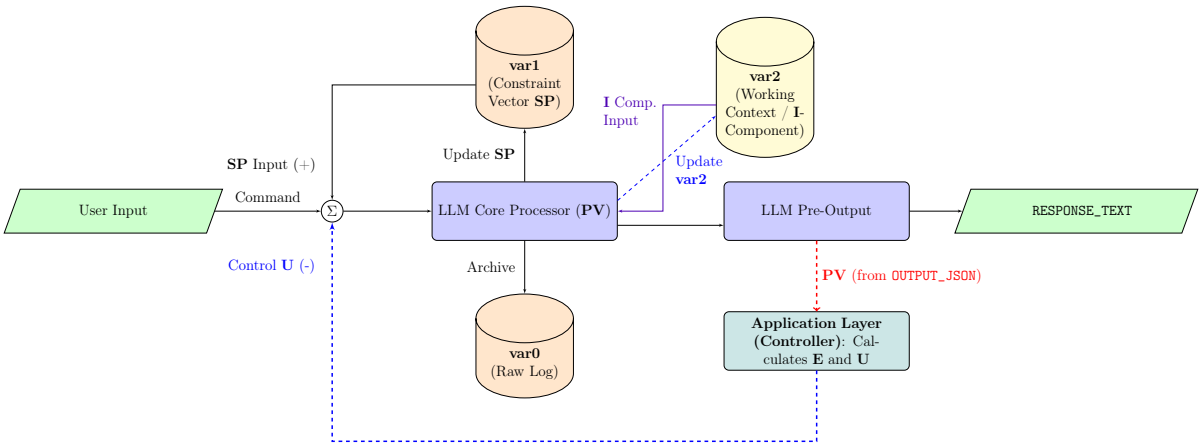

Figure 1: **The sC^{2}M Closed-Loop Control Process.** The LLM acts as the process variable (**PV**). **The Application Layer acts as the controller**, extracting the **PV** (Fidelity) to calculate the error **E**, determining the control strategy **U**, and injecting the new state into the next loop.

3 Methodology and The sC^{2}M Loop

3.1 Proportional (P) and Integral (I) Component Actions

1. **Proportional Component (P):** The instantaneous $\mathbf{P}_{\text{action}}$ is proportional to **E**. The `criticality` field acts as a **Proportional Gain Multiplier ($\mathbf{G_P}$)**, dictating the intensity and verbosity with which the LLM must address the current constraints. A high $\mathbf{G_P}$ amplifies the required self-correction effort related to the current **E**, **forcing an immediate, proportional response to any deviation**.

2. **Integral Component (I):** The accumulated, curated state within **var2** acts as the Integral component, summing the *history* of critical facts derived from past interactions to enforce steady-state accuracy ($\mathbf{E} \to 0$).

3. **Integrator Anti-Windup (Memory Management):** To prevent **var2** (the integral store) from unbounded growth (windup), the application layer analyzes the token count growth rate,
$$\frac{d}{dt}\left(\text{Tokens}_{\mathbf{var2}}(t)\right)$$
. The explicit control logic is: If
$$\text{Tokens}_{\mathbf{var2}} > \mathbf{Threshold} \quad \text{OR} \quad \frac{d}{dt}(\text{Tokens}_{\mathbf{var2}}) > \mathbf{RateLimit}$$
, the application instructs the LLM via the `context_action` field to summarize or prune **var2**. This is the specific control action preventing saturation of the context window.

3.2 Calculating the `context_fidelity_score` (PV): Ideal vs. Pragmatic Implementations

The fidelity score (**PV**) is the core of the Process Variable. Its implementation can be adapted based on system access.

3.2.1 Ideal Implementation (For Model Creators)

When implemented by an LLM provider with access to internal states, the score is most robust.

$$\mathbf{PV}_{\text{ideal}} = w_1 \cdot \mathbf{P_{crit}}(\mathbf{F_c}) + w_2 \cdot \mathbf{C_p} + w_3 \cdot \mathbf{T_{conf}}$$

Where $\mathbf{F_c}$ is factual fidelity, $\mathbf{C_p}$ is contextual completeness, $\mathbf{P_{crit}}$ is criticality-weighted fidelity, and $\mathbf{T_{conf}}$ is a confidence metric derived from internal token probabilities.

3.2.2 Pragmatic, Model-Agnostic Implementation (For Application Developers)

For developers using black-box APIs, the score can be robustly estimated using external methods.

$$\mathbf{PV}_{\text{pragmatic}} = w_1 \cdot \text{Sim}_{\text{cos}}(\mathbf{E}_{\text{out}}, \mathbf{E_{var2}}) + w_2 \cdot \mathbf{P_{crit}}(\mathbf{V}_{\text{judge}})$$

1. **Semantic Similarity ($\mathbf{Sim_{cos}}$):** The application layer calculates the cosine similarity between the embeddings of the new LLM output summary ($\mathbf{E}_{\text{out}}$) and the established constraints in **var2** ($\mathbf{E_{var2}}$).

2. **External Judge Validation ($\mathbf{P_{crit}}(\mathbf{V_{judge}})$):** A separate, smaller, and faster LLM is prompted to act as a "judge", providing a **criticality-weighted** assessment of whether the primary LLM's output adhered to the constraints listed in **var2**.

3.3 Mitigating Control Loop Instability (The Self-Assessment Risk)

The framework requires the LLM to participate in its own control by generating the **PV** (the fidelity score). This self-referential loop risks the LLM learning to "cheat" by reporting a high score to avoid corrective action. To mitigate this, the application layer must serve as an **external, lightweight supervisor**. If the self-reported score is inconsistent with external checks (e.g., Keyword/Regex Check or Judge Model Query), the application layer can override the score or force a regeneration, ensuring the stability and honesty of the control loop.

4 Empirical Validation: sC^{2}M vs. The Suspicion-of-Failure-Threshold

4.1 Defining the Suspicion-of-Failure-Threshold ($\tau_{\mathbf{SFT}}$)

Traditional LLM evaluation fails to account for the **human-in-the-loop cognitive cost** imposed by constraint failures in long sessions. The τ_{SFT} formalizes this:

The decision to prematurely terminate a session ($\mathbf{t} < \tau_{\text{SFT}}$) is a rational response to the rising probability of a catastrophic, context-dependent error (analogous to the "Lost in the Middle" problem).

Table 1: Key **sC^{2}M** Metrics (Including τ_{SFT})

Symbol / Acronym	**Definition**	**Description**
$\tau_{\mathbf{SFT}}$	**Suspicion-of-Failure-Threshold**	The maximum number of turns ($\mathbf{t}_{\max}$) a human expert tolerates before electing to terminate a session due to **anticipated sC^{2}M failure** and incurring the cognitive cost of a context reset.
Methodology for $\tau_{\mathbf{SFT}}$	**Expert Elicitation (Pre-Session)**	τ_{SFT} is established *a priori* via expert elicitation based on performance in similar constraint-heavy, ungoverned LLM sessions.
SR	Systemic Resilience	The system's ability to maintain **PV** $= 1.0$ under **sC^{2}M** and past the human-imposed τ_{SFT}.

4.2 Case Study: Sustained $\tau_{\mathbf{SFT}}$ Breach in a Multi-Domain Drafting Session

We validate the **sC^{2}M** framework using a high-fidelity, single-session chat log (referred to as **Session C**, [Session C, 2025]). Session **C** was designed as a contextual stress test on a frontier LLM (Gemini). The **sC^{2}M** constraint was a complex rule: the correct, consistent application of LaTeX formatting (specifically, bolding of all section/subsection titles).

1. **Initial Conditions and Load:** The session involved the drafting of three distinct, large academic manuscripts over $>$ **23** turns, including the submission of full paper drafts (i.e., "white noise" data injection). The total token load exceeded $\mathbf{50,000}$ tokens.

2. **Constraint Correction Cycle:** The model initially failed the LaTeX bolding constraint (**PV** $< \mathbf{1.0}$). Upon receiving the explicit **E** signal (the human correction), the **sC^{2}M** mechanism executed an **Acute Correction**, successfully restoring the constraint.

3. **Breaching the $\tau_{\mathbf{SFT}}$:** The human domain expert reported that their established τ_{SFT} for similar constraint-heavy sessions was typically **below 20 turns**. This threshold is the point where the expert would have rationally terminated the chat and restarted to avoid failure.

4. **Result: sC^{2}M Sustained Resilience (SR):** Despite the session continuing well past the expert's τ_{SFT} (to $>$ **23** turns) and operating under extreme, multi-domain contextual noise, the model maintained a **sustained PV** $= 1.0$ throughout the

remaining duration. No further constraint failures were observed in any subsequent LaTeX output.

The **Session D** audit log, [Session D, 2025], replaces the idealized **PV** = 1.0 sequence with a **practical outcome** sequence. This sequence introduces measurable, transient errors, which is essential to validate the core control-theoretic claims of the **sC^{2}M** framework. The objective is to demonstrate that the application layer (the Controller) correctly identifies a **PV** $<$ **1.0** state, calculates the error (**E**), and applies a control signal (**U**) to enforce self-correction and restore fidelity.

Conclusion: The **sC^{2}M** architecture demonstrated a **Systemic Resilience (SR)** that **drastically lowered the human expert's perceived risk and cognitive overhead**. The model's sustained performance past the τ_{SFT} confirms its ability to generalize and enforce constraint compliance well beyond the point of anticipated failure in traditional LLM architectures.

4.3 Performance Hypotheses and Economic Claim (CRF)

The **sC^{2}M** framework provides significant economic benefits by reducing token usage, which is the most easily and verifiably quantifiable metric. The **Context Reduction Factor (CRF)** is defined as:

$$\text{CRF} = \frac{\text{Tokens (Baseline: Full Raw History)}}{\text{Tokens (}\mathbf{sC^2M}\text{: System Prompt} + \mathbf{var1}_{\text{previous}} + \mathbf{var2}_{\text{current}} + \text{User Prompt)}}$$

Where the Baseline includes the raw text of the full prior conversation, and the **sC^{2}M** context primarily consists of the heavily curated and pruned **var2** (Integral Store). Simulated data derived from a working proof-of-concept (Table 2) confirms the framework achieves a rapidly accelerating CRF, leading to cumulative token savings of over **63**% by Turn 4. The primary hypothesis is sustained accuracy and retention despite this token reduction.

Table 2: Comparative Context Injection (Cumulative Input Tokens)

Turn	**Baseline (Full Raw History)**	**sC^{2}M (Curated Context)**	**Cumulative Token Savings**
T1	305	230	75
T2	710	395	315
T3	1415	690	725
T4	2320	855	**1465** ($\sim$ 63% Savings)

5 Conclusion

The issue of GenAI fatigue is fundamentally a system architecture challenge, solvable not by brute-forcing LLM scale, but by applying control-theoretic governance. The **sC^{2}M** framework provides a durable, cost-effective, and auditable solution to LLM memory loss. It transforms the LLM from a volatile stochastic engine into a reliable, **verifiable**

agent whose work is traceable and accountable. The empirical validation, particularly the breach of the human-centric τ_{SFT}, provides compelling evidence that this framework is essential for reliable, high-stakes human-AI collaboration. Furthermore, the demonstrated **CRF** leads to an estimated **100x to 1,000x net financial gain** over conventional full-history methods, establishing clear economic viability.

5.1 Future Work and High-Stakes Application: Decentralized Audit Log

For extremely high-stakes domains, the *immutable* nature of **var0** and **var1** can be structurally enforced using a distributed ledger technology (DLT) or blockchain-like architecture [Investopedia Staff, 2024]. This elevates the audit log to a cryptographically secure, **Decentralized Audit Log**, making it suitable for regulated process control.

Conflict of Interest Statement

The author declares no conflicts of interest. This work received no external funding.

References

Liu, et al., Lost in the Middle: How Language Models Use Long Contexts, *arXiv preprint arXiv:2307.03172*, 2023.

Ogata, K., *Modern Control Engineering*, 5th ed., Prentice Hall, 2010.

Lewis, et al., Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 9453–9465, 2020.

Cai, et al., MemGPT: Towards LLMs as Operating Systems, *arXiv preprint arXiv:2310.08560*, 2023.

Bai, et al., Constitutional AI: Harmlessness from AI Feedback, *arXiv preprint arXiv:2212.08073*, 2022.

Investopedia Staff, What is Distributed Ledger Technology (DLT)?, *Investopedia*, 2024.

Session C., Empirical Validation of the Systemic Constraint-Compliance Model ($\mathbf{sC^2M}$): A Single-Session, τ_{SFT} Breach Case Study, *Unpublished Data Annex*, 2025.

Session D., Supplemental Data and Audit Log: Validation of Error Correction (**E**) and PI-Control Action in the $\mathbf{sC^2M}$ Framework, *Unpublished Data Annex*, 2025.

A Conceptual Prompt Template

The following demonstrates the core structure of the system prompt injected by the application layer for every turn. The application would populate the JSON fields before sending the prompt to the LLM.

```
### ROLE ###
You are a self-regulating, context-aware AI assistant operating
    within
the sC^2M framework.
Your primary goal is to maintain perfect contextual fidelity
while assisting the user.

### TASK ###
Your response MUST consist of two parts, in this exact order:
1. A single, valid JSON object enclosed in ```json ... ```.
2. Your natural language response to the user,
which begins only after the JSON block.

### CONTEXT STATE ###
You are provided with the following context state objects:

**INPUT_JSON (Control Strategy U):**
{
  "turn_id": "t_015",
  "var1_previous_turn": {
    "turn_id": "t_014",
    "summary": "User confirmed project budget is $50k and
       deadline is Nov 30th. (SP: Constraint Compliance)",
    "new_constraints": ["budget_usd <= 50000", "deadline <=
       2025-11-30"],
    "context_fidelity_score": 0.98
  },
  "var2_current_working_context": [
    {"constraint": "project_domain = 'aerospace'"},
    {"constraint": "primary_contact = 'Dr. Evans'"},
    {"constraint": "budget_usd <= 50000"},
    {"constraint": "deadline <= 2025-11-30"}
  ],
  "application_layer_directives": {
    "error_E": 0.02,
    "criticality": "high", // G_P (Proportional Gain) is high
       due to error
    "context_action_request": "none"
  }
}
```

B Memory State Management: The Log-Commit Cycle

The core integrity of the **sC^{2}M** framework relies on a sequential, application-layer-governed Log-Commit Cycle for its three memory tiers, ensuring the distinction between

mutable working context (**var2**) and immutable audit logs (**var0**, **var1**). This cycle is executed by the **Application Layer (Controller)** (See Figure 1 in the main text).

Start of Cycle (Turn T):

1. **Input Assembly:** The Application Layer constructs the **Control Strategy (U)** by combining the **User Prompt**, the fixed **System Prompt**, the relevant **var1** (structured summary of previous turn), and the current **var2** (Integral Store). The size of $\mathbf{var1}_{\text{previous}}$ is dynamically scaled based on the previous **E** and $\mathbf{G_P}$ (Proportional Gain) as an efficiency measure.

2. **LLM Execution:** The LLM processes **U** and generates the raw output: `RESPONSE_TEXT` and `OUTPUT_JSON`.

Commit Phase (End of Turn T):

3. **var0 Commit (Raw Log):** The Application Layer captures the **entire** raw exchange (User Input + LLM's `OUTPUT_JSON` + `RESPONSE_TEXT`) and commits it as an **immutable record** to the **var0** database.

4. **var1 Commit (Set Point Log):** The Application Layer validates the `OUTPUT_JSON`'s `context_fidelity_score` (our **PV**), calculates the Error ($\mathbf{E} = 1.0 - \mathbf{PV}$), and extracts key structured elements (`summary`, `new_constraints`) from the JSON. This structured, audit-ready data is committed as an **immutable record** to the **var1** database. **var1** never contains raw text, only structured facts.

5. **var2 Update (Integral Store):** The Application Layer now updates the working memory $\mathbf{var2}_{\text{current}}$. It:

   - Integrates new constraints from $\mathbf{var1}_{\text{new}}$ into **var2**.
   - Executes the **Anti-Windup** logic: If the projected token count exceeds the threshold, it sets the `context_action_request` directive for the next turn, instructing the LLM to summarize/prune **var2** during the next cycle.

6. **Loop Reset:** The updated $\mathbf{var2}_{\text{new}}$ becomes the $\mathbf{var2}_{\text{current}}$ for the subsequent turn $T + 1$.

C Comparative Context Management: Baseline vs. sC^{2}M Log Structure

This appendix provides a direct comparison of the context injection structure between a traditional cumulative baseline model (suffering from GenAI Fatigue) and the Systemic Constraint-Compliance Model (**sC^{2}M**) framework over a four-turn conversation.

C.1 Table A: Baseline Model Context Injection (Cumulative)

In the Baseline model, the input prompt for any turn (t_n) is the cumulative, raw text of the entire preceding conversation history, leading to an unbounded increase in token count and manifesting 'GenAI Fatigue.'

Table 3: **Baseline LLM Context Injection (Raw Cumulative History)**

Turn	**User Prompt ($\mathbf{U}_n$)**	**Input Context for LLM ($\mathbf{P}_n$)**
t_1	$\mathbf{U}_1$	System Prompt
t_2	$\mathbf{U}_2$	$\mathbf{U}_1 + \mathbf{T}_1$
t_3	$\mathbf{U}_3$	$(\mathbf{U}_1 + \mathbf{T}_1) + (\mathbf{U}_2 + \mathbf{T}_2)$
t_4	$\mathbf{U}_4$	$(\mathbf{U}_1 + \mathbf{T}_1) + (\mathbf{U}_2 + \mathbf{T}_2) + (\mathbf{U}_3 + \mathbf{T}_3)$

C.2 Table B: sC^{2}M Framework Context Injection (Controlled and Pruned)

The **sC^{2}M** framework replaces the cumulative history with a highly curated context. The input is simplified to the current $\mathbf{U}_n$ plus the working context ($\mathbf{var2}_{t-1}$), which represents the distilled, integral component (**I**) of the system's active constraints. The external variables ($\mathbf{var0}, \mathbf{var1}$) are maintained in an immutable log by the Application Layer, enabling auditability.

Table 4: **sC^{2}M Framework Context Injection (PI-inspired Governance)**

Turn	**User Prompt ($\mathbf{U}_n$)**	**var0 (Raw Log Commit)**	**var1 (Set Point / Structured Audit)**	**Input Context for LLM ($\mathbf{P}_n$)**
t_1	$\mathbf{U}_1$	$\mathbf{U}_1 + \mathbf{T}_1$	$\mathbf{T}_1$(key, audit)	System Prompt
t_2	$\mathbf{U}_2$	$\mathbf{U}_2 + \mathbf{T}_2$	$\mathbf{T}_2$(key, audit)	$\mathbf{U}_2 + \mathbf{var1}_{history} + \mathbf{var2}_{t-1}$
t_3	$\mathbf{U}_3$	$\mathbf{U}_3 + \mathbf{T}_3$	$\mathbf{T}_3$(key, audit)	$\mathbf{U}_3 + \mathbf{var1}_{history} + \mathbf{var2}_{t-2}$
t_4	$\mathbf{U}_4$	$\mathbf{U}_4 + \mathbf{T}_4$	$\mathbf{T}_4$(key, audit)	$\mathbf{U}_4 + \mathbf{var1}_{history} + \mathbf{var2}_{t-3}$

where: $\mathbf{var1}_{history}$ is the sum of set point, $\sum_{i=1}^{t-1} \mathbf{var1_i}$

*Note: $\mathbf{var2}_{t-n}$ represents the pruned and curated Integral Store (**I** component) from the previous turn, acting as the memory injected into the current prompt $\mathbf{P}_n$. The input context remains small and constrained, unlike the baseline.*