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 Lu[4], Yufei Cui[1] and **Xiao-Wen Chang[1]**
[1]McGill University
[2]Mila–Quebec AI Institute
[3]SimpleWay.AI
[4]Université 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 retraining while preserving unrelated behavior. In real deployment, edits arrive as long streams, creating a *plasticity–stability dilemma*: repeated locate-then-edit "hard writes" can accumulate interference over time, while rigid preservation constraints may protect only explicitly constrained directions, allowing past edits or unconstrained behaviors to deviate.

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 together with two regularizers that control deviation from the pre-trained weights and from a designated anchor mapping. This objective admits an efficient Woodbury-based online recursion, 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 CounterFact and ZsRE across multiple model families show stable scaling to 10K edits, outperforming strong baselines in both edit success and holistic stability, while retaining early edits and preserving general capabilities on GLUE and held-out reasoning/code benchmarks.

Code will be at here.

1 Introduction

Despite the large amount of knowledge they store within their parameters, large language models (LLMs) [Yang and others, 2024; OpenAI, 2023; DeepSeek-AI and others, 2025] inevitably contain outdated, incomplete, or incorrect knowledge [De Cao *et al.*, 2021; Mitchell *et al.*, 2022] when statically deployed without re-training or access to external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 high computational cost of retraining from scratch, many applications require updating models through *edits* to a subset of parameters, with the goal of integrating 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, and the editor must remain reliable after each edit [Hartvigsen *et al.*, 2023; Gupta *et al.*, 2024].

The many-edits regime presents a dilemma. To remain useful over long streams, an editor must both memorize incoming information and preserve knowledge acquired from earlier edits. In practice, failures often appear as two coupled forms of forgetting:

1. *Retroactive Edit Forgetting*: future edits can overwrite edits applied in the past.
2. *General-Ability Degradation*: edits can deteriorate out-of-scope reasoning and language understanding.

Existing sequential editors typically fail for complementary reasons. *Locate-then-edit* approaches [Meng *et al.*, 2022; Meng *et al.*, 2023] perform *hard writes*, forcing the model to learn new associations with limited control over previously stored knowledge. As the number of edits accumulates, these updates may interfere with one another, leading to instability and retroactive forgetting of earlier facts. Conversely, *null-space* editors [Fang *et al.*, 2025; Sun *et al.*, 2025; Lyu *et al.*, 2025] project updates into a feasible subspace conditioned on an anchor mapping, thereby preserving the associations explicitly constrained by that mapping. However, as edits are repeatedly applied, maintaining complete preservation would require a growing set of constraints, while loosening these constraints may still allow degradation in general abilities or retention of previous facts. As a consequence, neither paradigm is sufficient for long sequential editing.

We focus on parameter-editing methods. This distinction is mechanism-based rather than dataset-based. [Wang *et al.*, 2025] Alternatively, sequential parameter editing can be viewed as *online regularized least squares* on a layer-wise key-value surrogate [Sayed, 2003]. Each edit contributes a quadratic fitting term, while preservation is controlled by two explicit deviation terms: one penalizing deviation from the initial model, and another penalizing deviation from a designated *anchor mapping*. This gives a soft-constraint formulation in which learning new edits and preserving previous knowledge are optimized within a single objective.

From this view, we propose **RLSEdit**, a recursive least-squares editor for long sequential editing. RLSEdit formu-

lates editing as a quadratic objective and derives an efficient online recursion via the Woodbury identity [Sherman and Morrison, 1950; Woodbury, 1950; Hager, 1989]. Its per-edit cost is independent of the number of previous edits and instead scales with the current edit rank/size. We further provide theoretical deviation bounds and an asymptotic characterization of the edit-preservation trade-off under long edit streams.

To summarize, our main contributions are:

- We formulate lifelong editing as online regularized least squares with explicit parameter-deviation and anchor-deviation controls, providing a soft-constraint alternative between hard writing and hard preservation.
- We derive a Woodbury-based online update whose per-edit cost is independent of the number of past edits.
- We provide deviation bounds and an asymptotic characterization of the adherence-preservation trade-off under long edit streams.
- We evaluate RLSEdit on `Llama-3` [Dubey and others, 2024] and `Qwen2.5` [Yang and others, 2024] after 10K edits, showing 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 explain how different editing methods balance between making new edits and preserving previous edits and knowledge, and why long-sequential editing is 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 $(\boldsymbol{k}, \boldsymbol{v})$ at selected token positions, where $\boldsymbol{k} \in \mathbb{R}^{d_k}$ is the input activation to the edited map and $\boldsymbol{v} \in \mathbb{R}^{d_v}$ is the corresponding output activation. For the t-th edit, we stack u_t such pairs to form $\boldsymbol{K}_t \in \mathbb{R}^{u_t \times d_k}$ and $\boldsymbol{V}_t \in \mathbb{R}^{u_t \times d_v}$.

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:

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

Here $(\boldsymbol{K}, \boldsymbol{V})$ denotes the current key-value associations contained within the model parameters, and $(\boldsymbol{K}^\star, \boldsymbol{V}^\star)$ 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

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

This leads to a soft fit to the new edits under a hard preservation constraint. In ALPHAEDIT [Fang *et al.*, 2025], $\boldsymbol{K}_{\mathrm{pres},t}$ 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 $\boldsymbol{K}_{\mathrm{pres},t}$ and the induced projector evolve 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 real-world LLM use cases. Models often exist in environments where they are deployed for long periods of time and where the number of required edits can grow continuously. In such longer edit streams, existing methods can suffer for several reasons: hard satisfaction of each batch can induce growing interference between 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.

Relation to Continual Learning. RLSEdit is also related to continual learning methods such as EWC [Kirkpatrick *et al.*, 2017] and online Laplace [Ritter *et al.*, 2018], which use quadratic penalties to restrict parameter drift from previously learned solutions. In our objective, the term $\lambda^2\|\boldsymbol{W} - \boldsymbol{W}_0\|_F^2$ plays a similar regularizing role. However, our setting is different: RLSEdit targets long sequential LLM editing through layer-wise key-value constraints, combines the drift penalty with an anchor-mapping term, and admits an exact Woodbury-based recursive update whose per-edit cost is independent of the edit history length.

Beyond Direct Parameter Writes. Several recent lines differ from direct streaming parameter updates mainly in editing mechanism. 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 can be evaluated on related editing tasks, but they intervene through additional components such as memory, retrieval, or prompting. They are therefore complementary to our focus on an efficient, single-objective streaming parameter editor. Model editing is also related to continual learning, especially quadratic-regularization methods such as EWC and online Laplace approximations [Kirkpatrick *et al.*, 2017; Ritter *et al.*, 2018]. However, RLSEdit is designed for post-training LLM editing and couples cumulative key–value least squares, an anchor-mapping penalty, and a Woodbury recursion with history-independent per-edit cost.

3 Methodology

We present our editing framework in three parts. We first introduce a recursive least-squares (RLS) formulation that accumulates all editing residuals in a single quadratic objective, with penalties on both deviation from the initial model parameters and deviation from an anchor mapping in Section 3.1. We then analyze the computational complexity of the resulting updates and compare them with existing sequential editors under long edit streams in Section 3.2. Finally, we contrast *soft* and *hard* constraint designs, showing how existing editors can be viewed as limiting cases of our formulation in Section 3.3.

3.1 A recursive least squares editor

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

Regularized Least-Squares Equation

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

$$\begin{aligned}\boldsymbol{W}_t^* \coloneqq \arg\min_{\boldsymbol{W}} \sum_{i=1}^{t} \|\boldsymbol{K}_i\boldsymbol{W} - \boldsymbol{V}_i\|_F^2 \\ +\lambda^2 \|\boldsymbol{W} - \boldsymbol{W}_0\|_F^2 \\ +\mu^2 \|\boldsymbol{K}_0\boldsymbol{W} - \boldsymbol{V}_0\|_F^2,\end{aligned} \tag{3}$$

where λ and μ are hyperparameters. The first term fits all edit pairs observed so far. The second term penalizes parameter drift from the initial weight $\boldsymbol{W}_0$, and the third term encourages the edited weight to preserve the anchor mapping $\boldsymbol{K}_0\boldsymbol{W} \approx \boldsymbol{V}_0$.

To rewrite Equation 3 as a standard matrix least-squares problem, define

$$\boldsymbol{A}_t = \begin{bmatrix} \lambda \boldsymbol{I}_{d_k} \\ \mu \boldsymbol{K}_0 \\ \boldsymbol{K}_1 \\ \vdots \\ \boldsymbol{K}_t \end{bmatrix}, \quad \boldsymbol{B}_t = \begin{bmatrix} \lambda \boldsymbol{W}_0 \\ \mu \boldsymbol{V}_0 \\ \boldsymbol{V}_1 \\ \vdots \\ \boldsymbol{V}_t \end{bmatrix}. \tag{4}$$

Then Equation 3 becomes

$$\boldsymbol{W}_t^* = \arg\min_{\boldsymbol{W}} \|\boldsymbol{A}_t\boldsymbol{W} - \boldsymbol{B}_t\|_F^2. \tag{5}$$

To make the closed-form solution explicit, consider one output column at a time. For a column w of $\boldsymbol{W}$ and the corresponding column b_t of $\boldsymbol{B}_t$, Equation 5 reduces to

$$\min_w \|\boldsymbol{A}_t w - b_t\|_2^2.$$

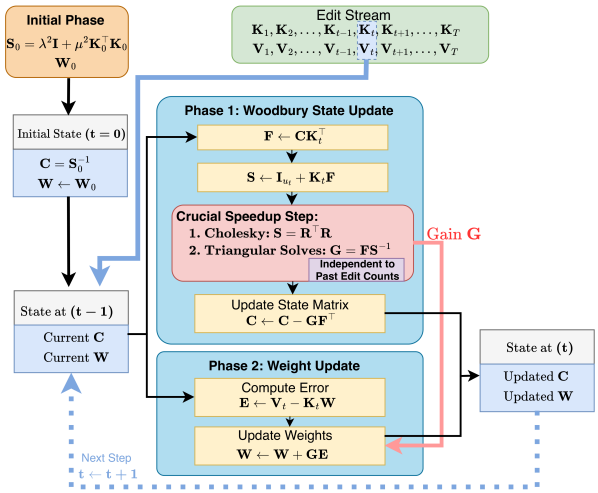

Figure 1: The recursive workflow of our RLS-Woodbury editor. The process alternates between updating the covariance state via the Woodbury identity and updating the weights. The highlighted block shows how the update avoids a full $O(d_k^3)$ factorization during the edit stream by solving only small $u_t \times u_t$ systems.

Taking the derivative and setting it to zero gives the normal equation

$$\boldsymbol{A}_t^\top \boldsymbol{A}_t w_t^* = \boldsymbol{A}_t^\top b_t.$$

Stacking the solutions for all output columns gives the matrix normal equation

$$\boldsymbol{A}_t^\top \boldsymbol{A}_t \boldsymbol{W}_t^* = \boldsymbol{A}_t^\top \boldsymbol{B}_t.$$

Let

$$\boldsymbol{S}_t \coloneqq \boldsymbol{A}_t^\top \boldsymbol{A}_t = \lambda^2 \boldsymbol{I} + \mu^2 \boldsymbol{K}_0^\top \boldsymbol{K}_0 + \sum_{i=1}^{t} \boldsymbol{K}_i^\top \boldsymbol{K}_i, \tag{6}$$

$$\boldsymbol{T}_t \coloneqq \boldsymbol{A}_t^\top \boldsymbol{B}_t = \lambda^2 \boldsymbol{W}_0 + \mu^2 \boldsymbol{K}_0^\top \boldsymbol{V}_0 + \sum_{i=1}^{t} \boldsymbol{K}_i^\top \boldsymbol{V}_i. \tag{7}$$

When $\lambda > 0$, $\boldsymbol{S}_t \succ 0$, so the minimizer is unique and is given by

$$\boldsymbol{W}_t^* = \boldsymbol{S}_t^{-1} \boldsymbol{T}_t. \tag{8}$$

This solution jointly fits all edit pairs up to time t, while the two regularizers keep the edited weights close to $\boldsymbol{W}_0$ and preserve the anchor mapping $\boldsymbol{K}_0\boldsymbol{W} \approx \boldsymbol{V}_0$.

Efficient Recursion via Normal Equations

Direct computation of Equation 8 requires inverting $\boldsymbol{S}_t$, which is expensive when repeated over a long edit stream. We therefore derive an efficient recursive update. From Equation 8, the minimizer $\boldsymbol{W}_t^*$ satisfies

$$\left(\boldsymbol{A}_t^\top \boldsymbol{A}_t\right) \boldsymbol{W}_t^* = \boldsymbol{A}_t^\top \boldsymbol{B}_t. \tag{9}$$

Let $\boldsymbol{C}_t \coloneqq \boldsymbol{S}_t^{-1}$. Using Equation 6, we have

$$\boldsymbol{C}_t^{-1} = \boldsymbol{C}_{t-1}^{-1} + \boldsymbol{K}_t^\top \boldsymbol{K}_t. \tag{10}$$

Algorithm 1 RLS-Woodbury Editing

Require: Initial weight $\boldsymbol{W}_0$; anchor pair $(\boldsymbol{K}_0, \boldsymbol{V}_0)$; penalties (λ, μ); edit stream $\{(\boldsymbol{K}_t, \boldsymbol{V}_t)\}_{t=1}^{T}$.
Ensure: Edited weight $\boldsymbol{W}_T^*$.

$\boldsymbol{S}_0 \leftarrow \lambda^2 \boldsymbol{I}_{d_k} + \mu^2 \boldsymbol{K}_0^\top \boldsymbol{K}_0$
$\boldsymbol{T}_0 \leftarrow \lambda^2 \boldsymbol{W}_0 + \mu^2 \boldsymbol{K}_0^\top \boldsymbol{V}_0$
$\boldsymbol{S}_0 = \boldsymbol{R}_0^\top \boldsymbol{R}_0$ ▷ Cholesky factorization
$\boldsymbol{C}_0 \leftarrow \boldsymbol{S}_0^{-1}$ ▷ Computed via triangular solves
$\boldsymbol{W}_0^* \leftarrow \boldsymbol{C}_0 \boldsymbol{T}_0$
for $t = 1, 2, \ldots, T$ **do**
  Covariance update
  $\boldsymbol{F}_t \leftarrow \boldsymbol{C}_{t-1} \boldsymbol{K}_t^\top$
  $\boldsymbol{H}_t \leftarrow \boldsymbol{I}_{u_t} + \boldsymbol{K}_t \boldsymbol{F}_t$
  $\boldsymbol{H}_t = \boldsymbol{R}_t^\top \boldsymbol{R}_t$ ▷ Cholesky factorization
  $\boldsymbol{Y}_t \leftarrow \boldsymbol{F}_t \boldsymbol{R}_t^{-1}$ ▷ Triangular solve
  $\boldsymbol{C}_t \leftarrow \boldsymbol{C}_{t-1} - \boldsymbol{Y}_t \boldsymbol{Y}_t^\top$
  Weight update
  $\boldsymbol{E}_t \leftarrow \boldsymbol{V}_t - \boldsymbol{K}_t \boldsymbol{W}_{t-1}^*$
  $\boldsymbol{G}_t \leftarrow \boldsymbol{F}_t \boldsymbol{H}_t^{-1}$ ▷ Gain matrix
  $\boldsymbol{W}_t^* \leftarrow \boldsymbol{W}_{t-1}^* + \boldsymbol{G}_t \boldsymbol{E}_t$
end for
return $\boldsymbol{W}_T^*$

Next, define

$$\boldsymbol{F}_t \coloneqq \boldsymbol{C}_{t-1} \boldsymbol{K}_t^\top \in \mathbb{R}^{d_k \times u_t}, \qquad \boldsymbol{H}_t \coloneqq \boldsymbol{I}_{u_t} + \boldsymbol{K}_t \boldsymbol{F}_t.$$

By the Sherman–Morrison–Woodbury identity,

$$\boldsymbol{C}_t = \boldsymbol{C}_{t-1} - \boldsymbol{F}_t \boldsymbol{H}_t^{-1} \boldsymbol{F}_t^\top. \tag{11}$$

In implementation, we factorize $\boldsymbol{H}_t = \boldsymbol{R}_t^\top \boldsymbol{R}_t$ by Cholesky and use triangular solves, avoiding explicit matrix inversion.

From Equation 7, we also have

$$\boldsymbol{T}_t = \boldsymbol{T}_{t-1} + \boldsymbol{K}_t^\top \boldsymbol{V}_t.$$

Combining this with the covariance recursion gives the weight update

$$\boldsymbol{W}_t^* = \boldsymbol{W}_{t-1}^* + \boldsymbol{C}_t \boldsymbol{K}_t^\top \left(\boldsymbol{V}_t - \boldsymbol{K}_t \boldsymbol{W}_{t-1}^* \right). \tag{12}$$

Equivalently, since $\boldsymbol{C}_t \boldsymbol{K}_t^\top = \boldsymbol{F}_t \boldsymbol{H}_t^{-1}$, the update only requires solving the small $u_t \times u_t$ system in $\boldsymbol{H}_t$. Letting $\boldsymbol{E}_t \coloneqq \boldsymbol{V}_t - \boldsymbol{K}_t \boldsymbol{W}_{t-1}^*$, each edit requires only updating $\boldsymbol{C}_t$ via Equation 11 and then updating $\boldsymbol{W}_t^*$ via Equation 12.

3.2 Complexity analysis

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

RLS-Woodbury Updates

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

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

The covariance-state update is dominated by forming these products, solving the resulting $u_t \times u_t$ system, and applying the low-rank correction:

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

For the weight update, we reuse the same $u_t \times u_t$ solve to apply the gain $\boldsymbol{G}_t = \boldsymbol{C}_t \boldsymbol{K}_t^\top = \boldsymbol{F}_t \boldsymbol{H}_t^{-1} \in \mathbb{R}^{d_k \times u_t}$ and update $\boldsymbol{W}_t$ using the residual $\boldsymbol{E}_t$. This step is dominated by the key-value multiplication against d_v outputs:

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

Overall, the per-edit runtime is therefore

$$\text{Per edit:} \qquad O\big(d_k^2 u_t + d_k d_v u_t + d_k u_t^2 + u_t^3\big),$$

which simplifies to $O\big(d_k^2 u_t + d_k d_v u_t\big)$ 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 weight update onto the null space of a fixed preserved-knowledge set. In sequential editing, it also regularizes against disrupting previously updated knowledge represented by accumulated key-value pairs. A direct sequential implementation therefore involves history-dependent dense matrices over the feature dimension and repeated projection or factorization steps. Let m_{t-1} denote the number of previously updated pairs and let u_t denote the number of pairs in the current edit. The dominant cost can include a dense $d_k \times d_k$ factorization, together with Gram matrix construction over the accumulated pairs:

$$\text{Per edit:} \qquad O(d_k^3) \;+\; O\big(d_k^2 (m_{t-1} + u_t)\big).$$

Even with more careful incremental implementations, the cost remains tied to the accumulated preservation set or the dimension of the remaining feasible subspace. In contrast, RLSEdit avoids an $O(d_k^3)$ factorization during the edit stream by maintaining $\boldsymbol{C}_t = (\boldsymbol{A}_t^\top \boldsymbol{A}_t)^{-1}$ and using a Woodbury recursion. Each edit only solves a $u_t \times u_t$ system, where typically $u_t \ll d_k$, making the update more efficient in long-edit settings, as shown in 3.

3.3 Hard versus Soft Constraints

RLSEdit can be viewed as a soft-constraint alternative to two common hard regimes. Locate-then-edit methods such as ROME enforce the new association as a hard write while softly penalizing background deviation. Null-space methods such as ALPHAEDIT instead enforce hard preservation by restricting updates to a feasible subspace. In contrast, RLSEdit keeps both edit adherence and preservation soft through the cumulative objective

$$\boldsymbol{W}_t^\star = \arg\min_{\boldsymbol{W}} \sum_{i=1}^{t} \|\boldsymbol{K}_i \boldsymbol{W} - \boldsymbol{V}_i\|_F^2 + \lambda^2 \|\boldsymbol{W} - \boldsymbol{W}_0\|_F^2 + \mu^2 \|\boldsymbol{K}_0 \boldsymbol{W} - \boldsymbol{V}_0\|_F^2.$$

As $\mu \to \infty$, the anchor mapping becomes a hard constraint. Similarly, assigning infinite weight to selected past fitting terms recovers hard preservation through the standard penalty method. Thus, RLSEdit interpolates between hard-write and hard-preserve behavior while remaining in a soft–soft regime for finite (λ, μ).

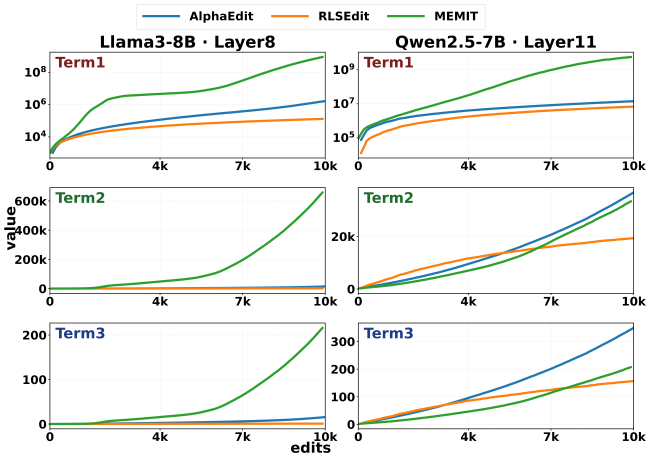

Figure 2: **Evolution of objective terms over** 10K **edits.** We compare RLSEDIT against baselines (ALPHAEDIT, MEMIT) on three metrics: **Term 1** ($\|\boldsymbol{K}_t\boldsymbol{W} - \boldsymbol{V}_t\|_F^2$) measures the fitting error for the current edit; **Term 2** ($\|\boldsymbol{W} - \boldsymbol{W}_0\|_F^2$)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

4 Theoretical Analysis

We provide deviation bounds in terms of (λ, μ): λ controls global parameter deviation from $\boldsymbol{W}_0$, and μ controls deviation of the anchor mapping $\boldsymbol{K}_0\boldsymbol{W}$.

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

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

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

(ii) (Anchor-map deviation) *If $\mu > 0$, then for any $T \geq 1$,*

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

Theorem 4.1 shows that λ and μ control different types of deviation. A larger λ keeps the solution closer to the original weights $\boldsymbol{W}_0$, while a larger μ better preserves the anchor mapping $\boldsymbol{K}_0\boldsymbol{W} \approx \boldsymbol{V}_0$. The residual $\boldsymbol{R}_t$ measures how well the current edit is satisfied before the t-th update. Thus, increasing λ or μ makes the update more conservative, but may also increase the edit residual when the new edit conflicts with the preservation constraints.

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

$$\boldsymbol{V}_i = \boldsymbol{K}_i\boldsymbol{W}^\star + \boldsymbol{E}_i, \qquad \sup_i \mathbb{E}\,\|\boldsymbol{E}_i\|_F^2 < \infty, \tag{13}$$

where $\boldsymbol{E}_i$ captures approximation error from other layers, context variability, and mismatch in the linearized output.

Sequential edits do not need to be i.i.d. We only assume long-run stability of the empirical second moments, namely that there exist matrices $\boldsymbol{\Sigma}_k$ and $\boldsymbol{\Sigma}_{kv}$ such that

$$\frac{1}{t}\sum_{i=1}^{t}\boldsymbol{K}_i^\top\boldsymbol{K}_i \to \boldsymbol{\Sigma}_k, \quad \frac{1}{t}\sum_{i=1}^{t}\boldsymbol{K}_i^\top\boldsymbol{V}_i \to \boldsymbol{\Sigma}_{kv}, \quad \text{s.t. } \boldsymbol{\Sigma}_k \succ 0 \text{ on the relevant subspace.} \tag{14}$$

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

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

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

$$\tilde{J}_t(\boldsymbol{W}) = \frac{1}{t}\sum_{i=1}^{t}\|\boldsymbol{K}_i\boldsymbol{W} - \boldsymbol{V}_i\|_F^2 + \alpha_t\|\boldsymbol{W} - \boldsymbol{W}_0\|_F^2 + \beta_t\|\boldsymbol{K}_0\boldsymbol{W} - \boldsymbol{V}_0\|_F^2. \tag{15}$$

Proposition 4.2 (Asymptotic behavior of the RLS editor)**.** *Assume Equation 13, and the moment convergence in Equation 14. Also assume $\sup_i \mathbb{E}\,\|\boldsymbol{K}_i\|_F^4 < \infty$ and $\sup_i \mathbb{E}\,\|\boldsymbol{V}_i\|_F^4 < \infty$. Let $\boldsymbol{W}_t^*$ be the minimizer of $J_t(\boldsymbol{W})$, with $\alpha_t = \lambda_t^2/t$ and $\beta_t = \mu_t^2/t$. Suppose that $\alpha_t \to \alpha$ and $\beta_t \to \beta$ for some $\alpha, \beta \in [0, \infty)$ as $t \to \infty$. Define the limiting quadratic risk $\mathcal{R}(\boldsymbol{W})$ through the limiting second moments in Equation 14. Then:*

(i) The normalized objectives $\tilde{J}_t$ converge pointwise to the regularized limiting risk

$$\mathcal{R}_{\text{ridge}}(\boldsymbol{W}) := \mathcal{R}(\boldsymbol{W}) + \alpha\|\boldsymbol{W} - \boldsymbol{W}_0\|_F^2 + \beta\|\boldsymbol{K}_0\boldsymbol{W} - \boldsymbol{V}_0\|_F^2. \tag{16}$$

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

(iii) The RLS editor converges to this minimizer:

$$\boldsymbol{W}_t^* \to \boldsymbol{W}^\dagger \qquad \text{as } t \to \infty. \tag{17}$$

Remark 4.3. Proposition 4.2 is an asymptotic characterization for long-run stable edit streams, not a universal guarantee for arbitrary non-stationary deployments. It does not assume that edits are i.i.d.; the key requirement is the convergence of the empirical second moments in Equation 14. If these limits do not exist, then the editor should not be interpreted as converging to a single limiting risk minimizer. In that case, the relevant guarantee is the finite-time deviation bound in Proposition 4.1.

If $\alpha = \beta = 0$ (e.g., when λ_t, μ_t are held fixed), then $\boldsymbol{W}^\dagger = \boldsymbol{W}^\star$ and $\boldsymbol{W}_t^*$ converges to the least-squares limiting minimizer. If $\alpha > 0$ and/or $\beta > 0$, then $\boldsymbol{W}^\dagger$ interpolates between the data-driven optimum $\boldsymbol{W}^\star$ and the anchor constraints encoded by $(\boldsymbol{W}_0, \boldsymbol{K}_0, \boldsymbol{V}_0)$: larger α shrinks $\boldsymbol{W}^\dagger$ toward $\boldsymbol{W}_0$, and larger β enforces $\boldsymbol{K}_0\boldsymbol{W}^\dagger \approx \boldsymbol{V}_0$ even as $t \to \infty$.

Method	Model	Efficacy ↑	Generalization ↑	Specificity ↑	Fluency ↑	Consistency ↑
RLSEdit (Ours)	Llama-3-8B	**89.94**$_{\pm 0.75}$	**72.84**$_{\pm 1.21}$	**60.56**$_{\pm 0.35}$	**615.58**$_{\pm 4.34}$	**26.27**$_{\pm 0.35}$
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**$_{\pm 1.07}$	68.55$_{\pm 0.47}$	73.37$_{\pm 0.44}$	**625.74**$_{\pm 0.71}$	31.62$_{\pm 0.81}$
AlphaEdit		94.10$_{\pm 0.42}$	**70.29**$_{\pm 2.30}$	**75.29**$_{\pm 0.65}$	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**$_{\pm 3.59}$
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, comparing **RLSEdit** with the baselines. We report mean ± 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 results are highlighted in bold, and the second-best results are underlined.

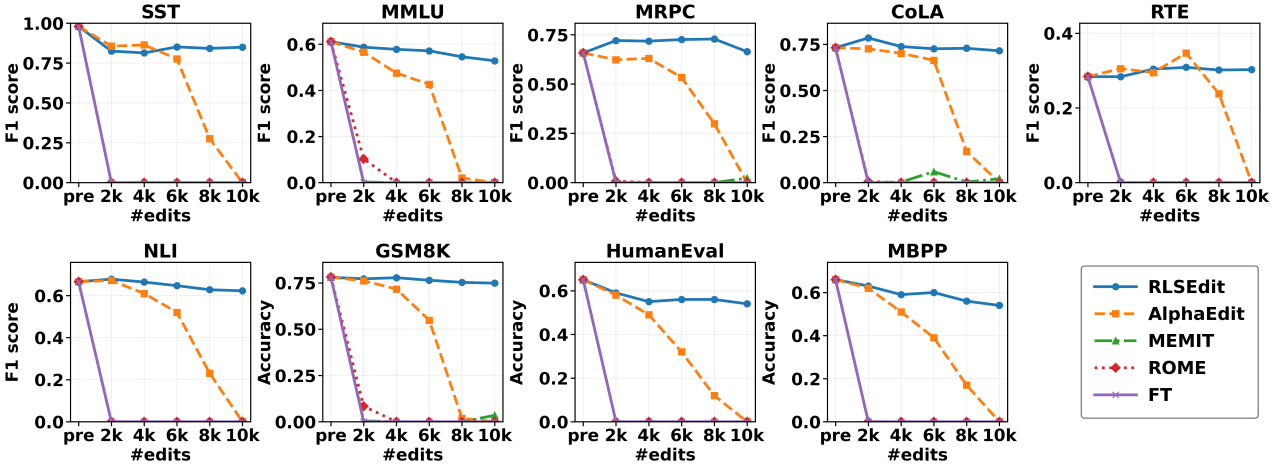

Figure 3: **General capability preservation.** We evaluate 5 GLUE tasks and additional benchmarks for general knowledge, math reasoning and coding ability (MMLU, GSM8K, HumanEval, MBPP) at multiple editing checkpoints (Pre-edit, 2k–10k edits). **RLSEdit** is compared against baselines and consistently 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).

Thus, increasing λ and μ makes the update more conservative. This improves preservation, but can reduce edit adherence when the new edit conflicts with the original mapping. A common policy is to set a deviation budget for the associated penalties, then tune (λ, μ) to satisfy both bounds.

The limits $\mu, \lambda \to \infty$ yield hard anchoring ($\boldsymbol{K}_0\boldsymbol{W} = \boldsymbol{V}_0$) and frozen parameters ($\boldsymbol{W}_t^* \to \boldsymbol{W}_0$).

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 mainly 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). To further test whether the trend holds beyond CounterFact, we also evaluate on **ZsRE**.

5.2 Main Results

Editing Results. Table 1 reports performance after 10K edits with batch size 100 on CounterFact. On Llama3-8B, **RLSEdit** achieves the best scores across all five metrics. In particular, it obtains a clear improvement in Efficacy over the second-best method (89.94 vs. 74.76), while also improving Generalization, Specificity, Fluency, and Consistency.

On Qwen2.5-7B, the results are more nuanced. **RLSEdit** achieves the best Efficacy and Fluency, while AlphaEdit gives slightly higher Generalization and Specificity, and ROME gives the best Consistency. This difference suggests that the editing–preservation trade-off is model-dependent rather than evidence of instability in **RLSEdit**. Overall, **RLSEdit** remains competitive with AlphaEdit on Qwen2.5-7B and substantially outperforms ROME, MEMIT, and FT on most metrics.

ZsRE Results. We further test **RLSEdit** on ZsRE [Levy *et al.*, 2017]. As shown in Table 2, **RLSEdit** achieves 81.20 Ef-

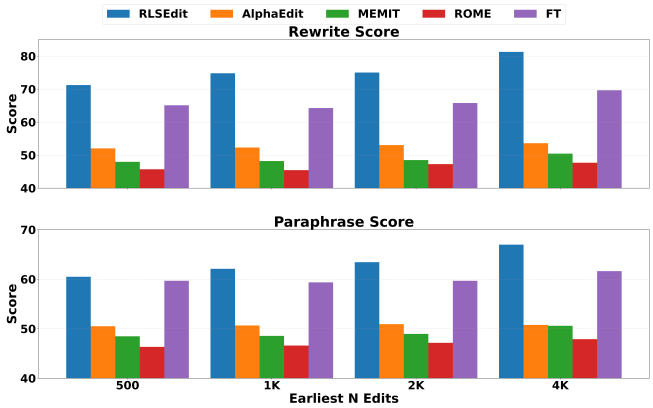

Figure 4: Improvements on early edits. After applying 10K sequential edits, we re-evaluate performance on the earliest edited cases (500, 1K, 2K, 4K). Each bar reports the Rewrite or Paraphrase score. **RLSEdit** consistently achieves the highest scores across all settings.

Metric	ROME	MEMIT	AlphaEdit	RLSEdit
ZsRE Efficacy ↑	1.52	0.96	70.04	**81.20**

Table 2: ZsRE results on `Llama3-8B`. **RLSEdit** shows the same trend beyond CounterFact.

ficacy on `Llama3-8B`, outperforming ALPHAEDIT, MEMIT, and ROME. This supports the generality of the empirical trend across different editing datasets.

General Capability Results. To assess how well editing methods preserve the pre-edited model's general abilities, we evaluate five tasks from GLUE [Wang *et al.*, 2018], together with additional benchmarks for *general knowledge* (MMLU), *math reasoning* (GSM8K), and *coding ability* (HumanEval, MBPP). We conduct the evaluation on multiple editing checkpoints of the `Llama3-8B` model, using 10K total edits with a batch size of 100.

Figure 3 summarizes the general capability evaluations. Across the evaluated language understanding, knowledge, math, and coding benchmarks, **RLSEdit** better preserves the model's general capabilities throughout the editing trajectory. This is especially important in the long sequential setting, where many edits may accumulate interference over time. In contrast, MEMIT, ROME, and FT show stronger degradation as edits accumulate. ALPHAEDIT performs competitively in the early stages, but drops more noticeably after a large number of edits. These results suggest that **RLSEdit** provides a better balance between edit reliability and general capability preservation.

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, where $N \in \{500, 1\text{K}, 2\text{K}, 4\text{K}\}$, after performing 10K sequential edits with 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, the baselines remain noticeably lower, 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 3: 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. We compare **RLSEdit** with ALPHAEDIT.

Effect of Regularization. The two regularization parameters play different roles. The parameter λ controls the drift from the original weights $\boldsymbol{W}_0$, while μ controls the anchor mapping $\boldsymbol{K}_0\boldsymbol{W}$. We find that the effect of λ is model-dependent. On `Llama3-8B`, setting $\lambda = 0$ while keeping $\mu = 12000$ reduces the Efficacy score to 87.0, with the other metrics also slightly lower. This suggests that the parameter-drift penalty improves the editing–preservation trade-off for this backbone. In contrast, `Qwen2.5-7B` performs best with $\lambda = 0$, indicating that this model is more robust to direct weight changes.

The anchor parameter μ is important in practice. Removing it can make the linear system ill-conditioned in long sequential editing. For a new backbone, we therefore tune (λ, μ) using a small pilot sweep and monitor the three objective terms: edit fitting, parameter drift, and anchor-mapping error.

Speed-up Analysis. Table 3 reports the update computation time for **RLSEdit** and ALPHAEDIT across two model backbones and three batch sizes. Across all six configurations, **RLSEdit** runs faster, reducing update time by $1.37\times$–$1.79\times$ relative to ALPHAEDIT. This is consistent with the theoretical time-complexity analysis in Section 3.2.

6 Conclusion

Existing model editing methods often degrade when the number of edits becomes large. We propose **RLSEdit**, a recursive least-squares framework for long sequential editing with soft editing and soft preservation constraints. RLSEdit introduces two regularization terms to control deviation from the pre-trained weights and from an anchor mapping, allowing it to balance edit adherence and preservation. The resulting objective admits an efficient Woodbury-based recursion whose per-edit update cost is independent of the edit history length.

Empirically, RLSEdit scales stably to 10K edits on `Llama-3` and `Qwen2.5`, achieves strong results on CounterFact and ZsRE, and better preserves general capabilities across language understanding, knowledge, math, and coding benchmarks. These results support RLSEdit as a practical approach for continuous model editing. A remaining limitation is that our experiments focus on 7B–8B backbones; evaluating larger models and more failure cases is an important direction for future work.

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.

Contribution Statement

Xinyu Wang, Sicheng Lyu, and Yu Gu contributed equally to this work. Xiao-Wen Chang is the corresponding author.

References

[Chen *et al.*, 2024] Qizhou Chen, Taolin Zhang, Xiaofeng He, Dongyang Li, Chengyu Wang, Longtao Huang, and Hui Xue'. Lifelong knowledge editing for llms with retrieval-augmented continuous prompt learning. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung 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.

[De Cao *et al.*, 2021] Nicola De Cao, Wilker Aziz, and Ivan 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.

[DeepSeek-AI and others, 2025] DeepSeek-AI et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. arXiv preprint arXiv:2501.12948.

[Deng *et al.*, 2025] Jingcheng Deng, Zihao Wei, Liang Pang, Hanxing Ding, Huawei Shen, and Xueqi 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.

[Dubey and others, 2024] Abhimanyu Dubey et al. The llama 3 herd of models, 2024. arXiv preprint arXiv:2407.21783.

[Fang *et al.*, 2025] Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Jie Shi, Xiang Wang, Xiangnan He, and Tat-Seng 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.

[Gupta *et al.*, 2024] Akshat Gupta, Anurag Rao, and Gopala 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, August 2024. Association for Computational Linguistics.

[Hager, 1989] William W. Hager. Updating the inverse of a matrix. *SIAM Review*, 31(2):221–239, 1989.

[Hartvigsen *et al.*, 2023] Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. Aging with GRACE: lifelong model editing with discrete key-value adaptors. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*, 2023.

[Jiang *et al.*, 2025] Houcheng Jiang, Junfeng Fang, Ningyu Zhang, Mingyang Wan, Guojun Ma, Xiang Wang, Xiangnan He, and Tat-Seng Chua. AnyEdit: Edit any knowledge encoded in language models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 27510–27533. PMLR, 13–19 Jul 2025.

[Kirkpatrick *et al.*, 2017] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[Levy *et al.*, 2017] Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. In Roger Levy and Lucia Specia, editors, *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 333–342, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[Lyu *et al.*, 2025] Sicheng Lyu, Yu Gu, Xinyu Wang, Jerry Huang, Sitao Luan, Yufei Cui, Xiao-Wen Chang, and Peng Lu. Evoedit: Evolving null-space alignment for robust and efficient knowledge editing, 2025. Accepted to Findings of ACL 2026.

[Meng *et al.*, 2022] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle 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.

[Meng *et al.*, 2023] Kevin Meng, Arnab Sen Sharma, Alex J. Andonian, Yonatan Belinkov, and David 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.

[Mitchell *et al.*, 2022] Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. Fast model editing at scale. In *International Conference on Learning Representations (ICLR)*, 2022.

[OpenAI, 2023] OpenAI. GPT-4 technical report, 2023. arXiv preprint arXiv:2303.08774.

[Ritter *et al.*, 2018] Hippolyt Ritter, Aleksandar Botev, and David Barber. Online structured laplace approximations

for overcoming catastrophic forgetting. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[Sayed, 2003] Ali H. Sayed. *Fundamentals of Adaptive Filtering*. Wiley-IEEE Press, 2003.

[Sherman and Morrison, 1950] Jack Sherman and Winifred 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.

[Sun *et al.*, 2025] Wei Sun, Tingyu Qu, Mingxiao Li, Jesse Davis, and Marie-Francine Moens. Mitigating negative interference in multilingual knowledge editing through null-space constraints. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics: ACL 2025*, pages 8796–8810, Vienna, Austria, July 2025. Association for Computational Linguistics.

[Wang *et al.*, 2018] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In Tal Linzen, Grzegorz Chrupała, and Afra Alishahi, editors, *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[Wang *et al.*, 2024] Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. WISE: rethinking the knowledge memory for lifelong model editing of large language models. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng 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.

[Wang *et al.*, 2025] Xinyu Wang, Linrui Ma, Jerry Huang, Peng Lu, Prasanna Parthasarathi, Xiao-Wen Chang, Boxing Chen, and Yufei Cui. Resona: Improving context copying in linear recurrence models with retrieval, 2025.

[Woodbury, 1950] Max A. Woodbury. Inverting modified matrices. Memorandum Report 42, Statistical Research Group, Princeton University, 1950.

[Yang and others, 2024] An Yang et al. Qwen2.5 technical report, 2024. arXiv preprint arXiv:2412.15115.

[Zhu *et al.*, 2020] Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix X. Yu, and Sanjiv Kumar. Modifying memories in transformer models, 2020. arXiv preprint arXiv:2012.00363.