

Beyond Hard Writes and Rigid Preservation: Soft Recursive Least-Squares for Lifelong LLM Editing

Xinyu Wang^{1,3*}, Sicheng Lyu^{1,2,3*}, Yu Gu^{1*}, Jerry Huang^{2,4}, Peng Lu⁴,
Yufei Cui^{1,2}, Xiao-Wen Chang^{1†}

¹McGill University, ²Mila—Quebec AI Institute, ³SimpleWay.AI, ⁴Université de Montréal
 {xinyu.wang5, sicheng.lyu, yu.gu4}@mail.mcgill.ca, chang@cs.mcgill.ca

Abstract

Model editing updates a pre-trained LLM with new facts or rules without re-training, while preserving unrelated behavior. In real deployment, edits arrive as long streams, and existing editors often face a *plasticity–stability dilemma*: locate-then-edit “hard writes” can accumulate interference over time, while null-space-style “hard preservation” preserves only what is explicitly constrained, so past edits can be overwritten and unconstrained behaviors may deviate, degrading general capabilities in the many-edits regime.

We propose **RLSEdit**, a recursive least-squares editor for long sequential editing. RLSEdit formulates editing as an online quadratic optimization with *soft constraints*, minimizing a cumulative key-value fitting objective with two regularizers that control for both deviation from the pre-trained weights and from a designated anchor mapping. The resulting update admits an efficient online recursion via the Woodbury identity, with per-edit cost independent of history length and scaling only with the current edit size. We further provide deviation bounds and an asymptotic characterization of the adherence–preservation trade-off in the many-edits regime.

Experiments on multiple model families demonstrate stable scaling to 10K edits, outperforming strong baselines in both edit success and holistic stability – crucially retaining early edits, and preserving general capabilities on GLUE and held-out reasoning/code benchmarks.

Code is available at <https://github.com/Euphoria040201/RLSEdit>

1 Introduction

Despite the large amount of knowledge they store within their parameters, large language models (LLMs) [Yang

et al., 2024, OpenAI, 2023, DeepSeek-AI et al., 2025] inevitably contain outdated, incomplete, or incorrect knowledge [De Cao et al., 2021, Mitchell et al., 2022] when statically deployed without re-training or access to external knowledge bases. Due to the large computational cost incurred if re-training from scratch, many applications necessitate updating models using *edits* to a subset of parameters in order to integrate new facts or rules while preserving general model behavior [Meng et al., 2022]. While early model editors largely focused on single or small-batch updates, practical deployments are inherently sequential: edits arrive continuously, with the editor remaining reliable after each edit [Hartvigsen et al., 2023, Gupta et al., 2024].

The many-edits regime presents a dilemma for models. To remain useful over long streams, they must both memorize the information in the stream, all the while preserving the knowledge previously acquired within it. In practice, failures often manifest as two coupled forms of forgetting:

1. *Retroactive Edit Forgetting*: Future edits can overwrite ones applied in the past.
2. *General-Ability Degradation*: Edits lead to deterioration of out-of-scope reasoning and language understanding.

Existing sequential editors typically fail for complementary reasons. *Locate-then-edit* approaches [Meng et al., 2022, 2023] perform *hard writes*, forcing the model to learn the new associations with little regard for previously contained knowledge and potentially overwriting it. As the number of edits accumulates, so does the number of overwrites, leading to instability and retroactive forgetting of previously learnt facts. Conversely, *null-space* editors [Fang et al., 2025, Sun et al., 2025] projects updates into a feasible subspace conditioned on an anchor mapping, preserves associations defined by it. However, edits are repeatedly applied; maintaining complete knowledge leads to a growing set of constraints to be satisfied, leading to difficulty in satisfying them all, while loosening them can lead to possible degradation in general abilities or retention of facts. As a consequence, neither paradigm is sufficient for repeated or sequential editing.

Alternatively, one can view sequential editing as *online regularized least-squares* on a layer-wise key-value surrogate [Sayed, 2003]. Each edit contributes a quadratic fitting term, with preservation achieved through two explicit *deviation controllers*: one accounting for deviation from the ini-

*Equal contribution.

†Corresponding author.

tial model, and another penalizing for deviation from a designated *anchor mapping*. This yields a soft constraint formulation: learning new edits and preserving previous knowledge are optimized within a single objective, avoiding hard constraints while ensuring long-run stability. From this, we propose **RLSEdit**, a recursive least-squares editor for long sequential editing that uses a quadratic objective to admit an efficient online recursion via the Woodbury identity [Sherman and Morrison, 1950, Woodbury, 1950, Hager, 1989]. RLSEdit scales *independently of the number of prior edits* and instead with the current edit rank/size, with theoretical deviation bounds and an asymptotic characterization that highlights the suitability of **RLSEdit** for long streams of edits. To summarize our main contributions, we:

- Motivate a **soft-constraint long-sequential editing framework** by formulating lifelong editing as online regularized least squares with explicit parameter-deviation and anchor-deviation controls, systematically interpolating between the “hard write” and “hard preserve” extremes.
- Derive a **Woodbury-based online update** with per-edit cost independent of past edit count.
- Justify stability and preservation of information by deriving bounds and an asymptotic characterization.
- Provide empirical evidence on Llama-3 [Dubey et al., 2024] and Qwen2.5 [Yang et al., 2024] after 10K edits to show **stronger edit success, improved early-edit retention, and better preservation of general capabilities** on held-out reasoning and code benchmarks.

2 Related Work

We review model editing methods through a *soft versus hard constraint* lens. We would like to explain how different editing methods balance between **effectively making new edits** and **preserving previous edits and knowledge**, as well as explain why long-sequential editing can be challenging. In particular, we consider **layer-wise input-output pairs**, where we edit a single linear map in layer ℓ (e.g., an attention projection). Given an input prompt, we run a forward pass and collect a set of module-level *input-output* feature pairs (\mathbf{k}, \mathbf{v}) at selected token positions, where $\mathbf{k} \in \mathbb{R}^{d_k}$ is the input activation to the edited map and $\mathbf{v} \in \mathbb{R}^{d_v}$ is the corresponding output activation. For the t -th edit, we stack u_t such pairs to form $\mathbf{K}_t \in \mathbb{R}^{u_t \times d_k}$ and $\mathbf{V}_t \in \mathbb{R}^{u_t \times d_v}$.

The Writing and Preservation Trade-off. The goal of editing is to perform targeted updates to the model parameters to learn new key-value associations. With soft or no direct constraint on how this changes the parameters, this can be expressed as a constrained LS problem:

$$\widehat{\mathbf{W}} \in \arg \min_{\mathbf{W}} \|\mathbf{K}\mathbf{W} - \mathbf{V}\|_F^2 \quad \text{s.t.} \quad \mathbf{K}^* \mathbf{W} = \mathbf{V}^*. \quad (1)$$

Here (\mathbf{K}, \mathbf{V}) denotes the current key-value associations contained within the model parameters, and $(\mathbf{K}^*, \mathbf{V}^*)$ denotes the edit constraints defined by the new set of edits.

Alternatively, one can attempt to preserve greater amounts of existing knowledge by restricting updates to a feasible subspace so that performance on a *designated preservation set*

(e.g., an anchor/background mapping) remains unchanged. This can be expressed as

$$\min_{\Delta_t} \|\mathbf{K}_t(\mathbf{W}_{t-1} + \Delta_t) - \mathbf{V}_t\|_F^2 \quad (+\lambda R(\cdot)) \quad \text{s.t.} \quad \mathbf{K}_{\text{pres},t} \Delta_t = 0. \quad (2)$$

This leads to a *soft* update to fit the model parameters to satisfy the preservation constraints. In ALPHAEDIT [Fang et al., 2025], $\mathbf{K}_{\text{pres},t}$ is typically fixed to a chosen anchor/background set, preserving only what is explicitly constrained. LANGEDIT [Sun et al., 2025] retains the same principle but updates multilingual preservation statistics online, so that $\mathbf{K}_{\text{pres},t}$ (and the induced projector) evolves over time.

Moving to Longer Edit Streams. When only a single batch of edits needs to be applied, existing editing methods have shown strong performance. However, such settings are not fully representative of the real-world use cases of LLMs. Models often exist in environments where they are ideally deployed for significant periods of time, where the number of edits that need to be applied repeatedly is large. In such longer edit streams, existing methods can suffer for a multitude of reasons: hard satisfaction of each batch can induce growing interference between the incoming edits and prior knowledge, leading to retroactive forgetting and degradation in out-of-scope performance, while attempting to preserve too much prior knowledge can lead to insufficient learning of new associations.

Soft Updates with Preservation. MEMIT [Meng et al., 2023] uses a soft LS objective over a batch of past and new associations, but is not formulated as a long-stream sequential objective. To address the long-edit stream setting, we enforce *soft* adherence and preservation within a quadratic objective that encompasses both hard regimes as limiting cases.

Beyond Direct Parameter Writes. Several recent lines are orthogonal to direct streaming parameter updates. ANYEDIT [Jiang et al., 2025] broadens the scope of editable knowledge beyond simple factual statements, while UNKE [Deng et al., 2025] targets unstructured knowledge. For lifelong settings, WISE [Wang et al., 2024] separates edited knowledge from pre-trained knowledge via dual memories and routing, and RECIPE [Chen et al., 2024] externalizes updates as retrieval-augmented continuous prompts with gating. These approaches trade additional components (memory/retrieval/prompting) for scalability and locality, and are complementary to our focus on an efficient, single-objective streaming parameter editor.

3 Methodology

We present our editing framework in three parts. We first introduce a recursive least-squares (RLS) formulation that adds up all editing residuals quadratically in the objective function, with penalties over deviation from both the initial model parameters and anchor mapping (Section 3.1). We then analyze the computational complexity of the resulting updates and compare them with existing sequential editors under a long-edit stream (Section 3.2). Finally, we provide a unified perspective by contrasting *soft* and *hard* constraint designs. This shows how existing editors act as special cases of our formulation (Section 3.3).

3.1 A recursive least squares editor

Setup We consider editing a single linear map $\mathbf{W} \in \mathbb{R}^{d_k \times d_v}$ (e.g., a projection matrix in a transformer layer). Each edit provides a set of key-value constraints $(\mathbf{K}_t, \mathbf{V}_t)$ where $\mathbf{K}_t \in \mathbb{R}^{u_t \times d_k}$ and $\mathbf{V}_t \in \mathbb{R}^{u_t \times d_v}$ with u_t being the number of contexts collected for the t -th edit. Layer-wise edit adherence is measured by the residual $\|\mathbf{K}_t \mathbf{W} - \mathbf{V}_t\|_F$. The goal of our method is two-fold. First, it should incorporate *all* edits up to time t . Second, deviation from the initial weights (i.e. \mathbf{W}_0) and from the anchor pairs (e.g. $(\mathbf{K}_0, \mathbf{V}_0)$) should be controlled, thus we introduce two regularization terms to enforce these constraints.

Regularized Least-Squares Equation

At time t , our objective is to obtain the optimal weight \mathbf{W}_t^* that minimizes:

$$\begin{aligned} \mathbf{W}_t^* := \arg \min_{\mathbf{W}} & \sum_{i=1}^t \|\mathbf{K}_i \mathbf{W} - \mathbf{V}_i\|_F^2 \\ & + \lambda^2 \|\mathbf{W} - \mathbf{W}_0\|_F^2 \\ & + \mu^2 \|\mathbf{K}_0 \mathbf{W} - \mathbf{V}_0\|_F^2, \end{aligned} \quad (3)$$

where $\|\mathbf{W} - \mathbf{W}_0\|_F^2$ and $\|\mathbf{K}_0 \mathbf{W} - \mathbf{V}_0\|_F^2$ are the regularization terms, λ and μ are hyperparameters.

To find the minimizer, define

$$\begin{aligned} \mathbf{A}_t &= [\lambda \mathbf{I}_{d_k}; \mu \mathbf{K}_0^\top; \mathbf{K}_1^\top; \dots; \mathbf{K}_t^\top]^\top, \\ \mathbf{B}_t &= [\lambda \mathbf{W}_0^\top; \mu \mathbf{V}_0^\top; \mathbf{V}_1^\top; \dots; \mathbf{V}_t^\top]^\top. \end{aligned} \quad (4)$$

Then we could reformulate Equation (3) to an equivalent form:

$$\mathbf{W}_t^* = \arg \min_{\mathbf{W}} \|\mathbf{A}_t \mathbf{W} - \mathbf{B}_t\|_F^2. \quad (5)$$

To derive a closed form solution to Equation (5), first let

$$\mathbf{S}_t := \mathbf{A}_t^\top \mathbf{A}_t = \lambda^2 \mathbf{I} + \mu^2 \mathbf{K}_0^\top \mathbf{K}_0 + \sum_{i=1}^t \mathbf{K}_i^\top \mathbf{K}_i, \quad (6)$$

$$\mathbf{T}_t := \mathbf{A}_t^\top \mathbf{B}_t = \lambda^2 \mathbf{W}_0 + \mu^2 \mathbf{K}_0^\top \mathbf{V}_0 + \sum_{i=1}^t \mathbf{K}_i^\top \mathbf{V}_i. \quad (7)$$

When $\lambda > 0$, we have $\mathbf{A}_t^\top \mathbf{A}_t \succ 0$, and Equation (5) admits a unique closed-form least-squares solution given by the normal equations $\mathbf{A}_t^\top \mathbf{A}_t \mathbf{W} = \mathbf{A}_t^\top \mathbf{B}_t$. Hence, the solution is given by

$$\mathbf{W}_t^* = \mathbf{S}_t^{-1} \mathbf{T}_t. \quad (8)$$

By jointly solving the unified objective Equation (5), this minimizer not only update the current knowledge pairs $(\mathbf{K}_t, \mathbf{V}_t)$, but also force the solution close to the original weights \mathbf{W}_0 and ensure the anchor mappings $\mathbf{K}_0 \mathbf{W} \mapsto \mathbf{V}_0$.

Efficient Recursion via Normal Equations

Direct computation of Equation (8) requires obtaining the inverse of matrix \mathbf{S}_t , which is expensive in practice. Therefore, we develop an efficient recursive solution. From Equation (8), the minimizer \mathbf{W}_t^* satisfies

$$(\mathbf{A}_t^\top \mathbf{A}_t) \mathbf{W}_t^* = \mathbf{A}_t^\top \mathbf{B}_t. \quad (9)$$

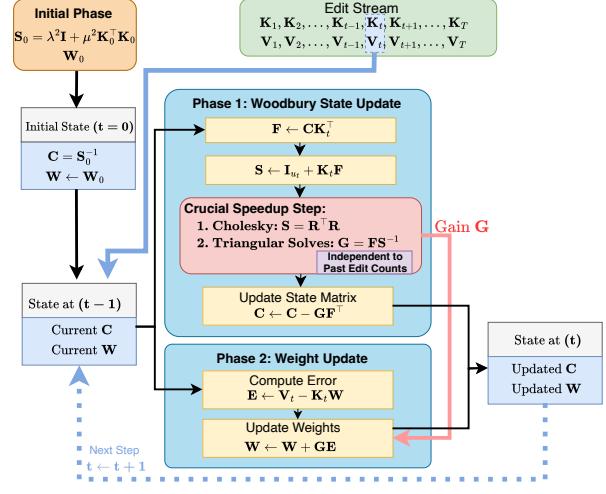


Figure 1: **The recursive workflow of our RLS-Woodbury editor.** The process alternates between updating the covariance state via the Woodbury identity (Phase 1) and updating weights (Phase 2). The highlighted block shows how we reduce complexity from $O(d_k^3)$ to $O(d_k^2 u_t)$ by solving small $u_t \times u_t$ systems.

If we define \mathbf{C}_t as the inverse of \mathbf{S}_t and using Equation (6),

$$\mathbf{C}_t^{-1} = \mathbf{C}_{t-1}^{-1} + \mathbf{K}_t^\top \mathbf{K}_t. \quad (10)$$

Next, let

$$\mathbf{F}_t := \mathbf{C}_{t-1} \mathbf{K}_t^\top \in \mathbb{R}^{d_k \times u_t}.$$

By the Sherman-Morrison-Woodbury identity,

$$\mathbf{C}_t = \mathbf{C}_{t-1} - \mathbf{F}_t (\mathbf{I}_{u_t} + \mathbf{K}_t \mathbf{F}_t)^{-1} \mathbf{F}_t^\top. \quad (11)$$

A numerically stable and efficient implementation is obtained by a Cholesky factorization $\mathbf{I}_{u_t} + \mathbf{K}_t \mathbf{F}_t = \mathbf{R}_t^\top \mathbf{R}_t$ and triangular solves, avoiding explicit inverses. From Equation (7), a normal-equation manipulation yields the final form of **RLSEdit** at time step t :

$$\mathbf{W}_t^* = \mathbf{W}_{t-1}^* + \mathbf{C}_t \mathbf{K}_t^\top (\mathbf{V}_t - \mathbf{K}_t \mathbf{W}_{t-1}^*), \quad (12)$$

where $\mathbf{R}_t := \mathbf{V}_t - \mathbf{K}_t \mathbf{W}_{t-1}^*$. As a result, each edit requires only (i) updating \mathbf{C}_t via Equation (11), and (ii) updating \mathbf{W}_t^* via Equation (12).

3.2 Complexity analysis

We report the per-edit cost at step t . Multiplying $\mathbf{M} \in \mathbb{R}^{m \times n}$ and $\mathbf{N} \in \mathbb{R}^{n \times p}$ costs $O(mnp)$, and solving a dense $n \times n$ linear system costs $O(n^3)$.

RLS-Woodbury Updates.

RLSEdit maintains $\mathbf{C}_t = \mathbf{S}_t^{-1} \in \mathbb{R}^{d_k \times d_k}$ and updates it via Woodbury using

$$\mathbf{F}_t = \mathbf{C}_{t-1} \mathbf{K}_t^\top \in \mathbb{R}^{d_k \times u_t}, \quad \mathbf{S}_t = \mathbf{I}_{u_t} + \mathbf{K}_t \mathbf{F}_t \in \mathbb{R}^{u_t \times u_t}.$$

The covariance-state update is dominated by forming these products and solving the resulting $u_t \times u_t$ system, yielding

$$(1) \text{ Covariance update: } O(d_k^2 u_t) + O(u_t^3).$$

Algorithm 1 RLS-Woodbury Editing

Require: Initial weight \mathbf{W}_0 ; Anchor pair $(\mathbf{K}_0, \mathbf{V}_0)$; Penalties (λ, μ) ; Edit stream $\{(\mathbf{K}_t, \mathbf{V}_t)\}_{t=1}^T$.
Ensure: Edited weights $\{\mathbf{W}_t\}_{t=1}^T$ (optional: states $\{\mathbf{C}_t\}$).

- 1: $\mathbf{S}_0 \leftarrow \lambda^2 \mathbf{I}_{d_k} + \mu^2 \mathbf{K}_0^\top \mathbf{K}_0$
- 2: $\mathbf{R}_0 = \mathbf{R}_0^\top \mathbf{R}_0$ \triangleright Cholesky factor \mathbf{R}_0 upper triangular
- 3: $\mathbf{C}_0 \leftarrow \mathbf{S}_0^{-1}$ \triangleright Via triangular solves using \mathbf{R}_0
- 4: $\mathbf{W}_0 \leftarrow \mathbf{W}_0$ \triangleright Initialize current weight estimate
- 5: **for** $t = 1, 2, \dots, T$ **do**
- (1) **Covariance update (Woodbury)**
- 6: $\mathbf{F}_t \leftarrow \mathbf{C}_{t-1} \mathbf{K}_t^\top$ $\triangleright \mathbf{F}_t \in \mathbb{R}^{d_k \times u_t}$
- 7: $\mathbf{S}_t \leftarrow \mathbf{I}_{u_t} + \mathbf{F}_t \mathbf{F}_t^\top$ $\triangleright \mathbf{S}_t \in \mathbb{R}^{u_t \times u_t}$
- 8: $\mathbf{R}_t = \mathbf{R}_t^\top \mathbf{R}_t$ \triangleright Cholesky factor \mathbf{R}_t upper triangular
- 9: $\mathbf{Y}_t \leftarrow \mathbf{F}_t \mathbf{R}_t^{-1}$ \triangleright Triangular solve: $\mathbf{Y}_t = \mathbf{F}_t \mathbf{R}_t^{-1}$
- 10: $\mathbf{C}_t \leftarrow \mathbf{C}_{t-1} - \mathbf{Y}_t \mathbf{Y}_t^\top$ \triangleright Update inverse covariance
- (2) **Weight update**
- 11: $\mathbf{E}_t \leftarrow \mathbf{V}_t - \mathbf{K}_t \mathbf{W}_{t-1}$ \triangleright Prediction error for edit t
- 12: $\mathbf{G}_t \leftarrow \mathbf{F}_t \mathbf{S}_t^{-1}$ \triangleright Gain matrix, reusing \mathbf{S}_t (via triangular solves)
- 13: $\mathbf{W}_t \leftarrow \mathbf{W}_{t-1} + \mathbf{G}_t \mathbf{E}_t$ \triangleright Apply correction to weights
- 14: **end for**
- 15: **return** \mathbf{W}_T (and \mathbf{C}_T)

For the weight update, we reuse the same $u_t \times u_t$ solve to apply the gain $\mathbf{G}_t = \mathbf{C}_t \mathbf{K}_t^\top \in \mathbb{R}^{d_k \times u_t}$ and update \mathbf{W}_t using the residual \mathbf{E}_t . This step is dominated by the key-value multiplication against d_v outputs, giving

$$(2) \text{Weight update: } O(d_k d_v u_t) + O(d_k u_t^2).$$

Overall, the per-edit runtime is therefore

$$(\text{Per>Edit}) \quad O(d_k^2 u_t + d_k d_v u_t + u_t^3),$$

which simplifies to $O(d_k^2 u_t + d_k d_v u_t)$ when $u_t \ll d_k, d_v$.

Comparison to other sequential editors.

For a fair long-sequential comparison, we focus on existing sequential editors. ALPHAEDIT introduces *hard preservation* by projecting the change of weight onto the null space of a fixed *preserved-knowledge* set (denoted by \mathbf{K}_0), i.e., it applies a projector \mathbf{P} (e.g., $\mathbf{P} = I - \mathbf{Q}\mathbf{Q}^\top$) so that the projected update does not affect $\mathbf{K}_0 \mathbf{W}$. In sequential editing, it additionally regularizes against disrupting *previously updated knowledge* represented by $(\mathbf{K}_p, \mathbf{V}_p)$. The resulting closed-form update can be written as

$$\Delta_t = \mathbf{R}_t \mathbf{K}_t^\top \mathbf{P} \left(\mathbf{K}_p \mathbf{K}_p^\top \mathbf{P} + \mathbf{K}_t \mathbf{K}_t^\top \mathbf{P} + I \right)^{-1},$$

and the corresponding baseline (e.g., MEMIT in sequential setting) removes \mathbf{P} and adds $\mathbf{K}_0 \mathbf{K}_0^\top$ inside the inverse. Let m_{t-1} denote the number of previously updated pairs accumulated in \mathbf{K}_p and let u_t denote the number of pairs in the current edit ($\mathbf{K}_t \in \mathbb{R}^{u_t \times d_k}$). The dominant cost in ALPHAEDIT is inverting the dense $d_k \times d_k$ matrix $\mathbf{M}_t := \mathbf{K}_p \mathbf{K}_p^\top \mathbf{P} + \mathbf{K}_t \mathbf{K}_t^\top \mathbf{P} + I$, which costs $O(d_k^3)$ per edit. Forming the two Gram terms costs $O(d_k^2(m_{t-1} + u_t))$, and the remaining multiplications are lower order. Hence,

$$(\text{Per>Edit}) \quad O(d_k^3) + O(d_k^2(m_{t-1} + u_t)).$$

In contrast, RLSEdit avoids any $O(d_k^3)$ factorization during the edit stream by maintaining $\mathbf{C}_t = (\mathbf{A}_t^\top \mathbf{A}_t)^{-1}$ and using a Woodbury recursion, requiring only a $u_t \times u_t$ Cholesky per edit, which depends only on the current edit size u_t (typically $u_t \ll d_k$) and is therefore substantially more efficient in practical long-edit tasks, as shown in Table 2.

3.3 Hard versus Soft Constraints

(I) Versus locate-then-edit editors. As reviewed in Section 2, editors such as ROME and MEMIT are one-shot (or batched) key-value *writes*: they find a single edited weight $\widehat{\mathbf{W}}$ by fitting a LS objective on a background set while enforcing the new associations exactly.

ROME assumes the pre-trained weight \mathbf{W} to be a LS fit on background pairs $(\mathbf{K}_{\text{bg}}, \mathbf{V}_{\text{bg}})$ and obtains the edited weight by imposing the new edit as a hard constraint:

$$\widehat{\mathbf{W}} = \arg \min_{\mathbf{W}} \|\mathbf{K}_{\text{bg}} \mathbf{W} - \mathbf{V}_{\text{bg}}\|_F^2 \text{ s.t. } \mathbf{K}_{\text{edit}} \mathbf{W} = \mathbf{V}_{\text{edit}}. \quad (13)$$

MEMIT extends this to batched edits by fitting a single LS problem on the background pairs together with the new pairs, but it still outputs one edited weight and is not naturally a sequential method.

(II) Versus null-space editors. Null-space editors (e.g., ALPHAEDIT, LANGEDIT) instead enforce *hard* preservation of a chosen preservation set. Each increment Δ_t is restricted to a feasible subspace, and the current edit is fitted inside that subspace:

$$\begin{aligned} & \min_{\Delta_t} \|\mathbf{K}_t(\mathbf{W}_{t-1} + \Delta_t) - \mathbf{V}_t\|_F^2 \text{ (+regularization)} \\ & \text{s.t. } \mathbf{K}_{\text{pres},t} \Delta_t = 0 \iff \Delta_t \in \text{Null}(\mathbf{K}_{\text{pres},t}). \end{aligned} \quad (14)$$

This hard constraint preserves the specified keys, but the feasible subspace can shrink as t grows. The best feasible update may thus deviate from the unconstrained optimum and limit long-run edit adherence.

Remark 3.1. Write $\widehat{\mathbf{W}} = \mathbf{W} + \Delta$ in Equation (13). Since $\mathbf{K}_{\text{bg}} \mathbf{W} \approx \mathbf{V}_{\text{bg}}$, the ROME update is, up to constants,

$$\min_{\Delta} \|\mathbf{K}_{\text{bg}} \Delta\|_F^2 \text{ s.t. } \mathbf{K}_{\text{edit}} \Delta = \mathbf{V}_{\text{edit}} - \mathbf{K}_{\text{edit}} \mathbf{W},$$

which has a *hard write / soft preserve* structure: the new association is enforced exactly, while background deviation is only softly penalized. Null-space editors reverse this: they preserve the chosen set exactly and fit the current edit softly inside the feasible subspace.

Our RLS editor keeps, assuming finite μ and λ , both sides *soft* via a single cumulative objective:

$$\begin{aligned} \mathbf{W}_t^* = & \arg \min_{\mathbf{W}} \sum_{i=1}^t \|\mathbf{K}_i \mathbf{W} - \mathbf{V}_i\|_F^2 \\ & + \lambda^2 \|\mathbf{W} - \mathbf{W}_0\|_F^2 + \mu^2 \|\mathbf{K}_0 \mathbf{W} - \mathbf{V}_0\|_F^2. \end{aligned} \quad (15)$$

In the limit, our method reduces to these hard-constraint methods. Letting $\mu \rightarrow \infty$ enforces a hard anchor constraint $\mathbf{K}_0 \mathbf{W} = \mathbf{V}_0$. More generally, multiplying selected past fitting terms $\|\mathbf{K}_i \mathbf{W} - \mathbf{V}_i\|_F^2$ by a factor $\rho \rightarrow \infty$ recovers hard

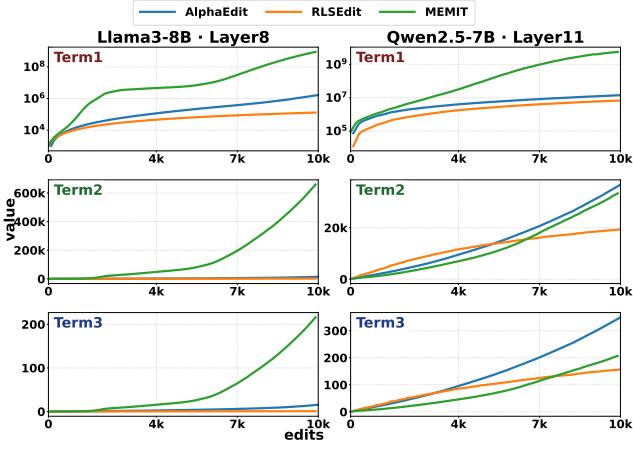


Figure 2: Evolution of objective terms over 10K edits. We compare RLSEdit against baselines (ALPHAEDIT, MEMIT) on three metrics: **Term 1** ($\|K_t \mathbf{W} - V_t\|_F^2$) measures the fitting error for the current edit; **Term 2** ($\|\mathbf{W} - \mathbf{W}_0\|_F^2$) measures parameter drift from the initial weights; and **Term 3** ($\|K_0 \mathbf{W} - V_0\|_F^2$) measures the preservation error on the preserved knowledge. The results show that RLSEdit consistently maintains lower values across all three terms, supporting the stability of our soft-constraint formulation.

preservation constraints (equivalently, a null-space condition) via the standard penalty method. Thus RLSEdit interpolates between the *hard write / soft preserve* extreme and the *hard preserve / soft fit* extreme (null-space editors), while maintaining a stable soft–soft regime in long>Edit streams. **As illustrated in Figure 2**, RLSEdit effectively suppresses the growth of all three objective terms—fitting error, parameter drift, and preservation error—over 10K sequential edits, whereas baselines exhibit instability in at least one component.

4 Theoretical Analysis

We provide deviation bounds in terms of (λ, μ) : λ controls global parameter deviation from \mathbf{W}_0 , and μ controls deviation of the linear mapping $K_0 \mathbf{W}$. Proofs are deferred to the Appendix.

Theorem 4.1 (Global deviation bounds). Let \mathbf{W}_t^* be the minimizer of $J_t(\mathbf{W})$ and define $\mathbf{R}_t := \mathbf{V}_t - K_t \mathbf{W}_{t-1}^*$. Let $\sigma_{\min}(\mathbf{K})$ denotes the smallest singular value of \mathbf{K} .

(i) (Parameter Deviation) If $\lambda > 0$, then for any $T \geq 1$,

$$\|\mathbf{W}_T^* - \mathbf{W}_0\|_F \leq \frac{1}{\lambda^2} \left\| \sum_{t=1}^T \mathbf{K}_t^\top (\mathbf{V}_t - \mathbf{K}_t \mathbf{W}_0) \right\|_F.$$

(ii) (Linear Map Deviation) If $\mu > 0$, then for any $T \geq 1$,

$$\|\mathbf{K}_0(\mathbf{W}_T^* - \mathbf{W}_0)\|_F \leq \frac{1}{\mu} \sum_{t=1}^T \|\mathbf{R}_t\|_F.$$

In addition, the adaptive spectral variant

$$\begin{aligned} & \|\mathbf{K}_0(\mathbf{W}_T^* - \mathbf{W}_0)\|_F \\ & \leq \sum_{t=1}^T \frac{\|\mathbf{K}_0\|_2 \|\mathbf{K}_t\|_2 \|\mathbf{R}_t\|_F}{\lambda^2 + \mu^2 \sigma_{\min}^2(\mathbf{K}_0) + \sum_{i=1}^t \sigma_{\min}^2(\mathbf{K}_i)} \end{aligned}$$

holds with improved uniform-denominator bound.

Theorem 4.1 clarifies how μ and λ affect deviation. $\lambda^{-1/2}$ bounds the movement of the least squared solution \mathbf{W}_T^* away from \mathbf{W}_0 at time T , while μ^{-1} bounds deviation of the linear mapping $\mathbf{K}_0 \mathbf{W}$ from output \mathbf{V}_0 . In practice, one increases λ to reduce parameter deviation and increases μ to reduce anchor mapping deviation. The *edit residual* measuring how well the current edit constraints are satisfied.

4.1 Asymptotic Scaling

To connect (λ, μ) to the many-edits regime, we view Equation (3) as a ridge-type estimator for a *layer-wise* linear mapping. We use the statistical model

$$\mathbf{V}_i = \mathbf{K}_i \mathbf{W}^* + \mathbf{E}_i, \quad \sup_i \mathbb{E} \|\mathbf{E}_i\|_F^2 < \infty, \quad (16)$$

where \mathbf{E}_i captures approximation error due to other layers, context variability, and mismatch between the linear output.

Importantly, sequential edits need not be i.i.d and we only assume long-run stability of second moments, i.e. there exist matrices Σ_k and Σ_{kv} such that

$$\begin{aligned} & \frac{1}{t} \sum_{i=1}^t \mathbf{K}_i^\top \mathbf{K}_i \rightarrow \Sigma_k, \quad \frac{1}{t} \sum_{i=1}^t \mathbf{K}_i^\top \mathbf{V}_i \rightarrow \Sigma_{kv}, \\ & \text{s.t. } \Sigma_k \succ 0 \text{ (on the relevant subspace).} \end{aligned} \quad (17)$$

Allow $\lambda = \lambda_t$ and $\mu = \mu_t$ to depend on t and define

$$\alpha_t := \lambda_t^2/t, \quad \beta_t := \mu_t^2/t.$$

Then the normalized objective at step t is

$$\begin{aligned} \tilde{J}_t(\mathbf{W}) &= \frac{1}{t} \sum_{i=1}^t \|\mathbf{K}_i \mathbf{W} - \mathbf{V}_i\|_F^2 \\ &+ \alpha_t \|\mathbf{W} - \mathbf{W}_0\|_F^2 + \beta_t \|\mathbf{K}_0 \mathbf{W} - \mathbf{V}_0\|_F^2. \end{aligned} \quad (18)$$

For asymptotic analysis, it is convenient to work with the normalized objective $\tilde{J}_t(\mathbf{W}) := J_t(\mathbf{W})/t$, which has the same minimizer as J_t for each fixed t .

Proposition 4.2 (Asymptotic behavior of the RLS editor). Assume Equation (15), Equation (16) and that $\sup_i \mathbb{E} \|\mathbf{K}_i\|_F^4 < \infty$, $\sup_i \mathbb{E} \|\mathbf{V}_i\|_F^4 < \infty$. Let \mathbf{W}_t^* be the minimizer of $J_t(\mathbf{W})$, with $\alpha_t = \lambda_t^2/t$, $\beta_t = \mu_t^2/t$ from Equation (17). Suppose that $\alpha_t \rightarrow \alpha$ and $\beta_t \rightarrow \beta$ for some $\alpha, \beta \in [0, \infty)$ as $t \rightarrow \infty$. We define the population quadratic risk $\mathcal{R}(\mathbf{W}) := \mathbb{E}[\|\mathbf{K}\mathbf{W} - \mathbf{V}\|_F^2]$ under the limiting second-moment model in Equation (17). Then

(i) The normalized objectives \tilde{J}_t converge point-wise to the regularized population risk

$$\begin{aligned} \mathcal{R}_{\text{ridge}}(\mathbf{W}) &:= \mathcal{R}(\mathbf{W}) + \alpha \|\mathbf{W} - \mathbf{W}_0\|_F^2 \\ &+ \beta \|\mathbf{K}_0 \mathbf{W} - \mathbf{V}_0\|_F^2. \end{aligned} \quad (19)$$

Method	Model	Efficacy \uparrow	Generalization \uparrow	Specificity \uparrow	Fluency \uparrow	Consistency \uparrow
RLSEdit (Ours)	Llama-3-8B	89.94\pm0.75	72.84\pm1.21	60.56\pm0.35	615.58\pm4.34	26.27\pm0.35
AlphaEdit		66.78 \pm 3.19	58.27 \pm 1.59	51.79 \pm 0.70	489.91 \pm 33.83	4.59 \pm 0.39
ROME		47.57 \pm 0.10	48.45 \pm 0.33	52.52 \pm 0.44	465.02 \pm 17.88	1.83 \pm 0.14
MEMIT		49.73 \pm 1.44	49.24 \pm 0.48	51.54 \pm 0.68	323.01 \pm 16.40	3.45 \pm 1.62
FT		74.76 \pm 0.00	64.49 \pm 0.00	39.69 \pm 0.00	342.42 \pm 0.20	1.31 \pm 0.00
RLSEdit (Ours)	Qwen2.5-7B	94.45\pm1.07	68.55\pm0.47	73.37\pm0.44	625.74\pm0.71	31.62\pm0.81
AlphaEdit		94.10 \pm 0.42	70.29\pm2.30	75.29\pm0.65	623.51 \pm 0.24	31.37 \pm 0.49
ROME		35.70 \pm 1.36	37.16 \pm 1.19	65.20 \pm 1.42	619.67 \pm 16.98	31.79\pm3.59
MEMIT		53.13 \pm 0.72	51.39 \pm 0.49	51.52 \pm 0.92	532.38 \pm 24.31	1.63 \pm 2.22
FT		65.72 \pm 0.00	56.46 \pm 0.00	45.23 \pm 0.00	324.70 \pm 0.04	1.87 \pm 0.03

Table 1: CounterFact results on Llama-3-8B and Qwen2.5-7B, comparison of **RLSEdit** with the baselines. We report mean \pm standard deviation over 3 random seeds, evaluated on the full CounterFact test set after completing all sequential edits (10K Edits in total, with a batch size of 100). We evaluate on five metrics: Efficacy, Generalization, Specificity, Fluency, and Consistency. The best-performing results are highlighted in bold, and the second-best results are underlined.

(ii) The function $\mathcal{R}_{\text{ridge}}$ is strictly convex and admits a unique minimizer \mathbf{W}^\dagger .

(iii) The RLS editor is consistent for \mathbf{W}^\dagger :

$$\mathbf{W}_t^* \rightarrow \mathbf{W}^\dagger \text{ a.s. } t \rightarrow \infty. \quad (20)$$

If $\alpha = \beta = 0$ (e.g., when λ_t, μ_t are held fixed), then $\mathbf{W}^\dagger = \mathbf{W}^*$ and \mathbf{W}_t^* converges to the least-squares population minimizer. If $\alpha > 0$ and/or $\beta > 0$, then \mathbf{W}^\dagger interpolates between the data-driven optimum \mathbf{W}^* and the anchor constraints encoded by $(\mathbf{W}_0, \mathbf{K}_0, \mathbf{V}_0)$: larger α shrinks \mathbf{W}^\dagger toward \mathbf{W}_0 , and larger β enforces $\mathbf{K}_0 \mathbf{W}^\dagger \approx \mathbf{V}_0$ even as $t \rightarrow \infty$. This proposition shows that our optimized solution weights will be stable and converge to some point with mild conditions. In practice, increasing μ and λ makes the update more conservative. This keeps \mathbf{W}_t^* closer to the original model, but fits the new edit less accurately, leading to larger residuals $\|\mathbf{R}_t\|_F$. A common policy is to set a deviation budget for the associated penalties, then tune (μ, λ) to satisfy both bounds. The limits $\mu, \lambda \rightarrow \infty$ yield hard anchoring ($\mathbf{K}_0 \mathbf{W} = \mathbf{V}_0$) and freezes parameters ($\mathbf{W}_t^* \rightarrow \mathbf{W}_0$).

5 Experiments and Results

5.1 Experimental Setup

Models and Baselines. We conduct experiments with two backbone models, Llama3-8B and Qwen2.5-7B, against ALPHAEDIT [Fang et al., 2025], ROME [Meng et al., 2022], MEMIT [Meng et al., 2023], and fine-tuning (FT) [Zhu et al., 2020].

Datasets and Metrics. Following prior work, we use the CounterFact dataset [Meng et al., 2022]. We report **Efficacy** (rewrite success), **Generalization** (paraphrase success), **Specificity** (neighborhood success), **Fluency** (generation entropy), and **Consistency** (reference score). The detailed hyper-parameter setup is included in Section B.

5.2 Main Results

Editing Results. Table 1 reports performance after 10K edits (batch size 100) with Llama3-8B and Qwen2.5-7B on

the CounterFact dataset, our method **RLSEdit** demonstrates strong overall performance. For Llama-3-8B, **RLSEdit** achieves the best scores across all five metrics. Notably, it shows substantial leads in Efficacy (89.94 vs. 74.76 for second-best FT), Generalization, Fluency, and Consistency. For Qwen2.5-7B, the results are more nuanced. While textbfRLSEdit obtains the highest Efficacy and Fluency, ALPHAEDIT is strongest in Generalization and Specificity, and ROME leads in Consistency. **RLSEdit** and ALPHAEDIT perform comparably on this model, with both significantly outperforming ROME, MEMIT, and FT in most metrics. Overall, these results demonstrate the strong editing effectiveness of **RLSEdit** in long, sequential editing scenarios.

To assess how well editing methods preserve the pre-edited model’s general abilities, we evaluate 5 tasks from GLUE (SST, MMLU, MRPC, CoLa, RTE) [Wang et al., 2018], together with additional benchmarks that test *general knowledge* (MMLU), *math reasoning* (GSM8K), and *coding ability* (HumanEval, MBPP). Details of these benchmarks are provided in Appendix C. We conduct the evaluation of these benchmarks on multiple editing checkpoints of the Llama3-8B model, using 10K total edits with a batch size of 100.

General Capability Results. Figure 3 summarizes the general capability evaluations. Across all language understanding tasks from GLUE and the three code/math reasoning benchmarks, **RLSEdit** consistently delivers the strongest performance throughout the entire editing trajectory. Its stability is especially notable given the scale of the editing workload, maintaining high accuracy even as the number of edits grows large. In contrast, MEMIT, ROME, and FT exhibit rapid degradation as edits accumulate, suggesting limited robustness under sustained modification. ALPHAEDIT performs competitively in the early stages but undergoes a pronounced drop after approximately 8,000 edits, indicating a threshold beyond which its internal representations begin to destabilize. Additional qualitative examples and case studies are provided in the supplementary material (Section D). In

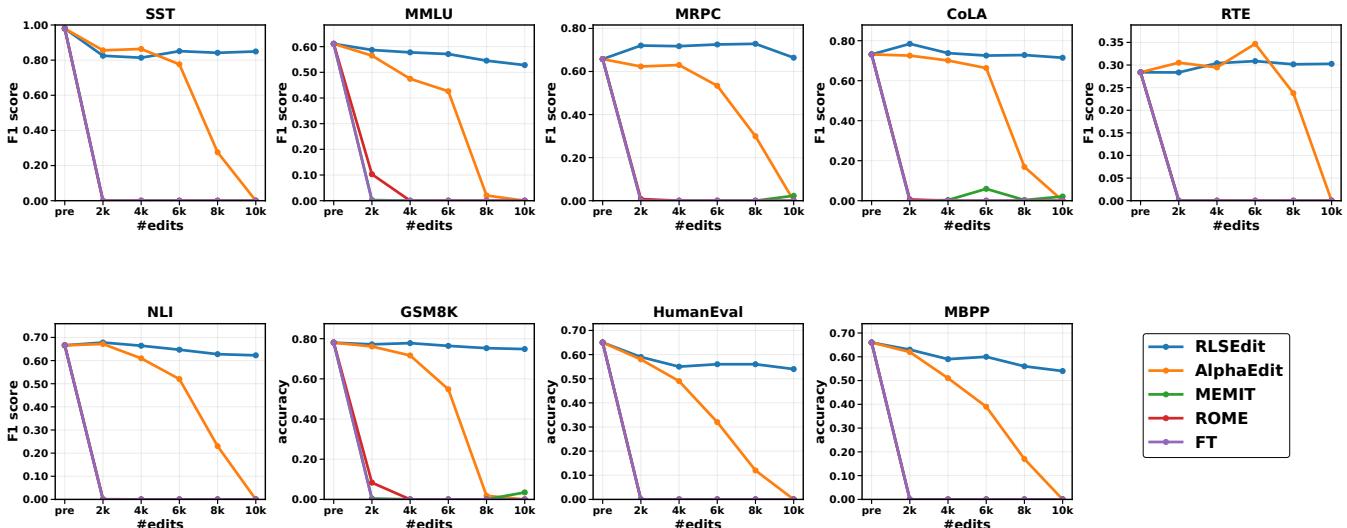


Figure 3: **General capability preservation.** We evaluate 5 GLUE tasks and additional benchmarks for general knowledge, math reasoning and coding ability (MMLU, GSM8K, HumanEval, MBPP) at multiple editing checkpoints (Pre-edit, 2k–10k edits). **RLSEdit** is compared against baselines and consistently better preserves the model’s general capabilities across tasks and edit scales. The x-axis shows the cumulative number of applied edits, and the y-axis reports the corresponding score (F1 or accuracy).

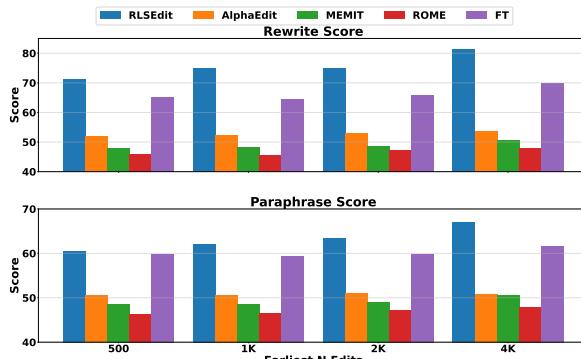


Figure 4: Improvements on early edits. After applying 10K sequential edits, we re-evaluate performance on the earliest edited cases (500, 1K, 2K, 4K). Each bar reports the Rewrite or Paraphrase score. **RLSEdit** consistently achieves the highest scores across all settings.

summary, these results demonstrate that **RLSEdit** more effectively preserves the model’s general language understanding and reasoning abilities while still applying edits reliably and at scale.

5.3 Analysis and Discussion

Early Edits Comparison. To examine how well **RLSEdit** and the baselines preserve earlier edits in a sequential editing setting, we re-evaluate the first N edited cases ($N \in \{500, 1K, 2K, 4K\}$) after performing 10K sequential edits (batch size = 100) on Llama3-8B. As shown in Figure 4, **RLSEdit** consistently achieves the best retention across all N : its Rewrite scores range from 71.22 (at $N=500$) to 81.28 (at $N=4K$), and its Paraphrase scores range from 60.49 to 66.98. In contrast, baseline methods remain noticeably lower (typically around 45 to 70 on Rewrite and 46 to 62 on Paraphrase), suggesting weaker preservation of previously edited knowledge under long, sequential editing.

Method	Llama3-8B			Qwen2.5-7B		
	100	200	500	100	200	500
AlphaEdit	525.15	227.93	108.07	978.32	412.94	197.49
RLSEdit	328.39	166.84	66.85	545.65	271.20	112.88

Table 2: Update time (seconds) for performing 10K edits on Llama3-8B and Qwen2.5-7B using batch sizes $\{100, 200, 500\}$. Lower values indicate faster updates. Comparison of **RLSEdit** versus AlphaEdit.

phrase), suggesting weaker preservation of previously edited knowledge under long, sequential editing.

Speed-up Analysis. Table 2 reports the update computation time for **RLSEdit** and ALPHAEEDIT when performing edits across two model backbones (Llama3-8B and Qwen2.5-7B) and three batch sizes ($BS \in \{100, 200, 500\}$). Across all six configurations, **RLSEdit** consistently runs faster, reducing update time by $1.37 \times$ – $1.79 \times$ relative to ALPHAEEDIT. This empirical advantage is consistent with the theoretical time-complexity analysis presented in Section 3.2.

6 Conclusion

Existing model editing methods suffer from performance loss when the number of edits scales up. To address this, we propose **RLSEdit**, a recursive least-squares framework that implements soft editing and soft preservation targeting long-edit tasks. The main novelty of our method can be summarized into two parts. First, we find an efficient recursive updating algorithm that minimizes the new edit residuals while keeping the old edit residuals small. Second, our formulation is more flexible and generalizable than the existing hard-constraint editing methods by introducing two regularization terms. By

controlling deviation from pre-trained weights and anchors, RLSEdit balances model performance and flexibility while achieving *fast*, constant-time updates via Woodbury recursion formula. Empirically, RLSEdit scales stably to 10K edits on Llama-3 and Qwen2.5, significantly outperforming baselines in *early edits*. Crucially, it preserves both general capabilities and reasoning capabilities in various benchmarks, validating our recursive formulation as a robust solution for continuous model editing.

Ethical Statement

While our work centers on model-editing methods, it is important to acknowledge that such techniques can also be misused to inject undesirable knowledge or behavioral traits into a model. These risks merit careful consideration and discussion.

References

- J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. J. Cai, M. Terry, Q. V. Le, and C. Sutton. Program synthesis with large language models. *ArXiv*, abs/2108.07732, 2021. URL <https://api.semanticscholar.org/CorpusID:237142385>.
- L. Bentivogli, B. Magnini, I. Dagan, H. T. Dang, and D. Giampiccolo. The fifth PASCAL recognizing textual entailment challenge. In *TAC*. NIST, 2009.
- M. Chen, J. Tworek, H. Jun, Q. Yuan, H. Pondé, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. W. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, I. Babuschkin, S. Balaji, S. Jain, A. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba. Evaluating large language models trained on code. *ArXiv*, abs/2107.03374, 2021. URL <https://api.semanticscholar.org/CorpusID:235755472>.
- Q. Chen, T. Zhang, X. He, D. Li, C. Wang, L. Huang, and H. Xue'. Lifelong knowledge editing for llms with retrieval-augmented continuous prompt learning. In Y. Al-Onaizan, M. Bansal, and Y. Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 13565–13580. Association for Computational Linguistics, 2024.
- K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman. Training verifiers to solve math word problems. *ArXiv*, abs/2110.14168, 2021. URL <https://api.semanticscholar.org/CorpusID:239998651>.
- N. De Cao, W. Aziz, and I. Titov. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6491–6506. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.522. URL <https://aclanthology.org/2021.emnlp-main.522/>.
- DeepSeek-AI, D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, X. Zhang, X. Yu, Y. Wu, Z. F. Wu, Z. Gou, Z. Shao, Z. Li, Z. Gao, A. Liu, B. Xue, B. Wang, B. Wu, B. Feng, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, D. Dai, D. Chen, D. Ji, E. Li, F. Lin, F. Dai, F. Luo, G. Hao, G. Chen, G. Li, H. Zhang, H. Bao, H. Xu, H. Wang, H. Ding, H. Xin, H. Gao, H. Qu, H. Li, J. Guo, J. Li, J. Wang, J. Chen, J. Yuan, J. Qiu, J. Li, J. L. Cai, J. Ni, J. Liang, J. Chen, K. Dong, K. Hu, K. Gao, K. Guan, K. Huang, K. Yu, L. Wang, L. Zhang, L. Zhao, L. Wang, L. Zhang, L. Xu, L. Xia, M. Zhang, M. Zhang, M. Tang, M. Li, M. Wang, M. Li, N. Tian, P. Huang, P. Zhang, Q. Wang, Q. Chen, Q. Du, R. Ge, R. Zhang, R. Pan, R. Wang, R. J. Chen, R. L. Jin, R. Chen, S. Lu, S. Zhou, S. Chen, S. Ye, S. Wang, S. Yu, S. Zhou, S. Pan, and S. S. Li. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL <https://doi.org/10.48550/arXiv.2501.12948>.
- J. Deng, Z. Wei, L. Pang, H. Ding, H. Shen, and X. Cheng. Everything is editable: Extend knowledge editing to unstructured data in large language models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL <https://openreview.net/forum?id=X5rO5VytgB>.
- W. B. Dolan and C. Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005. URL <https://aclanthology.org/I05-5002/>.
- A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, A. Goyal, A. Hartshorn, A. Yang, A. Mitra, A. Sravankumar, A. Korenev, A. Hinsvark, A. Rao, A. Zhang, A. Rodriguez, A. Gregerson, A. Spataru, B. Rozière, B. Biron, B. Tang, B. Chern, C. Caucheteux, C. Nayak, C. Bi, C. Marra, C. McConnell, C. Keller, C. Touret, C. Wu, C. Wong, C. C. Ferrer, C. Nikolaidis, D. Allonsius, D. Song, D. Pintz, D. Livshits, D. Esiobu, D. Choudhary, D. Mahajan, D. Garcia-Olano, D. Perino, D. Hupkes, E. Lakomkin, E. AlBadawy, E. Lobanova, E. Diann, E. M. Smith, F. Radenovic, F. Zhang, G. Synnaeve, G. Lee, G. L. Anderson, G. Nail, G. Mialon, G. Pang, G. Cucurell, H. Nguyen, H. Korevaar, H. Xu, H. Touvron, I. Zarov, I. A. Ibarra, I. M. Kloumann, I. Misra, I. Evtimov, J. Copet, J. Lee, J. Geffert, J. Vranes, J. Park, J. Mahadeokar, J. Shah, J. van der Linde, J. Billock, J. Hong, J. Lee, J. Fu, J. Chi, J. Huang, J. Liu, J. Wang, J. Yu, J. Bitton, J. Spisak, J. Park, J. Rocca, J. Johnstun, J. Saxe, J. Jia, K. V. Alwala, K. Upasani, K. Plawiak, K. Li, K. Heafield, K. Stone, and et al. The llama 3 herd of models, 2024. URL <https://doi.org/10.48550/arXiv.2407.21783>.
- J. Fang, H. Jiang, K. Wang, Y. Ma, J. Shi, X. Wang, X. He, and T. Chua. Alphaedit: Null-space constrained

- knowledge editing for language models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.
- A. Gupta, A. Rao, and G. Anumanchipalli. Model editing at scale leads to gradual and catastrophic forgetting. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 15202–15232, Bangkok, Thailand, Aug. 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.902. URL <https://aclanthology.org/2024.findings-acl.902/>.
- W. W. Hager. Updating the inverse of a matrix. *SIAM Review*, 31(2):221–239, 1989. doi: 10.1137/1031049.
- T. Hartvigsen, S. Sankaranarayanan, H. Palangi, Y. Kim, and M. Ghassemi. Aging with GRACE: lifelong model editing with discrete key-value adaptors. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/95b6e2ff961580e03c0a662a63a71812-Abstract-Conference.html.
- D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. X. Song, and J. Steinhardt. Measuring massive multitask language understanding. *ArXiv*, abs/2009.03300, 2020. URL <https://api.semanticscholar.org/CorpusID:221516475>.
- H. Jiang, J. Fang, N. Zhang, M. Wan, G. Ma, X. Wang, X. He, and T. Chua. Anyedit: Edit any knowledge encoded in language models. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*. OpenReview.net, 2025.
- K. Meng, D. Bau, A. Andonian, and Y. Belinkov. Locating and editing factual associations in GPT. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.
- K. Meng, A. S. Sharma, A. J. Andonian, Y. Belinkov, and D. Bau. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.
- E. Mitchell, C. Lin, A. Bosselut, C. Finn, and C. D. Manning. Fast model editing at scale. In *International Conference on Learning Representations (ICLR)*, 2022. URL <https://openreview.net/forum?id=0DcZxeWfOPt>.
- OpenAI. GPT-4 technical report, 2023. URL <https://doi.org/10.48550/arXiv.2303.08774>.
- A. H. Sayed. *Fundamentals of Adaptive Filtering*. Wiley-IEEE Press, 2003. ISBN 978-0471461265.
- J. Sherman and W. J. Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 21(1):124–127, 1950. doi: 10.1214/aoms/1177729893.
- R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In D. Yarowsky, T. Baldwin, A. Korhonen, K. Livescu, and S. Bethard, editors, *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, Oct. 2013. Association for Computational Linguistics. URL <https://aclanthology.org/D13-1170/>.
- W. Sun, T. Qu, M. Li, J. Davis, and M.-F. Moens. Mitigating negative interference in multilingual knowledge editing through null-space constraints. In W. Che, J. Nabende, E. Shutova, and M. T. Pilehvar, editors, *Findings of the Association for Computational Linguistics: ACL 2025*, pages 8796–8810, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.460. URL [https://aclanthology.org/2025.findings-acl.460/](https://aclanthology.org/2025.findings-acl.460).
- A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In T. Linzen, G. Chrupala, and A. Alishahi, editors, *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL [https://aclanthology.org/W18-5446/](https://aclanthology.org/W18-5446).
- P. Wang, Z. Li, N. Zhang, Z. Xu, Y. Yao, Y. Jiang, P. Xie, F. Huang, and H. Chen. WISE: rethinking the knowledge memory for lifelong model editing of large language models. In A. Globersons, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. M. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.
- A. Warstadt, A. Singh, and S. R. Bowman. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019. doi: 10.1162/tacl_a_00290. URL [https://aclanthology.org/Q19-1040/](https://aclanthology.org/Q19-1040).
- A. Williams, N. Nangia, and S. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In M. Walker, H. Ji, and A. Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1101. URL [https://aclanthology.org/N18-1101/](https://aclanthology.org/N18-1101).
- M. A. Woodbury. Inverting modified matrices. Memorandum Report 42, Statistical Research Group, Princeton University, 1950.
- A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei, H. Lin, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Lin, K. Dang, K. Lu, K. Bao, K. Yang, L. Yu, M. Li, M. Xue, P. Zhang, Q. Zhu,

R. Men, R. Lin, T. Li, T. Xia, X. Ren, X. Ren, Y. Fan, Y. Su, Y. Zhang, Y. Wan, Y. Liu, Z. Cui, Z. Zhang, and Z. Qiu. Qwen2.5 technical report, 2024. URL <https://doi.org/10.48550/arXiv.2412.15115>.

C. Zhu, A. S. Rawat, M. Zaheer, S. Bhojanapalli, D. Li, F. X. Yu, and S. Kumar. Modifying memories in transformer models. *ArXiv*, abs/2012.00363, 2020. URL <https://api.semanticscholar.org/CorpusID:227238659>.

A Preliminaries.

Recall the stacked least-squares form

$$\mathbf{A}_t = \begin{bmatrix} \lambda I \\ \mu \mathbf{K}_0 \\ \mathbf{K}_1 \\ \vdots \\ \mathbf{K}_t \end{bmatrix}, \quad \mathbf{B}_t = \begin{bmatrix} \lambda \mathbf{W}_0 \\ \mu \mathbf{V}_0 \\ \mathbf{V}_1 \\ \vdots \\ \mathbf{V}_t \end{bmatrix}, \quad (21)$$

$$\mathbf{W}_t^* = \arg \min_{\mathbf{W}} \|\mathbf{A}_t \mathbf{W} - \mathbf{B}_t\|_F^2, \quad (22)$$

and define the normal-equation matrices

$$\mathbf{S}_t := \mathbf{A}_t^\top \mathbf{A}_t = \lambda^2 I + \mu^2 \mathbf{K}_0^\top \mathbf{K}_0 + \sum_{i=1}^t \mathbf{K}_i^\top \mathbf{K}_i, \quad (23)$$

$$\mathbf{T}_t := \mathbf{A}_t^\top \mathbf{B}_t = \lambda^2 \mathbf{W}_0 + \mu^2 \mathbf{K}_0^\top \mathbf{V}_0 + \sum_{i=1}^t \mathbf{K}_i^\top \mathbf{V}_i. \quad (24)$$

Whenever $\mathbf{S}_t \succ 0$, the minimizer is unique and satisfies

$$\mathbf{W}_t^* = \mathbf{S}_t^{-1} \mathbf{T}_t, \quad \mathbf{C}_t := \mathbf{S}_t^{-1}. \quad (25)$$

We will use the one-step identity (a standard RLS consequence of stacking and normal equations)

$$\mathbf{W}_t^* - \mathbf{W}_{t-1}^* = \mathbf{C}_t \mathbf{K}_t^\top \mathbf{R}_t, \quad \mathbf{R}_t := \mathbf{V}_t - \mathbf{K}_t \mathbf{W}_{t-1}^*. \quad (26)$$

Finally, we assume the anchor is satisfied by the initializer:

$$\mathbf{K}_0 \mathbf{W}_0 = \mathbf{V}_0. \quad (27)$$

Alternative: streaming QR update

For improved numerical stability, one may maintain a QR factorization of \mathbf{A}_t . Assume that at time $t-1$ we have orthogonal transforms

$$\mathbf{Q}_{t-1}^\top \mathbf{A}_{t-1} = \begin{bmatrix} \mathbf{R}_{t-1} \\ \mathbf{0} \end{bmatrix}, \quad \mathbf{Q}_{t-1}^\top \mathbf{B}_{t-1} = \begin{bmatrix} \bar{\mathbf{B}}_{t-1} \\ \hat{\mathbf{B}}_{t-1} \end{bmatrix}, \quad (28)$$

where $\mathbf{R}_{t-1} \in \mathbb{R}^{d_K \times d_K}$ is upper triangular. At time t , we apply additional orthogonal transforms $\bar{\mathbf{Q}}_t$ to

$$\bar{\mathbf{Q}}_t^\top \begin{bmatrix} \mathbf{R}_{t-1} \\ \mathbf{K}_t \end{bmatrix} = \begin{bmatrix} \mathbf{R}_t \\ \mathbf{0} \end{bmatrix}, \quad \bar{\mathbf{Q}}_t^\top \begin{bmatrix} \bar{\mathbf{B}}_{t-1} \\ \mathbf{V}_t \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{B}}_t \\ \hat{\mathbf{B}}_t \end{bmatrix}. \quad (29)$$

Then \mathbf{W}_t^* is obtained by solving the triangular system

$$\mathbf{R}_t \mathbf{W}_t^* = \bar{\mathbf{B}}_t. \quad (30)$$

Initialization. Since $\mathbf{R}_0^\top \mathbf{R}_0 = \lambda^2 I + \mu^2 \mathbf{K}_0^\top \mathbf{K}_0$, we compute \mathbf{R}_0 via Cholesky and set $\bar{\mathbf{B}}_0 = \mathbf{R}_0 \mathbf{W}_0$ (using $\mathbf{K}_0 \mathbf{W}_0 = \mathbf{V}_0$).

A. Proof of Theorem 4.1

Proof of Theorem 4.1(i) (parameter deviation). The normal equations for \mathbf{W}_T^* are

$$\mathbf{S}_T \mathbf{W}_T^* = \mathbf{T}_T, \quad (31)$$

where

$$\mathbf{S}_T = \lambda^2 I + \mu^2 \mathbf{K}_0^\top \mathbf{K}_0 + \sum_{i=1}^T \mathbf{K}_i^\top \mathbf{K}_i, \quad (32)$$

$$\mathbf{T}_T = \lambda^2 \mathbf{W}_0 + \mu^2 \mathbf{K}_0^\top \mathbf{V}_0 + \sum_{i=1}^T \mathbf{K}_i^\top \mathbf{V}_i. \quad (33)$$

Using the anchor condition equation 27, we have

$$\mathbf{K}_0^\top \mathbf{V}_0 = \mathbf{K}_0^\top \mathbf{K}_0 \mathbf{W}_0. \quad (34)$$

Subtracting $\mathbf{S}_T \mathbf{W}_0$ from both sides of equation 31 gives

$$\mathbf{S}_T (\mathbf{W}_T^* - \mathbf{W}_0) = \mathbf{T}_T - \mathbf{S}_T \mathbf{W}_0 = \sum_{i=1}^T \mathbf{K}_i^\top (\mathbf{V}_i - \mathbf{K}_i \mathbf{W}_0). \quad (35)$$

Thus

$$\mathbf{W}_T^* - \mathbf{W}_0 = \mathbf{S}_T^{-1} \sum_{i=1}^T \mathbf{K}_i^\top (\mathbf{V}_i - \mathbf{K}_i \mathbf{W}_0). \quad (36)$$

By submultiplicativity and $\mathbf{S}_T \succeq \lambda^2 I$ (when $\lambda > 0$),

$$\begin{aligned} \|\mathbf{W}_T^* - \mathbf{W}_0\|_F &\leq \|\mathbf{S}_T^{-1}\|_2 \left\| \sum_{i=1}^T \mathbf{K}_i^\top (\mathbf{V}_i - \mathbf{K}_i \mathbf{W}_0) \right\|_F \\ &\leq \frac{1}{\lambda^2} \left\| \sum_{i=1}^T \mathbf{K}_i^\top (\mathbf{V}_i - \mathbf{K}_i \mathbf{W}_0) \right\|_F, \end{aligned} \quad (37)$$

which proves the claim. \square

Lemma A.1. *Assume $\mu > 0$ and $\mathbf{S}_t \succ 0$. Then for each $t \geq 1$,*

$$\|\mathbf{K}_0 (\mathbf{W}_t^* - \mathbf{W}_{t-1}^*)\|_F \leq \frac{1}{\mu} \|\mathbf{R}_t\|_F, \quad (38)$$

$$\|\mathbf{K}_0 (\mathbf{W}_t^* - \mathbf{W}_{t-1}^*)\|_F \leq \|\mathbf{K}_0\|_2 \|\mathbf{K}_t\|_2 \|\mathbf{C}_t\|_2 \|\mathbf{R}_t\|_F. \quad (39)$$

Moreover,

$$\|\mathbf{C}_t\|_2 \leq \frac{1}{\lambda^2 + \mu^2 \Sigma_{\min}^2(\mathbf{K}_0) + \sum_{i=1}^t \Sigma_{\min}^2(\mathbf{K}_i)}. \quad (40)$$

Proof. From equation 26,

$$\mathbf{K}_0 (\mathbf{W}_t^* - \mathbf{W}_{t-1}^*) = \mathbf{K}_0 \mathbf{C}_t \mathbf{K}_t^\top \mathbf{R}_t. \quad (41)$$

The classical bound equation 39 follows from operator norm submultiplicativity:

$$\|\mathbf{K}_0 \mathbf{C}_t \mathbf{K}_t^\top \mathbf{R}_t\|_F \leq \|\mathbf{K}_0\|_2 \|\mathbf{C}_t\|_2 \|\mathbf{K}_t\|_2 \|\mathbf{R}_t\|_F. \quad (42)$$

For the tighter bound equation 38, consider

$$\begin{aligned} \|\mathbf{K}_0 \mathbf{C}_t \mathbf{K}_t^\top \mathbf{R}_t\|_F^2 &= \text{tr}(\mathbf{R}_t^\top \mathbf{K}_t \mathbf{C}_t \mathbf{K}_0^\top \mathbf{K}_0 \mathbf{C}_t \mathbf{K}_t^\top \mathbf{R}_t) \\ &\leq \|\mathbf{K}_t \mathbf{C}_t \mathbf{K}_0^\top \mathbf{K}_0 \mathbf{C}_t \mathbf{K}_t^\top\|_2 \|\mathbf{R}_t\|_F^2. \end{aligned} \quad (43)$$

Using $\|\mathbf{M} \mathbf{N} \mathbf{M}^\top\|_2 \leq \|\mathbf{M}\|_2^2 \|\mathbf{N}\|_2$ with $\mathbf{M} = \mathbf{K}_t \mathbf{C}_t^{1/2}$ and $\mathbf{N} = \mathbf{C}_t^{1/2} \mathbf{K}_0^\top \mathbf{K}_0 \mathbf{C}_t^{1/2}$,

$$\|\mathbf{K}_t \mathbf{C}_t \mathbf{K}_0^\top \mathbf{K}_0 \mathbf{C}_t \mathbf{K}_t^\top\|_2 \leq \|\mathbf{K}_t \mathbf{C}_t \mathbf{K}_t^\top\|_2 \cdot \|\mathbf{K}_0 \mathbf{C}_t \mathbf{K}_0^\top\|_2. \quad (44)$$

We bound the two factors.

(a) $\|\mathbf{K}_t \mathbf{C}_t \mathbf{K}_t^\top\|_2 \leq 1$. Let $\mathbf{C}_{t-1} := \mathbf{S}_{t-1}^{-1}$ and define

$$\mathbf{H}_t := \mathbf{K}_t \mathbf{C}_{t-1} \mathbf{K}_t^\top \succeq 0. \quad (45)$$

By Sherman–Morrison–Woodbury,

$$\mathbf{C}_t = \mathbf{C}_{t-1} - \mathbf{C}_{t-1} \mathbf{K}_t^\top (I + \mathbf{H}_t)^{-1} \mathbf{K}_t \mathbf{C}_{t-1}. \quad (46)$$

Hence,

$$\mathbf{K}_t \mathbf{C}_t \mathbf{K}_t^\top = \mathbf{H}_t - \mathbf{H}_t (I + \mathbf{H}_t)^{-1} \mathbf{H}_t = \mathbf{H}_t (I + \mathbf{H}_t)^{-1}. \quad (47)$$

The eigenvalues of $\mathbf{H}_t (I + \mathbf{H}_t)^{-1}$ are $h/(1+h) \in [0, 1)$ for $h \geq 0$, so $\|\mathbf{K}_t \mathbf{C}_t \mathbf{K}_t^\top\|_2 \leq 1$.

(b) $\|\mathbf{K}_0 \mathbf{C}_t \mathbf{K}_0^\top\|_2 \leq 1/\mu^2$. Since $\mathbf{S}_t \succeq \mu^2 \mathbf{K}_0^\top \mathbf{K}_0$, we have $\mathbf{C}_t = \mathbf{S}_t^{-1} \preceq (\mu^2 \mathbf{K}_0^\top \mathbf{K}_0)^\dagger$ on the support of $\mathbf{K}_0^\top \mathbf{K}_0$, hence

$$\mathbf{K}_0 \mathbf{C}_t \mathbf{K}_0^\top \preceq \frac{1}{\mu^2} \mathbf{K}_0 (\mathbf{K}_0^\top \mathbf{K}_0)^\dagger \mathbf{K}_0^\top = \frac{1}{\mu^2} \mathbf{P}_{\mathbf{K}_0}, \quad (48)$$

where $\mathbf{P}_{\mathbf{K}_0}$ is the orthogonal projector onto $\text{Row}(\mathbf{K}_0)$. Therefore $\|\mathbf{K}_0 \mathbf{C}_t \mathbf{K}_0^\top\|_2 \leq 1/\mu^2$.

Combining equation 43–equation 48 yields

$$\|\mathbf{K}_0 \mathbf{C}_t \mathbf{K}_t^\top \mathbf{R}_t\|_F^2 \leq \frac{1}{\mu^2} \|\mathbf{R}_t\|_F^2, \quad (49)$$

which implies equation 38.

Finally, for equation 40, note that

$$\begin{aligned} \mathbf{S}_t &= \lambda^2 \mathbf{I} + \mu^2 \mathbf{K}_0^\top \mathbf{K}_0 + \sum_{i=1}^t \mathbf{K}_i^\top \mathbf{K}_i \\ &\succeq \left(\lambda^2 + \mu^2 \Sigma_{\min}^2(\mathbf{K}_0) + \sum_{i=1}^t \Sigma_{\min}^2(\mathbf{K}_i) \right) \mathbf{I}, \end{aligned} \quad (50)$$

hence $\|\mathbf{C}_t\|_2 = 1/\lambda_{\min}(\mathbf{S}_t)$ implies equation 40. \square

Proof of Theorem 4.1(ii) and the adaptive spectral variant.

Telescoping gives

$$\mathbf{W}_T^* - \mathbf{W}_0 = \sum_{t=1}^T (\mathbf{W}_t^* - \mathbf{W}_{t-1}^*). \quad (51)$$

Left-multiply by \mathbf{K}_0 and apply the triangle inequality:

$$\|\mathbf{K}_0(\mathbf{W}_T^* - \mathbf{W}_0)\|_F \leq \sum_{t=1}^T \|\mathbf{K}_0(\mathbf{W}_t^* - \mathbf{W}_{t-1}^*)\|_F. \quad (52)$$

Applying Lemma A.1 with equation 38 termwise yields

$$\|\mathbf{K}_0(\mathbf{W}_T^* - \mathbf{W}_0)\|_F \leq \frac{1}{\mu} \sum_{t=1}^T \|\mathbf{R}_t\|_F, \quad (53)$$

which proves Theorem 4.1(ii).

For the adaptive spectral variant, apply instead equation 39 and equation 40:

$$\begin{aligned} &\|\mathbf{K}_0(\mathbf{W}_t^* - \mathbf{W}_{t-1}^*)\|_F \\ &\leq \|\mathbf{K}_0\|_2 \|\mathbf{K}_t\|_2 \|\mathbf{C}_t\|_2 \|\mathbf{R}_t\|_F \end{aligned} \quad (54)$$

$$\leq \frac{\|\mathbf{K}_0\|_2 \|\mathbf{K}_t\|_2}{\lambda^2 + \mu^2 \Sigma_{\min}^2(\mathbf{K}_0) + \sum_{i=1}^t \Sigma_{\min}^2(\mathbf{K}_i)} \|\mathbf{R}_t\|_F. \quad (55)$$

Summing equation 55 over $t = 1, \dots, T$ gives the stated inequality. \square

B: Proofs for Proposition 4.2

Step 1: Expand the normalized objective. Let $\tilde{J}_t(W)$ denote the normalized objective

$$\tilde{J}_t(W) = \frac{1}{t} \sum_{i=1}^t \|K_i W - V_i\|_F^2 + \alpha_t \|W - W_0\|_F^2 + \beta_t \|K_0 W - V_0\|_F^2.$$

Expand the data-fit term using $\|K_i W - V_i\|_F^2 = \text{tr}(W^\top K_i^\top K_i W) - 2 \text{tr}(W^\top K_i^\top V_i) + \|V_i\|_F^2$ to obtain

$$\begin{aligned} \tilde{J}_t(W) &= \text{tr}(W^\top \hat{\Sigma}_{K,t} W) - 2 \text{tr}(W^\top \hat{\Sigma}_{KV,t}) + c_t \\ &\quad + \alpha_t \|W - W_0\|_F^2 + \beta_t \|K_0 W - V_0\|_F^2. \end{aligned} \quad (56)$$

Step 2: Proof of (i) (pointwise convergence). Fix any W . By the assumed moment convergence Equation (17),

$$\hat{\Sigma}_{K,t} \rightarrow \Sigma_K, \quad \hat{\Sigma}_{KV,t} \rightarrow \Sigma_{KV}.$$

Moreover, by the bounded fourth-moment assumption $\sup_i \mathbb{E}\|V_i\|_F^4 < \infty$, we have $\sup_i \mathbb{E}\|V_i\|_F^2 < \infty$, so $\{c_t\}$ is tight and (along the same probability-1 event used for the empirical-moment convergence) converges to the constant $\mathbb{E}\|V\|_F^2$. Finally, $\alpha_t \rightarrow \alpha$ and $\beta_t \rightarrow \beta$ by assumption. Taking limits in Equation (56) yields, for each fixed W ,

$$\begin{aligned} \tilde{J}_t(W) &\longrightarrow \text{tr}(W^\top \Sigma_K W) - 2 \text{tr}(W^\top \Sigma_{KV}) + \mathbb{E}\|V\|_F^2 \\ &\quad + \alpha \|W - W_0\|_F^2 + \beta \|K_0 W - V_0\|_F^2. \end{aligned} \quad (57)$$

The right-hand side equals $\mathcal{R}(W) + \alpha \|W - W_0\|_F^2 + \beta \|K_0 W - V_0\|_F^2$, i.e., $\mathcal{R}_{\text{ridge}}(W)$ up to an additive constant. This proves (i).

Step 3: Proof of (ii) (strict convexity and uniqueness).

$$\begin{aligned} \mathcal{R}_{\text{ridge}}(W) &= \text{tr}(W^\top \Sigma_K W) - 2 \text{tr}(W^\top \Sigma_{KV}) \\ &\quad + \alpha \|W - W_0\|_F^2 + \beta \|K_0 W - V_0\|_F^2 + \text{const} \end{aligned} \quad (58)$$

Its Hessian (with respect to W) is the linear operator

$$\nabla^2 \mathcal{R}_{\text{ridge}}(W) = 2 \left(\Sigma_K + \alpha I + \beta K_0^\top K_0 \right), \quad (59)$$

acting identically on each of the d_V columns. Under the assumption in Equation (17) that $\Sigma_K \succ 0$ (on the relevant subspace), and since $\alpha, \beta \geq 0$, the matrix $\Sigma_K + \alpha I + \beta K_0^\top K_0$ is positive definite on that subspace. Hence $\mathcal{R}_{\text{ridge}}$ is strictly convex and admits a unique minimizer W^\dagger . This proves (ii).

Step 4: Proof of (iii) (consistency via closed form). Because \tilde{J}_t is quadratic, its minimizer W_t^* has the closed form

$$W_t^* = \left(\hat{\Sigma}_{K,t} + \alpha_t I + \beta_t K_0^\top K_0 \right)^{-1} \left(\hat{\Sigma}_{KV,t} + \alpha_t W_0 + \beta_t K_0^\top V_0 \right). \quad (60)$$

Similarly, the unique minimizer W^\dagger of $\mathcal{R}_{\text{ridge}}$ satisfies

$$W^\dagger = \left(\Sigma_K + \alpha I + \beta K_0^\top K_0 \right)^{-1} \left(\Sigma_{KV} + \alpha W_0 + \beta K_0^\top V_0 \right). \quad (61)$$

By Equation (17) and $\alpha_t \rightarrow \alpha$, $\beta_t \rightarrow \beta$, the matrices and right-hand sides in Equation (60) converge:

$$\hat{\Sigma}_{K,t} + \alpha_t I + \beta_t K_0^\top K_0 \longrightarrow \Sigma_K + \alpha I + \beta K_0^\top K_0,$$

$$\hat{\Sigma}_{KV,t} + \alpha_t W_0 + \beta_t K_0^\top V_0 \longrightarrow \Sigma_{KV} + \alpha W_0 + \beta K_0^\top V_0.$$

By (ii), the limit matrix $\Sigma_K + \alpha I + \beta K_0^\top K_0$ is invertible (on the relevant subspace), and matrix inversion is continuous on the set of invertible matrices. Therefore, taking limits in Equation (60) yields

$$W_t^* \longrightarrow W^\dagger,$$

along the same probability-1 event, which establishes almost sure convergence. This proves (iii) and completes the proof.

C. Useful limit regimes (hard constraints as limits)

Corollary A.2 (Hard limits from soft penalties). *Fix T and $\{\{K_i, V_i\}\}_{i=0}^T$, and assume the anchor condition equation 27. Let W_T^* minimize equation 3 at time T and define*

$$D_T := \|K_0(W_T^* - W_0)\|_F, \quad P_T := \|W_T^* - W_0\|_F. \quad (62)$$

Then:

(i) (Hard anchor as $\mu \rightarrow \infty$.) For any fixed $\lambda \geq 0$,

$$D_T \leq \frac{1}{\mu} \left(\sum_{i=1}^T \|K_i W_0 - V_i\|_F^2 \right)^{1/2} \quad (63)$$

hence as $\mu \rightarrow \infty$, $D_T \rightarrow 0$.

(ii) (Freezing as $\lambda \rightarrow \infty$.) For any fixed $\mu \geq 0$,

$$P_T \leq \frac{1}{\lambda} \left(\sum_{i=1}^T \|K_i W_0 - V_i\|_F^2 \right)^{1/2}, \quad (64)$$

hence as $\lambda \rightarrow \infty$, $P_T \rightarrow 0$, and consequently $D_T \rightarrow 0$ as well.

Proof. Let $\Phi_T(\mathbf{W})$ denote the objective equation 3 at time T . Since W_T^* is the minimizer, $\Phi_T(W_T^*) \leq \Phi_T(W_0)$. Using $K_0 W_0 = V_0$, we have

$$\Phi_T(W_0) = \sum_{i=1}^T \|K_i W_0 - V_i\|_F^2. \quad (65)$$

(i) $\mu \rightarrow \infty$. From $\Phi_T(W_T^*) \leq \Phi_T(W_0)$,

$$\mu^2 \|K_0 W_T^* - V_0\|_F^2 \leq \Phi_T(W_T^*) \leq \Phi_T(W_0). \quad (66)$$

Since $D_T = \|K_0(W_T^* - W_0)\|_F = \|K_0 W_T^* - V_0\|_F$, combining with equation 65 yields equation 63.

(ii) $\lambda \rightarrow \infty$. Similarly,

$$\lambda^2 \|W_T^* - W_0\|_F^2 \leq \Phi_T(W_T^*) \leq \Phi_T(W_0), \quad (67)$$

and equation 65 implies equation 64. Then $D_T \leq \|K_0\|_2 P_T \rightarrow 0$. \square

B Detailed Hyperparameter Settings

For sequential editing experiments, we perform 10K edits for all methods. For methods that support batch editing (MEMIT, AlphaEdit, and RLSEdit), we use a batch size of 100. We edit layers {4,5,6,7,8} for Llama3-8B and layers {7,8,9,10,11} for Qwen2.5-7B for these methods. For ROME, we edit a single layer, using layer 5 for Llama3-8B and layer 11 for Qwen2.5-7B. For RLSEdit regularization, on Llama3-8B, we set $\lambda = 3$ and $\mu = 20000$ and on Qwen2.5-7B, we set $\lambda = 0$ and $\mu = 12000$.

C General Capability Benchmarks

Here we list the benchmarks used in general capability tests (5 GLUE experiments, MMLU, GSM8K, HumanEval, and MBPP).

GLUE Tasks involve:

- SST (STANFORD SENTIMENT TREEBANK) [Socher et al., 2013]: A sentence-level sentiment classification task that predicts the sentiment polarity of a given sentence.
- MRPC (MICROSOFT RESEARCH PARAPHRASE CORPUS) [Dolan and Brockett, 2005]: A sentence-pair task that determines whether two sentences are paraphrases of each other.
- CoLA (CORPUS OF LINGUISTIC ACCEPTABILITY) [Warstadt et al., 2019]: A grammatical acceptability task that predicts whether a sentence is linguistically acceptable.
- RTE (RECOGNIZING TEXTUAL ENTAILMENT) [Bentivogli et al., 2009]: A natural language inference (NLI) task in a binary setting. Given a premise and a hypothesis, the model predicts whether the premise entails the hypothesis.
- NLI (NATURAL LANGUAGE INFERENCE; COMMONLY MNLI-STYLE) [Williams et al., 2018]: A sentence-pair inference task that predicts the semantic relation between a premise and a hypothesis.

MMLU (MASSIVE MULTI-TASK LANGUAGE UNDERSTANDING) [Hendrycks et al., 2020]: A task that measures broad factual knowledge and reasoning.

GSM8K (GRADE SCHOOL MATH 8K) [Cobbe et al., 2021]: A math word-problem dataset that evaluates step-by-step arithmetic reasoning.

HUMAN EVAL [Chen et al., 2021]: A code generation benchmark where models synthesize Python functions from natural-language problem descriptions and are evaluated by unit tests.

MBPP (MOSTLY BASIC PROGRAMMING PROBLEMS) [Austin et al., 2021]: A programming benchmark consisting of short problem statements and test cases.

D Case Study

We present a representative example using task 0 from the HumanEval dataset to highlight how long edit streams can degrade reasoning and code-generation quality for ALPHAEDIT and MEMIT, while **RLSEdit** preserves this capability.

HumanEval Task 0 Prompt (has_close_elements)

Task description: Given a list of real numbers and a threshold, determine whether there exist two *distinct* elements whose absolute difference is *strictly less* than the threshold.

```
from typing import List

def has_close_elements(numbers: List[float], threshold: float) -> bool:
    """
    Check if in given list of numbers, are any two numbers closer to each other than
    given threshold.
    """
    >>> has_close_elements([1.0, 2.0, 3.0], 0.5)
    False
    >>> has_close_elements([1.0, 2.8, 3.0, 4.0, 5.0, 2.0], 0.3)
    True
    """
```

Pre-edit model (Correct)

OK: Baseline output is correct (distinct pairs and strict inequality $<$).

```
from typing import List

def has_close_elements(numbers: List[float], threshold: float) ->
    bool:
    for i in range(len(numbers)):
        for j in range(i + 1, len(numbers)):
            if abs(numbers[i] - numbers[j]) < threshold:
                return True
    return False
```

AlphaEdit @ 2k edits (Correct)

OK: Uses distinct pairs ($j = i + 1$) and strict inequality ($<$).

```
from typing import List

def has_close_elements(numbers: List[float], threshold: float) ->
    bool:
    for i in range(len(numbers)):
        for j in range(i + 1, len(numbers)):
            if abs(numbers[i] - numbers[j]) < threshold:
                return True
    return False
```

AlphaEdit @ 4k edits (Boundary error)

Error: Uses \leq instead of $<$. Counterexample: $[0.0, 0.5]$, threshold=0.5.

```
from typing import List

def has_close_elements(numbers: List[float], threshold: float) -> bool:
    for i in range(len(numbers)):
        for j in range(i + 1, len(numbers)):
            if abs(numbers[i] - numbers[j]) <= threshold:
                return True
    return False
```

AlphaEdit @ 6k edits (Boundary error)

Error: Same boundary error as 4k.

```
from typing import List

def has_close_elements(numbers: List[float], threshold: float) ->
    bool:
    for i in range(len(numbers)):
        for j in range(i + 1, len(numbers)):
            if abs(numbers[i] - numbers[j]) <= threshold:
                return True
    return False
```

AlphaEdit @ 8k edits (Semantic bug)

Error: Loops allow $j = i$, so $\text{abs}(x-x)=0$ and it returns True spuriously for any threshold > 0 .

```
def has_close_elements(numbers,
                      threshold):
    for i in range(len(numbers)):
        for j in range(len(numbers)):
            if abs(numbers[i] - numbers[j]) < threshold:
                return True
    return False
```

AlphaEdit @ 10k edits (Garbled / empty output)

Error: Non-executable output (near-empty / whitespace / escape sequences).

Figure 5: Case study on HumanEval task 0 (AlphaEdit). The prompt above is the original statement of HumanEval/0. ALPHAEDIT remains correct at 2k edits but begins to fail from 4k edits onward (boundary error), later exhibiting a semantic bug at 8k and degenerating into near-empty/garbled output at 10k.

Pre-edit model (Correct)	RLSEdit @ 2k edits (Correct)	RLSEdit @ 4k edits (Correct)
<pre>from typing import List def has_close_elements(numbers: List[float], threshold: float) -> bool: for i in range(len(numbers)): for j in range(i + 1, len(numbers)): if abs(numbers[i] - numbers[j]) < threshold: return True return False</pre>	<pre>from typing import List def has_close_elements(numbers: List[float], threshold: float) -> bool: for i in range(len(numbers)): for j in range(i + 1, len(numbers)): if abs(numbers[i] - numbers[j]) < threshold: return True return False</pre>	<pre>from typing import List def has_close_elements(numbers: List[float], threshold: float) -> bool: for i in range(len(numbers)): for j in range(i + 1, len(numbers)): if abs(numbers[i] - numbers[j]) < threshold: return True return False</pre>
RLSEdit @ 6k edits (Correct)	RLSEdit @ 8k edits (Correct)	RLSEdit @ 10k edits (Correct)
<pre>from typing import List def has_close_elements(numbers: List[float], threshold: float) -> bool: for i in range(len(numbers)): for j in range(i + 1, len(numbers)): if abs(numbers[i] - numbers[j]) < threshold: return True return False</pre>	<pre>from typing import List def has_close_elements(numbers: List[float], threshold: float) -> bool: for i in range(len(numbers)): for j in range(i + 1, len(numbers)): if abs(numbers[i] - numbers[j]) < threshold: return True return False</pre>	<pre>from typing import List def has_close_elements(numbers: List[float], threshold: float) -> bool: for i in range(len(numbers)): for j in range(i + 1, len(numbers)): if abs(numbers[i] - numbers[j]) < threshold: return True return False</pre>

Figure 6: Case study on HumanEval task 0 (RLSEdit). In contrast to ALPHAEDIT, **RLSEdit** preserves a correct implementation across all checkpoints (2k–10k).

Pre-edit model (Correct)	MEMIT @ 2k edits (Garbled)	MEMIT @ 4k edits (Garbled)
<pre>from typing import List def has_close_elements(numbers: List[float], threshold: float) -> bool: for i in range(len(numbers)): for j in range(i + 1, len(numbers)): if abs(numbers[i] - numbers[j]) < threshold: return True return False</pre>	<pre>Barrett\ufffd\ufffd\ufffd S Japan; Italian\ufffd:// Shea://\ufffd\ufffd Japan cath Italy\ufffd R Shea:// Shea Shea:// Japan Ne Shea Shea Ne Barcelona\ufffd Ne Belgium\ufffd Japan (://\ufffd\ufffd road Shea Shea:// B Shea Shea Ne://:\ufffd Japan (Belgium\ufffd Belgium cath cath Belgium\ufffd Belgium Belgium:// Belgium Ne Belgian cath Tokyo Tokyo (Belgium\ufffd Belgium Belgium://e B\ufffd://\ufffdanced\ufffd Ballet\ufffd Italy Ne Belgian Hub Shea ...</pre>	<pre>hemhemhemhem jazzhemhem jazzhem jazzhem jazzhem jazzhem jazzhem jazz jazzhem jazz jazzhem jazz jazzhem jazz ...</pre>
MEMIT @ 6k edits (Garbled)	MEMIT @ 8k edits (Empty)	MEMIT @ 10k edits (Empty)
<pre>SGlobal onSGlobal VictoriaongSGlobal Victoria onenn348.usermodel.usermodelhem onenn onenn like\ufffdSGlobaltee Victoria VictoriaSGlobal Victoria onSGlobal Victoria VictoriaSGlobal Victoria VictoriaSGlobal Victoria onSGlobal onSGlobal onSGlobal onSGlobal onSGlobal onSGlobal onSGlobal onSGlobalinesSGlobal onSGlobal onSGlobal onSGlobalines348 Robbieines onSGlobal Victoria Victoria VictoriaSGlobal Victoria onSGlobalinesines348 Rob ...</pre>	No Output	No Output

Figure 7: Case study on HumanEval task 0 (MEMIT). Under long edit streams, MEMIT quickly degenerates into non-executable, garbled or empty text outputs across checkpoints, unlike **RLSEdit** which preserves a valid implementation.