LIE-TUNING: EXACT LOW-RANK MATRIX FLOWS FOR RANK-PRESERVING ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Parameter-efficient fine-tuning must balance adaptation capacity against preservation of a pretrained linear map. LoRA constrains the rank of an *additive update*, but $W_\theta = W_0 + BA$ does not guarantee $\mathrm{rank}(W_\theta) = \mathrm{rank}(W_0)$. Orthogonal multiplicative methods provide that guarantee, but they also preserve the entire singular spectrum and therefore exclude anisotropic scaling.

We propose **Lie-Tuning**, a framework that decouples *rank preservation* from *spectral rigidity*. It adapts a frozen map through invertible general-linear actions, $W_\theta = \exp(\xi_L)W_0$ or $W_\theta = W_0 \exp(\xi_R)$. Invertibility preserves the rank of W_0 exactly, while a general, non-skew-symmetric generator can change singular values through scaling and shearing. The two action sides provide complementary inductive biases: left multiplication post-composes the pretrained map in its output space, whereas right multiplication pre-composes it in its input space. We instantiate this framework with **MExRA**, whose generator $\xi = BA$ is low rank, and **MExBA**, whose generator is block diagonal. For MExRA, we derive the exact identity $\exp(BA) = I + B\phi_1(AB)A$. It retains all orders of the exponential while reducing an ambient $d \times d$ exponential to an $r \times r$ kernel, requiring $O(dr)$ trainable parameters and $O(dr^2 + r^3)$ kernel construction. The same ϕ_1 representation has a second computational consequence: it forms $\exp(\xi) - I$ without explicitly subtracting the identity, avoiding near-identity cancellation and directly representing the first variation about exact identity initialization. This matrix analogue of `expm1` makes the residual numerically resolvable during training. The learned action remains mergeable for inference. On LLaMA-2-7B, MExRA reaches 25.00% HumanEval Pass@1 in the regularized setting versus LoRA's 21.95%. On GSM8K, its three-seed mean accuracy is 35.28%, compared with 33.46% for LoRA and 29.11% for OFT. Right-acting MExBA achieves 46.11% German Belebele accuracy versus 43.78% for LoRA. MExRA also achieves the highest GSM8K accuracy among the compared methods throughout sequential adaptation, while numerical tests verify accurate exponential residual evaluation near identity.

1 INTRODUCTION

The pre-training and fine-tuning paradigm has made parameter-efficient fine-tuning (PEFT) central to adapting large language models. LoRA (Hu et al., 2022) is especially attractive because it learns only a low-rank additive update, $W_\theta = W_0 + BA$. The low-rank constraint applies to BA, however, not to the adapted map W_θ: addition can change the rank and null-space structure inherited from W_0.

Multiplicative adaptation offers a different contract. If G is invertible, then GW_0 has exactly the same rank as W_0. Orthogonal methods such as OFT, BOFT, and Householder Reflection Adaptation exploit this principle with orthogonal G (Qiu et al., 2023; Liu et al., 2024; Yuan et al., 2024). Yet orthogonality preserves more than rank. For a left action QW_0, $Q^\top Q = I$ implies $(QW_0)^\top (QW_0) = W_0^\top W_0$; the analogous identity holds for a right action. Thus every singular value is fixed. This yields a clear design tension: additive low-rank tuning permits spectral change without guaranteeing the rank of the pretrained map, whereas orthogonal tuning guarantees rank but forbids spectral change.

Lie-Tuning. We resolve this tension by letting elements of a matrix group act on the frozen linear map; the possibly rectangular matrix W_0 itself need not be a group element. The general-linear group

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

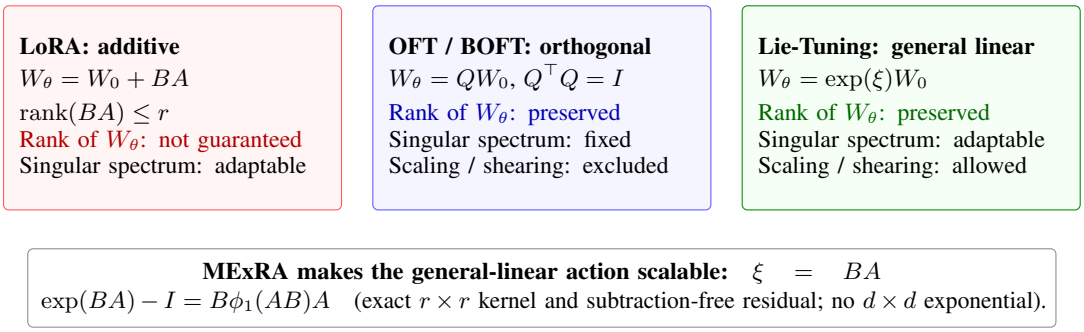

Figure 1: **Rank preservation and spectral flexibility are distinct design choices.** LoRA constrains the rank of the additive update but not the rank of the adapted map. Orthogonal multiplication preserves rank and all singular values. Lie-Tuning uses an invertible general-linear action to preserve rank without requiring the singular spectrum to remain fixed. The right-action versions obey the same comparison.

can act on either the codomain or the domain:

$$W_\theta = \exp(\xi_L)W_0 \quad \text{or} \quad W_\theta = W_0 \exp(\xi_R). \tag{1}$$

Every matrix exponential is invertible, so either action preserves $\mathrm{rank}(W_0)$ exactly. Unlike an orthogonal action, a general-linear action does not freeze the singular spectrum: its symmetric component can rescale feature directions, while its non-normal component can shear them. Left multiplication post-composes W_0 in the output space; right multiplication pre-composes it in the input space. We treat their task associations as inductive-bias hypotheses and test both directions rather than asserting a universal rule.

Exact and scalable matrix flows. A direct $d \times d$ matrix exponential costs $O(d^3)$. Our main instantiation, **MExRA**, instead uses the low-rank generator $\xi = BA$ and an exact reduction to the small kernel AB. The identity $\exp(BA) = I + B\phi_1(AB)A$ retains every order of the exponential, remains defined when AB is singular, and yields a residual implementation with $O(dr)$ parameters. The same ϕ_1 representation has two distinct computational consequences: small-kernel reduction makes the exponential affordable, while direct residual formation makes its near-identity update numerically resolvable without explicitly forming $\exp(\xi) - I$. We complement MExRA with **MExBA**, which trades global low-rank structure for full-rank transformations inside local blocks. Skew-symmetric restrictions give orthogonal specializations, making group, structure, and action side explicit design axes within one framework.

Our contributions are as follows:

1. **Invariant and design space.** We formalize left and right invertible actions on rectangular pretrained maps, prove their exact rank and subspace invariants, and separate rank preservation from the stronger isometry imposed by orthogonal tuning.

2. **Exact small-kernel flows.** We reduce the exact exponential of a low-rank generator to an $r \times r$ ϕ_1 kernel, retaining every order of the flow with $O(dr)$ trainable parameters and without an ambient matrix exponential.

3. **Cancellation-free near-identity residuals.** We form $\exp(\xi) - I$ through $\xi\phi_1(\xi)$ and extract ϕ_1 from an augmented small-kernel exponential, without explicitly subtracting the identity or inverting the kernel. This represents the zero residual exactly and directly exposes its first variation about identity initialization.

4. **Directional evidence.** Experiments across code generation, cross-lingual transfer, and continual adaptation evaluate when output-side and input-side actions are useful, with MExRA variants attaining the strongest reported results in the settings highlighted above.

2 Related Work

Parameter-efficient and multiplicative adaptation. PEFT adapts pretrained models with small task-specific components, including bottleneck adapters (Houlsby et al., 2019; Pfeiffer et al., 2020a;b), prompt-based methods (Li & Liang, 2021; Lester et al., 2021; Liu et al., 2022), sparse updates (Ben Zaken et al., 2022), and structured adapters (Karimi Mahabadi et al., 2021; Edalati et al., 2025). LoRA (Hu et al., 2022) established additive low-rank updates as a scalable default, followed by adaptive-rank and quantized variants (Zhang et al., 2023; Dettmers et al., 2023). LoRMA directly studies multiplicative low-rank adaptation (Bihany et al., 2025). Lie-Tuning is distinguished by an invertible group action with an explicit rank invariant and by exact evaluation of its low-rank exponential flow.

Orthogonal and group-structured fine-tuning. Orthogonality has long been used to control spectral behavior in neural networks (Arjovsky et al., 2016; Wisdom et al., 2016; Vorontsov et al., 2017; Helfrich et al., 2018). OFT, BOFT, and Householder Reflection Adaptation bring orthogonal multiplication to foundation-model adaptation (Qiu et al., 2023; Liu et al., 2024; Yuan et al., 2024). These methods already preserve matrix rank; our claim is not that rank preservation is unique to Lie-Tuning. Their purely orthogonal variants preserve the full singular spectrum, whereas a general-linear action preserves rank while permitting anisotropic scaling and shearing. Lie-Tuning includes a skew-symmetric, special-orthogonal restriction for settings where isometry is the desired prior.

Lie transformations for PEFT. Most closely, Si et al. (2025) formulate tensor-valued parameters as an Abelian Lie group under element-wise multiplication and propose $W \leftarrow W \odot \exp(\Delta W)$. Their efficient LieRA realization uses the first-order approximation $W + W \odot \Delta W$, primarily retaining the locality of high-order tensor structure. Lie-Tuning instead uses ordinary left or right matrix multiplication by elements of non-commutative matrix groups. MExRA evaluates the exponential exactly—all higher-order terms are retained—through a small $r \times r$ kernel, and the resulting invertible matrix action preserves the rank of the frozen linear map.

Matrix functions and manifold parameterizations. Our formulation builds on optimization and parameterization over matrix manifolds (Absil et al., 2008; Hall, 2015; Lezcano-Casado, 2019) and classical algorithms for matrix functions and exponential-related integrals (Van Loan, 1978; Higham, 2008; Al-Mohy & Higham, 2011). The technical role of the low-rank factorization here is to reduce the exact ambient exponential to a small matrix function, rather than to approximate the exponential at first order.

3 Methodology

We first state the invariant that motivates Lie-Tuning, then separate its three design axes: action side, transformation group, and generator structure. We subsequently give exact, scalable realizations for low-rank and block-structured generators.

3.1 Lie-Tuning: Rank-Preserving Left and Right Actions

Let $W_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ denote a frozen linear map. Lie-Tuning learns a generator in the Lie algebra of a transformation group acting on the codomain or domain of this map:

$$\begin{aligned} W_\theta^{\text{left}} &= G_L W_0, \quad G_L = \exp(\xi_L) \in GL^+(d_{\text{out}}, \mathbb{R}), \\ W_\theta^{\text{right}} &= W_0 G_R, \quad G_R = \exp(\xi_R) \in GL^+(d_{\text{in}}, \mathbb{R}). \end{aligned} \tag{2}$$

Here GL^+ denotes the identity component of the general linear group. This action-based formulation applies without requiring the generally rectangular or rank-deficient W_0 itself to belong to a general linear group.

Because matrix exponentials are invertible, both actions preserve the rank of W_0 exactly, including for rectangular or rank-deficient weights and under ordered composition. Left action also preserves $\ker(W_0)$, while right action preserves $\text{im}(W_0)$. Orthogonal actions additionally fix all singular values; general-linear exponential actions allow them to change. Lie-Tuning thus retains the rank

Table 1: **Independent design axes in Lie-Tuning.** MExRA and MExBA choose a generator structure; either can use a left or right action and a general-linear or orthogonal group.

Axis	Choices	Effect
Action side	Left / right	Output post-composition / input pre-composition
Group	GL^+ / SO	Rank + flexible spectrum / rank + fixed spectrum
Generator	Low rank / block	Global channel mixing / local full-rank mixing

guarantee of multiplicative orthogonal methods without their fixed-spectrum constraint. Appendix A gives the formal statements and proofs (Propositions 2 and 3).

Left action post-composes the pretrained map and modifies its output representation; right action pre-composes it and modifies the representation presented to W_0. These are distinct hypotheses about where adaptation should occur, not consequences of rank preservation alone.

3.2 MExRA: Matrix-Exponential Rank Adaptation

3.2.1 Low-Rank Generator

Optimizing an unconstrained $d \times d$ generator would require d^2 trainable parameters. MExRA instead parameterizes

$$\xi = BA, \qquad B \in \mathbb{R}^{d\times r}, \quad A \in \mathbb{R}^{r\times d}, \tag{3}$$

where $d = d_{\text{out}}$ for a left action and $d = d_{\text{in}}$ for a right action. Any scalar generator scaling can be absorbed into either low-rank factor, so we omit it from the mathematical notation; Appendix F records the scaling used in the implementation. We initialize $B = 0$, while A follows the direction- and group-specific initialization described in Section 3.6. Consequently, $\xi = 0$ and training begins exactly at $G = \exp(\xi) = I$.

3.2.2 Exact Small-Kernel Exponential

A direct evaluation of $\exp(BA)$ would operate on a $d \times d$ matrix. The following identity moves the only matrix-function evaluation to the $r \times r$ kernel $M = AB$ and does not require M to be invertible.

Proposition 1 (Exact low-rank matrix exponential)**.** *Define the first ϕ-function by the entire matrix power series*

$$\phi_1(M) = \sum_{k=0}^{\infty} \frac{M^k}{(k+1)!}. \tag{4}$$

Then, for every $A \in \mathbb{R}^{r\times d}$ and $B \in \mathbb{R}^{d\times r}$,

$$\exp(BA) = I_d + B\phi_1(AB)A. \tag{5}$$

Proof. For $k \geq 1$, $(BA)^k = B(AB)^{k-1}A$. Substitution into the exponential series gives

$$\exp(BA) = I_d + \sum_{k=1}^{\infty} \frac{B(AB)^{k-1}A}{k!} \tag{6}$$

$$= I_d + B\left(\sum_{j=0}^{\infty} \frac{(AB)^j}{(j+1)!}\right) A, \tag{7}$$

which is Eq. 5. The series converges for every square matrix, including singular AB. □

When M is invertible, $\phi_1(M) = (\exp(M) - I)M^{-1}$, but Eq. 4 is the universal definition and avoids making invertibility part of the method. Section 3.5 shows how evaluating ϕ_1 without forming $\exp(M) - I$ supplies the cancellation-free residual used here; Appendix C gives the complete numerical derivation.

We never form the ambient exponential during training. With $\Phi_L = \phi_1(A_L B_L)$ and $h_0 = W_0 x$, left MExRA computes

$$y = h_0 + B_L \Phi_L A_L h_0. \tag{8}$$

With $\Phi_R = \phi_1(A_R B_R)$, right MExRA computes

$$y = W_0\big(x + B_R \Phi_R A_R x\big). \tag{9}$$

Both are exact evaluations of Eq. 2; "low rank" describes the generator parameterization, while Proposition 2 governs the rank of the adapted pretrained map.

Complexity and mergeability. MExRA has $2dr$ trainable parameters. Forming AB costs $O(dr^2)$ and evaluating its small matrix function costs $O(r^3)$. Once Φ is available, applying the residual to one feature vector costs $O(dr + r^2)$. At inference, the learned invertible action can be multiplied into W_0, leaving no separate adapter branch.

3.3 MExBA: Matrix-Exponential Block Adaptation on $GL(d, \mathbb{R})$

MExBA provides a complementary structural prior. Whereas MExRA mixes all ambient channels through a globally supported but low-rank generator, MExBA permits a full-rank generator inside each local feature block but prevents cross-block interaction. Specifically,

$$\xi = \mathrm{diag}(M_1, \ldots, M_k), \qquad M_i \in \mathbb{R}^{b \times b}, \qquad k = d/b. \tag{10}$$

As with MExRA, d is chosen from the output or input dimension according to the action side.

The exponential preserves this structure:

$$\exp(\xi) = \mathrm{diag}(\exp(M_1), \ldots, \exp(M_k)). \tag{11}$$

MExBA therefore uses db parameters, $O(db^2)$ work to exponentiate all blocks, and $O(db)$ work to apply them to a feature vector. Each exponential block is invertible, so the global action retains the rank guarantee while allowing scaling and shearing within blocks.

3.4 Orthogonal Specializations

To isolate the effect of the transformation group while retaining the same action-side and structural choices, we also consider skew-symmetric generators. If $\xi^\top = -\xi$, then $\exp(\xi) \in SO(d)$ and $\exp(\xi)^\top \exp(\xi) = I$. These variants share the rank guarantee but deliberately restore the fixed-spectrum constraint characterized in Proposition 3.

O-MExRA (Orthogonal Matrix-Exponential Rank Adaptation). We use

$$\xi = BA^\top - AB^\top, \qquad A, B \in \mathbb{R}^{d \times r}. \tag{12}$$

With $U = [B, -A]$ and $V = [A, B]$, we have $\xi = UV^\top$, so Proposition 1 applies through the $2r \times 2r$ kernel $V^\top U$.

O-MExBA (Orthogonal Matrix-Exponential Block Adaptation). O-MExBA restricts each local block generator to be skew-symmetric: $\xi = \mathrm{diag}(Q_1, \ldots, Q_k)$, where $Q_i = -Q_i^\top \in \mathbb{R}^{b \times b}$.

This is an exponential-map specialization related to, but not a claim to subsume every parameterization used by, OFT, BOFT, or reflection-based methods. Appendix E compares the exponential and Cayley realizations.

3.5 Subtraction-Free Exponential Residuals via ϕ_1

The small-kernel identities above make the exponential affordable, but affordability alone does not provide a numerically faithful residual. Lie-Tuning starts from the identity, so its exponential residual $\Delta(\xi) = \exp(\xi) - I$ is initially zero and remains in the cancellation-prone regime whenever early iterates stay near the initialization. Computing $\exp(\xi)$ first and then subtracting I attempts to recover an $O(\|\xi\|)$ update after materializing an $O(1)$ matrix. In finite precision, error incurred at the scale

of the computed exponential can therefore dominate the residual even when the exponential itself is accurate in absolute terms.

We instead compute the exponential remainder directly through the exact identity

$$\Delta(\xi) = \exp(\xi) - I = \xi\phi_1(\xi) = \phi_1(\xi)\xi. \tag{13}$$

This is the matrix analogue of the scalar `expm1` operation: just as `expm1` returns $e^z - 1$ without first rounding e^z near one, Eq. 13 forms the matrix remainder without obtaining it by subtracting I from a separately rounded $\exp(\xi)$. Because ϕ_1 is entire and $\phi_1(0) = I$, the expression is defined at the identity and for singular generators, without explicitly subtracting I or inverting ξ. For MExRA, this gives Eqs. 8–9; Appendix C.2 collects the residual forms for all variants.

For any reduced kernel $M \in \mathbb{R}^{k\times k}$, $\phi_1(M)$ itself can be extracted without inverting M or explicitly forming $\exp(M) - I$:

$$\exp\left(\begin{bmatrix} M & I_k \\ 0 & 0 \end{bmatrix}\right) = \begin{bmatrix} \exp(M) & \phi_1(M) \\ 0 & I_k \end{bmatrix}. \tag{14}$$

Thus the exact small-kernel reduction and the subtraction-free remainder address two distinct computational obstacles: the former removes the ambient exponential, while the latter represents the near-identity update at its natural $O(\|\xi\|)$ scale. More precisely, $\|\Delta(\xi) - \xi\| = O(\|\xi\|^2)$ as $\xi \to 0$, so the tangent update is represented directly rather than recovered from two nearly equal matrices. Appendix C develops the matrix-`expm1` analogy, the finite-precision error analysis, and the derivatives at identity initialization.

3.6 Initialization Strategy: Strict Identity and Orthogonal Basis

Lie-Tuning starts from a strict identity action: block generators are initialized to zero, and low-rank variants set $B = 0$. The remaining basis A controls the initial gradient directions. In our implementation, left MExRA uses orthonormal rows ($AA^\top = I_r$), right MExRA uses Kaiming initialization on normalized inputs, and O-MExRA uses orthonormal columns. These are implementation choices rather than prerequisites for Proposition 2; Appendix F gives their scaling rationale.

Initialization-preserving asymmetric weight decay. For factorized MExRA and O-MExRA adapters, including their right-multiplied variants, we use asymmetric AdamW parameter groups: the reported coefficient λ is applied only to the zero-initialized factor B, while A receives no weight decay. This choice protects the deliberately structured nonzero factor from direct decay-induced contraction after identity initialization; subsequent task gradients may still update A. Appendix F.3 gives the initial-gradient and first-step analysis.

Although the main paper focuses on bias-free Transformer projections, the same framework extends to affine layers with bias terms by lifting the weight map to the affine group; see Appendix B.

4 Experiments

4.1 Experimental Setup

We evaluate code generation, mathematical reasoning, cross-lingual transfer, continual adaptation, and near-identity residual accuracy. Unless otherwise stated, adaptation freezes **LLaMA-2-7B** Touvron et al. (2023) and uses three training epochs, learning rate 3×10^{-4}, global batch size 16, rank $r = 16$, weight decay 10^{-3}, and block size $2r = 32$ for block-structured methods. We compare standard **LoRA** Hu et al. (2022) and **OFT** Qiu et al. (2023) with MExRA, MExBA, and their orthogonal variants. Code generation and cross-lingual transfer additionally test right-multiplied variants. For nonzero weight decay, MExRA and O-MExRA on either side use $\lambda_B = \lambda$ and $\lambda_A = 0$ (Section 3.6); LoRA, OFT, and all block variants use standard AdamW decay.

Code generation, cross-lingual transfer, and continual adaptation report single-run checkpoints, with epoch-wise trajectories where applicable. Mathematical reasoning reports mean and sample standard deviation over seeds 42, 43, and 44 at each of three epochs, without weight decay or other regularization. Numerical repetitions and aggregation are specified in Section 4.6. HumanEval has 164 problems (about 0.61 percentage points each), so small differences warrant caution.

Table 2: **HumanEval Pass@1 after CodeAlpaca fine-tuning.** Scores (%) over three epochs with weight decay 0 or 10^{-3}.

Method	wd=0.0			wd=0.001		
	Ep. 1	Ep. 2	Ep. 3	Ep. 1	Ep. 2	Ep. 3
Base Model		11.59			11.59	
LoRA	16.46	21.34	20.73	18.29	21.95	21.95
OFT	13.41	17.68	19.51	14.02	18.29	18.90
MExRA	16.46	23.78	**24.39**	16.46	23.17	**25.00**
MExBA	18.29	23.78	23.78	18.29	24.39	23.17
O-MExRA	15.85	21.95	21.34	18.29	17.68	20.12
O-MExBA	20.12	21.95	21.34	18.29	21.95	21.34
R-MExRA	19.51	21.34	19.51	16.46	20.73	21.34
R-MExBA	18.90	20.73	21.34	17.68	23.17	21.34
R-O-MExRA	17.07	20.73	22.56	15.24	22.56	19.51
R-O-MExBA	17.07	22.56	21.95	16.46	21.34	20.73

4.2 Code Generation

Setup. We fine-tune on **CodeAlpaca-20k** Chaudhary (2023) and evaluate zero-shot functional correctness on **HumanEval** Chen et al. (2021). Generation uses greedy decoding with temperature 0. We evaluate two regularization settings for this task: no weight decay and weight decay 10^{-3}.

Results. Table 2 shows that the general linear variants are the strongest performers on HumanEval. With weight decay 10^{-3}, MExRA reaches the best score of 25.00% at epoch 3, improving over LoRA's best score of 21.95% and OFT's best score of 18.90%. The same pattern holds without weight decay, where MExRA reaches 24.39%, compared with 21.34% for LoRA, 19.51% for OFT, and 23.78% for MExBA.

These results support the central motivation for moving beyond strictly orthogonal updates on tasks that require selective amplification of task-relevant features. The $SO(d)$ variants are competitive but do not match the $GL(d)$ variants, suggesting that exact norm preservation is too restrictive for code generation. Right-multiplied variants are also useful, but their strongest scores are below the left-multiplied MExRA peak in this task, consistent with the view that code instruction tuning primarily changes output behavior rather than input-domain alignment.

4.3 Mathematical Reasoning

Purpose and training protocol. We compare LoRA, OFT, and left-acting MExRA on English **GSM8K** (Cobbe et al., 2021) using the same frozen LLaMA-2-7B backbone. All adapters target the query and output projections. LoRA and MExRA use rank 16 with 8,388,608 trainable parameters; OFT uses block size 64 with 8,257,536 trainable parameters. Training uses three epochs, learning rate 2×10^{-4}, global batch size 16, sequence length 1,024, no regularization, and zero weight decay. The GSM8K training partition contains 7,099 examples after reserving 374 validation examples. We report all three epoch checkpoints for seeds 42, 43, and 44.

Evaluation. We evaluate the trained mathematics adapters on all 1,319 English GSM8K test questions using greedy, zero-shot chain-of-thought generation with a common 512-token output limit and numeric exact match.

Results. Table 3 shows that MExRA reaches $35.28 \pm 0.98\%$ at the final epoch, exceeding OFT's $29.11 \pm 1.18\%$ by 6.17 percentage points and LoRA's $33.46 \pm 1.23\%$ by 1.82 points under comparable trainable-parameter budgets. MExRA outperforms OFT at every epoch, supporting the effectiveness of rank-preserving general-linear adaptation for mathematical reasoning.

Table 3: **Mathematical reasoning on GSM8K.** Accuracy (%), mean ± SD over three seeds, for every trained epoch. LoRA/MExRA use 8,388,608 trainable parameters; OFT uses 8,257,536.

Method	Epoch 1	Epoch 2	Epoch 3
LoRA	27.98 ± 0.84	32.12 ± 0.80	33.46 ± 1.23
OFT	26.79 ± 0.64	28.66 ± 0.42	29.11 ± 1.18
MExRA	27.93 ± 1.95	33.86 ± 0.80	35.28 ± 0.98

Table 4: **Cross-lingual transfer.** Accuracy (%): macro-average across C-Eval subjects and accuracy on the German Belebele split.

Method	**C-Eval Macro**			**Belebele-de**		
	Ep. 1	**Ep. 2**	**Ep. 3**	**Ep. 1**	**Ep. 2**	**Ep. 3**
Base Model		31.61			40.67	
LoRA	32.29	31.32	31.63	39.67	42.89	43.78
OFT	30.34	31.23	32.33	32.56	35.11	35.33
MExRA	31.96	31.08	30.41	37.89	40.44	42.22
MExBA	31.52	32.74	32.03	41.11	42.89	41.33
R-MExRA	27.75	**32.83**	31.97	40.11	40.11	40.78
R-MExBA	32.48	32.49	32.52	**44.00**	**45.56**	**46.11**

4.4 Cross-Lingual Transfer

Setup. We evaluate Chinese transfer on C-Eval (Huang et al., 2023) and German transfer on Belebele-de (Bandarkar et al., 2024). For C-Eval, we report macro-average accuracy across subjects; micro-average follows the same broad trend and is omitted from the main table for compactness. All models are trained with learning rate 3×10^{-4} and weight decay 10^{-3}.

Results. Table 4 highlights a directional effect that is largely invisible in standard additive PEFT. On C-Eval, most methods cluster near the base score, with R-MExRA obtaining the best macro-average score of 32.83%. The gains are modest, so we treat this result as suggestive rather than conclusive.

The German Belebele result is clearer. R-MExBA improves from the base score of 40.67% to 46.11%, outperforming LoRA's best score of 43.78% and left-multiplied MExBA's best score of 42.89%. This supports the hypothesis that right-multiplication is well-suited to cross-lingual transfer: it adapts the representation before the frozen pretrained map acts, functioning as an input-space alignment mechanism.

4.5 Continual Adaptation

Setup. We evaluate sequential adaptation across three stages: math, code, and general instruction tuning. After each stage, adapters are merged into the base model and the next stage starts from the merged checkpoint, matching the multiplicative composition view developed in Section G. We report GSM8K accuracy after each stage to measure how well math ability is retained after subsequent code and general-instruction adaptation.

Results. Table 5 shows that MExRA gives the strongest absolute performance throughout the sequence, ending at 24.41% GSM8K accuracy compared with 23.88% for LoRA and 17.13% for OFT. The result supports multiplicative composition for sequential adaptation, while remaining limited to one task order and single-run evaluation.

4.6 Resolving Exponential Residuals Near Identity

We evaluate the computational claim in Section 3.5: the exact exponential update should remain numerically resolvable near the identity without replacing it by a first-order approximation. We compare three implementations: **Stable**, the subtraction-free $M\phi_1(M)$ construction, evaluated

Table 5: **Continual adaptation.** GSM8K accuracy after each stage of a math → code → general instruction sequence. MExRA obtains the best score after every stage and the best final retained accuracy.

Method	Step 1 Math	Step 2 Code	Step 3 General
LoRA	41.09	12.21	23.88
OFT	42.23	10.08	17.13
MExRA	**42.30**	**12.59**	**24.41**
MExBA	40.11	11.83	23.20

Table 6: **Structured matrix-`expm1` accuracy in FP32.** Five scales from the complete norm sweep, with all three implementations. Entries are relative errors, aggregated equally across matrix families and kernel sizes; lower is better.

$\|M\|_F$	Stable	Naive	Linear
10^{-8}	2.527×10^{-8}	1.762×10^{-1}	2.527×10^{-8}
10^{-6}	5.265×10^{-8}	7.408×10^{-2}	1.306×10^{-7}
10^{-4}	7.148×10^{-8}	7.603×10^{-4}	1.255×10^{-5}
10^{-2}	7.665×10^{-8}	1.002×10^{-5}	1.245×10^{-3}
1	8.098×10^{-8}	2.764×10^{-7}	1.261×10^{-1}

through the augmented exponential in Eq. 14; **Naive**, explicit $\exp(M) - I$; and **Linear**, the first-order residual M. Stable and Naive represent the same mathematical exponential, while Linear isolates its tangent update.

Structured numerical accuracy. We evaluate six matrix families: normal, non-normal, dense, skew-symmetric, nilpotent, and low-rank-product. Kernel sizes are $r \in \{4, 8, 16, 32, 64\}$, with Frobenius norms $10^{-10}, 10^{-9}, \ldots, 10^{1}$. For each family, size, norm, and storage dtype (FP64, FP32, BF16, FP16), we evaluate 100 paired matrices with all three implementations: 144,000 matrix cases and 432,000 method measurements. BF16/FP16 computations use FP32 internal arithmetic. Reference residuals use 100-decimal-digit arithmetic for $r \leq 8$ and a subtraction-free FP64 block-exponential construction otherwise. We measure Frobenius relative error against the reference, first taking the median over samples within each family–size cell and then the median across the 30 cells.

Table 6 shows the two complementary benefits of the exact residual construction. At norm 10^{-8}, Stable retains approximately FP32-level accuracy, while Naive incurs relative error 0.1762: forming an order-one exponential before subtracting the identity obscures the small remainder. This is the regime predicted by the $u/\|M\|$ amplification in Appendix C. Linear is equally accurate at this scale because higher-order terms are negligible. As the norm increases, Stable retains those terms: at norm one, its error remains 8.10×10^{-8}, compared with 0.1261 for Linear. The same implementation thus combines near-identity resolution with faithful evaluation of finite exponential flows.

5 Conclusion

Lie-Tuning adapts frozen linear maps through invertible left or right actions, preserving rank while allowing spectral change; orthogonal specializations recover the stronger isometric prior. MExRA uses the same ϕ_1 representation for an exact $r \times r$ reduction and a matrix-`expm1` residual, removing the ambient exponential and avoiding cancellation near identity initialization. MExBA provides complementary block-local transformations under the same residual principle. Across the studied settings, the results support general-linear adaptation and explicit choice of action side.

AI USE STATEMENT

We used generative AI tools to assist with writing and language polishing, including improving the clarity, grammar, and readability of the manuscript. The authors take full responsibility for the final content of this paper.

REFERENCES

Pierre-Antoine Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, NJ, 2008. ISBN 9780691132983. doi: 10.1515/9781400830244.

Awad H. Al-Mohy and Nicholas J. Higham. Computing the action of the matrix exponential, with an application to exponential integrators. *SIAM Journal on Scientific Computing*, 33(2):488–511, 2011.

Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*, 2016.

Lucas Bandarkar, Davis Liang, Benjamin Muller, Mikel Artetxe, Satya Narayan Shukla, Donald Husa, Naman Goyal, Abhinandan Krishnan, Luke Zettlemoyer, and Madian Khabsa. The belebele benchmark: A parallel reading comprehension dataset in 122 language variants. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 749–775, 2024.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, 2022.

Harsh Bihany, Shubham Patel, and Ashutosh Modi. Lorma: Low-rank multiplicative adaptation for llms. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10113–10133. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.527. URL https://doi.org/10.18653/v1/2025.findings-acl.527.

Sahil Chaudhary. Code alpaca: An instruction-following llama model for code generation. https://github.com/sahil280114/codealpaca, 2023.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems*, 2023.

Ali Edalati, Marzieh Tahaei, Ivan Kobyzev, Vahid Partovi Nia, James J. Clark, and Mehdi Rezagholizadeh. KronA: Parameter-efficient tuning with kronecker adapter. In Peyman Passban, Andy Way, and Mehdi Rezagholizadeh (eds.), *Enhancing LLM Performance: Efficacy, Fine-Tuning, and Inference Techniques*, Machine Translation: Technologies and Applications, pp. 49–65. Springer Cham, 2025. doi: 10.1007/978-3-031-85747-8_4.

Brian C. Hall. *Lie Groups, Lie Algebras, and Representations: An Elementary Introduction*. Springer, 2 edition, 2015.

Kyle Helfrich, Devin Willmott, and Qiang Ye. Orthogonal recurrent neural networks with scaled cayley transform. In *International Conference on Machine Learning*, 2018.

Nicholas J. Higham. *Functions of Matrices: Theory and Computation*. SIAM, 2008.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, 2019.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. In *Advances in Neural Information Processing Systems*, 2023.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. In *Advances in Neural Information Processing Systems*, 2021.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021.

Mario Lezcano-Casado. Trivializations for gradient-based optimization on manifolds. In *Advances in Neural Information Processing Systems*, 2019.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, 2021.

Weiyang Liu, Zeju Qiu, Yao Feng, Yuliang Xiu, Yuxuan Xue, Longhui Yu, Haiwen Feng, Zhen Liu, Juyeon Heo, Songyou Peng, Yandong Wen, Michael J. Black, Adrian Weller, and Bernhard Schölkopf. Parameter-efficient orthogonal finetuning via butterfly factorization. In *International Conference on Learning Representations*, 2024.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-Tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 61–68, 2022.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. Adapterhub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2020a.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. MAD-X: An adapter-based framework for multi-task cross-lingual transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 2020b.

Zeju Qiu, Weiyang Liu, Haiwen Feng, Yuxuan Xue, Yao Feng, Zhen Liu, Dan Zhang, Adrian Weller, and Bernhard Schölkopf. Controlling text-to-image diffusion by orthogonal finetuning. In *Advances in Neural Information Processing Systems*, 2023.

Chongjie Si, Zhiyi Shi, Xuehui Wang, Yichen Xiao, Xiaokang Yang, and Wei Shen. Generalized tensor-based parameter-efficient fine-tuning via lie group transformations. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 197–207. IEEE, Oct 2025. doi: 10.1109/ICCV51701.2025.00026. URL https://doi.org/10.1109/ICCV51701.2025.00026.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Charles F. Van Loan. Computing integrals involving the matrix exponential. *IEEE Transactions on Automatic Control*, 23(3):395–404, 1978.

Eugene Vorontsov, Chiheb Trabelsi, Samuel Kadoury, and Chris Pal. On orthogonality and learning recurrent networks with long term dependencies. In *International Conference on Machine Learning*, 2017.

Scott Wisdom, Thomas Powers, John Hershey, Jonathan Le Roux, and Les Atlas. Full-capacity unitary recurrent neural networks. In *Advances in Neural Information Processing Systems*, 2016.

Shen Yuan, Haotian Liu, and Hongteng Xu. Bridging the gap between low-rank and orthogonal adaptation via householder reflection adaptation. In *Advances in Neural Information Processing Systems 37*, pp. 113484–113518. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2024. doi: 10.52202/079017-3606. URL https://doi.org/10.52202/079017-3606.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=lq62uWRJjiY.

A Rank, Subspace, and Spectral Properties

Proposition 2 (Exact rank and subspace invariants)**.** *For any $W_0 \in \mathbb{R}^{m\times n}$ and square generators $\xi_L \in \mathbb{R}^{m\times m}$ and $\xi_R \in \mathbb{R}^{n\times n}$,*

$$\operatorname{rank}(\exp(\xi_L)W_0) = \operatorname{rank}(W_0) = \operatorname{rank}(W_0\exp(\xi_R)), \tag{15}$$

$$\ker(\exp(\xi_L)W_0) = \ker(W_0), \qquad \operatorname{im}(W_0\exp(\xi_R)) = \operatorname{im}(W_0). \tag{16}$$

Proof. The matrix exponential is always invertible, with inverse $\exp(-\xi)$. Left or right multiplication by an invertible matrix therefore leaves rank unchanged. Moreover, $\exp(\xi_L)W_0x = 0$ if and only if $W_0x = 0$, which proves the kernel identity. Because $\exp(\xi_R)$ is surjective, $\{W_0\exp(\xi_R)x : x \in \mathbb{R}^n\} = \{W_0z : z \in \mathbb{R}^n\}$, which proves the image identity. □

The guarantee applies to every generator parameterization used in the paper and to ordered composition: if $G_1, \ldots, G_T$ are invertible, then $G_T \cdots G_1W_0$ has the same rank as W_0. This is a hard algebraic invariant rather than a penalty or an approximation.

Proposition 3 (Orthogonal rigidity versus general-linear flexibility)**.** *An orthogonal left or right action preserves every singular value of W_0. A general-linear exponential action preserves rank but need not preserve singular values.*

Proof. For orthogonal Q_L and Q_R, $(Q_LW_0)^\top(Q_LW_0) = W_0^\top W_0$ and $(W_0Q_R)(W_0Q_R)^\top = W_0W_0^\top$. Hence the singular values are unchanged. For a general-linear counterexample, take $W_0 = I_2$ and $\xi = \operatorname{diag}(t, 0)$. Then $\exp(\xi)W_0 = \operatorname{diag}(e^t, 1)$ has rank two for every finite t, while its singular values vary with t. □

B Extension to Affine Transformations: Handling Bias Terms via the Affine Group $\mathrm{Aff}(d, \mathbb{R})$

The main experiments use bias-free Transformer projections, but the same action can be extended to a linear layer with bias, $y = Wx + b$.

The invertible affine transformations form the affine group $\mathrm{Aff}(d, \mathbb{R})$, represented as a subgroup of $GL(d + 1, \mathbb{R})$ in homogeneous coordinates.

Proposition 4 (Affine exponential action)**.** *Let the pre-trained weights and bias be represented as an augmented matrix $\tilde{W}_0 \in \mathbb{R}^{(d_{out}+1)\times(d_{in}+1)}$, and let the Lie algebra generator $\tilde{\xi} \in \mathfrak{aff}(d_{out})$ be partitioned into a weight generator $\xi_W \in \mathbb{R}^{d_{out}\times d_{out}}$ and a bias translation generator $\xi_b \in \mathbb{R}^{d_{out}\times 1}$:*

$$\tilde{W}_0 = \begin{bmatrix} W_0 & b_0 \\ \mathbf{0}^\top & 1 \end{bmatrix}, \quad \tilde{\xi} = \begin{bmatrix} \xi_W & \xi_b \\ \mathbf{0}^\top & 0 \end{bmatrix} \tag{17}$$

The exponential action yields the exact updated weights and bias:

$$\tilde{W}_{new} = \exp(\tilde{\xi})\tilde{W}_0 = \begin{bmatrix} \exp(\xi_W)W_0 & \exp(\xi_W)b_0 + \phi_1(\xi_W)\xi_b \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{18}$$

where ϕ_1 is the entire matrix function in Eq. 4.

Proof. To compute the exponential map of the block-triangular generator $\tilde{\xi}$, we rely on the algebraic properties of the Taylor series expansion $\exp(\tilde{\xi}) = \sum_{k=0}^{\infty} \frac{\tilde{\xi}^k}{k!}$.

Observing the powers of $\tilde{\xi}$:

$$\tilde{\xi}^0 = \begin{bmatrix} I & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \quad \tilde{\xi}^1 = \begin{bmatrix} \xi_W & \xi_b \\ \mathbf{0}^\top & 0 \end{bmatrix}, \quad \tilde{\xi}^k = \begin{bmatrix} \xi_W^k & \xi_W^{k-1}\xi_b \\ \mathbf{0}^\top & 0 \end{bmatrix} \text{ for } k \geq 1. \tag{19}$$

Summing these terms block-wise directly yields:

$$\exp(\tilde{\xi}) = \begin{bmatrix} \sum_{k=0}^{\infty} \frac{\xi_W^k}{k!} & \left(\sum_{k=1}^{\infty} \frac{\xi_W^{k-1}}{k!}\right)\xi_b \\ \mathbf{0}^\top & 1 \end{bmatrix} = \begin{bmatrix} \exp(\xi_W) & \phi_1(\xi_W)\xi_b \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{20}$$

Applying this transformation to the pre-trained augmented weights $\tilde{W}_0$:

$$\tilde{W}_{new} = \begin{bmatrix} \exp(\xi_W) & \phi_1(\xi_W)\xi_b \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} W_0 & b_0 \\ \mathbf{0}^\top & 1 \end{bmatrix} = \begin{bmatrix} \exp(\xi_W)W_0 & \exp(\xi_W)b_0 + \phi_1(\xi_W)\xi_b \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{21}$$

□

Action interpretation. Equation 18 shows that bias adaptation can be handled consistently within the Lie-Tuning framework. The adapted bias $b_{new} = \exp(\xi_W)b_0 + \phi_1(\xi_W)\xi_b$ consists of two coupled dynamics:

1. $\exp(\xi_W)b_0$ transforms the pretrained bias by the same output-space action applied to W_0.
2. $\phi_1(\xi_W)\xi_b$ integrates the translation generator along that action.

Thus homogeneous augmentation extends the framework to bias terms without changing its exponential form.

C Matrix `expm1`: Cancellation-Free Residuals Near the Identity

C.1 From Scalar `expm1` to Matrix Exponential Residuals

The scalar function `expm1` computes $e^z - 1$ directly rather than evaluating e^z and subsequently subtracting one. The distinction matters near $z = 0$: rounding e^z at its $O(1)$ scale before subtraction can erase significant digits of the $O(|z|)$ remainder, even when the rounded exponential is accurate in absolute terms. The matrix residual required by Lie-Tuning has exactly the same structure,

$$\Delta(M) = \exp(M) - I. \tag{22}$$

Its matrix-`expm1` form follows from the entire function ϕ_1:

$$\Delta(M) = M\phi_1(M) = \phi_1(M)M = M + \frac{M^2}{2!} + \frac{M^3}{3!} + \cdots. \tag{23}$$

The two products agree because $\phi_1(M)$ is a power series in M. Equation 23 is not a truncation: it represents every order of the exponential remainder and is defined at $M = 0$ and for singular M.

C.2 Structured Residuals and Action Conventions

The left and right actions become

$$y_{\text{left}} = W_0 x + \Delta(\xi_L) W_0 x, \tag{24}$$

$$y_{\text{right}} = W_0 x + W_0 \Delta(\xi_R) x. \tag{25}$$

Each structured generator admits a subtraction-free residual without forming an ambient $d \times d$ transformation:

$$\Delta(BA)h = B\phi_1(AB)Ah, \tag{26}$$

$$\Delta(UV^\top)h = U\phi_1(V^\top U)V^\top h, \tag{27}$$

$$\Delta(M_i)h^{(i)} = \phi_1(M_i)M_i h^{(i)}. \tag{28}$$

The three lines give MExRA, O-MExRA, and one block of MExBA or O-MExBA, respectively.

The mathematical action side is independent of tensor layout: a left action may appear as right multiplication by transposed factors when batches are stored as row vectors. We use "left" and "right" exclusively in the column-vector convention of Eq. 2.

C.3 Subtractive Cancellation Near the Identity

The numerical distinction can be made explicit. Suppose an exponential routine first returns

$$\widehat{X} = \exp(M) + E_{\exp}, \tag{29}$$

and the subsequent floating-point subtraction produces

$$\widehat{\Delta}_{\text{sub}} = \text{fl}(\widehat{X} - I) = \Delta(M) + E_{\exp} + R_{\text{sub}}, \tag{30}$$

where $E_{\exp}$ is the forward error inherited from the exponential and R_{sub} is the rounding error of the final subtraction. Consequently,

$$\frac{\|\widehat{\Delta}_{\text{sub}} - \Delta(M)\|}{\|\Delta(M)\|} \le \frac{\|E_{\exp}\| + \|R_{\text{sub}}\|}{\|\Delta(M)\|}. \tag{31}$$

For $0 < \|M\| \ll 1$, Eq. 23 gives $\|\Delta(M) - M\| = O(\|M\|^2)$. Under the local error models $\|E_{\exp}\| \lesssim c_{\exp} u \|\exp(M)\|$ and $\|R_{\text{sub}}\| \lesssim c_{\text{sub}} u \|\widehat{X} - I\|$, the worst-case contribution inherited from forming the $O(1)$ exponential before extracting its small remainder is

$$\frac{\|\widehat{\Delta}_{\text{sub}} - \Delta(M)\|}{\|\Delta(M)\|} \lesssim \frac{c_{\exp} u}{\|M\|} + c_{\text{sub}} u, \tag{32}$$

up to higher-order terms. Here u denotes the largest unit roundoff among the arithmetic stages under discussion, and the constants absorb fixed dimension- and routine-dependent factors. Thus an exponential that is accurate at the scale of $\exp(M)$ need not resolve the much smaller quantity $\exp(M) - I$.

Direct ϕ_1 computation removes this specific $u/\|M\|$ amplification. Let $\widehat{\phi}_1(M) = \phi_1(M) + F_\phi$ and form the product in floating point:

$$\widehat{\Delta}_\phi = \text{fl}\Big(M\widehat{\phi}_1(M)\Big) = \Delta(M) + MF_\phi + R_{\text{mult}}. \tag{33}$$

If $\|F_\phi\| \lesssim c_\phi u \|\phi_1(M)\|$ and the matrix multiplication is normwise stable, then $\phi_1(M) = I + O(\|M\|)$ and $\|\Delta(M)\| \sim \|M\|$ imply the local bound

$$\frac{\|\widehat{\Delta}_\phi - \Delta(M)\|}{\|\Delta(M)\|} \lesssim (c_\phi + c_{\text{mult}})u, \tag{34}$$

excluding underflow and again absorbing fixed conditioning and dimension factors into the constants. The ϕ_1 route therefore avoids the near-identity error amplification caused by explicitly forming $\exp(M) - I$. This is a local comparison of the residual constructions, not a global stability claim for the factorized adapter, whose outer products and dtype conversions introduce their own ordinary rounding.

C.4 Inversion-Free Extraction of ϕ_1

The quotient $\phi_1(M) = (\exp(M) - I)M^{-1}$ is not a suitable implementation: it recreates the subtraction above, requires a solve with M, and is undefined when M is singular. Instead, we use the block-triangular identity in Eq. 14, a classical matrix-function construction (Van Loan, 1978).

Proposition 5 (Block Matrix Identity for ϕ_1)**.** *For $M \in \mathbb{R}^{k\times k}$, define*

$$Z = \begin{bmatrix} M & I_k \\ 0 & 0 \end{bmatrix}. \tag{35}$$

Then the upper-right block of $\exp(Z)$ is exactly $\phi_1(M)$, as stated in Eq. 14.

Proof. For every integer $j \geq 1$,

$$Z^j = \begin{bmatrix} M^j & M^{j-1} \\ 0 & 0 \end{bmatrix}. \tag{36}$$

Substituting these powers into the exponential series, the upper-right block becomes

$$[\exp(Z)]_{1,2} = \sum_{j=1}^{\infty} \frac{M^{j-1}}{j!} = I_k + \frac{M}{2!} + \frac{M^2}{3!} + \cdots = \phi_1(M). \tag{37}$$

The extraction requires no inverse or linear solve with M and never explicitly forms $\exp(M) - I$; operations internal to the chosen exponential routine are unchanged. □

Our implementation constructs Z only for the reduced kernel, evaluates its block exponential in FP32 on accelerators (FP64 on CPU), extracts the upper-right block, casts it back to the model dtype, and applies the factored residual in Eq. 28. The ambient $d \times d$ exponential and identity are never formed.

C.5 Why This Matters for Identity-Initialized Training

The cancellation regime is not an edge case for Lie-Tuning; it is the point from which optimization starts. MExRA initializes $B = 0$, hence $BA = 0$ and $\exp(BA) = I$. The factored residual gives this identity exactly while retaining the first-order adaptation direction:

$$D_B[B\phi_1(AB)A]|_{B=0}[\dot{B}] = \dot{B}A, \tag{38}$$

because $\phi_1(0) = I$. Consequently, MExRA begins with zero functional perturbation but a well-defined tangent update, while all higher-order terms enter continuously through ϕ_1 as training moves away from the identity.

For O-MExRA, the ambient generator $\xi(B) = BA^\top - AB^\top$ is also zero at $B = 0$, with first variation

$$D_B\big[\Delta(BA^\top - AB^\top)\big]\big|_{B=0}[\dot{B}] = \dot{B}A^\top - A\dot{B}^\top. \tag{39}$$

For a directly parameterized block, $D_M\Delta(M)|_{M=0}[\dot{M}] = \dot{M}$; under a skew parameterization $M = P - P^\top$, this becomes $\dot{P} - \dot{P}^\top$. Thus every variant starts at the identity while exposing its intended tangent-space update.

These derivatives show that all variants begin with zero functional perturbation while retaining their intended first variation. The matrix-`expm1` construction represents that tangent-space residual without recovering it from two nearly equal matrices. The exact small-kernel reduction, by itself, does not prevent cancellation when the residual is formed by subtracting I from a rounded exponential. The final addition of the residual to the base activation remains subject to ordinary floating-point rounding; the contribution here is to remove the avoidable cancellation internal to constructing the residual itself.

D Analysis of Numerical Instability in O-MExRA

O-MExRA uses the skew-symmetric generator $\xi = BA^\top - AB^\top = UV^\top$, where $U = [B, -A]$ and $V = [A, B]$. Its reduced kernel is

$$K = V^\top U = \begin{bmatrix} A^\top B & -A^\top A \\ B^\top B & -B^\top A \end{bmatrix}. \tag{40}$$

At the identity initialization $B = 0$,

$$K_0 = \begin{bmatrix} 0 & -A^\top A \\ 0 & 0 \end{bmatrix}, \qquad K_0^2 = 0. \tag{41}$$

Consequently, every eigenvalue of K_0 is zero and its spectral radius is exactly zero, irrespective of the magnitude of $A^\top A$. A large $A^\top A$ therefore cannot be interpreted as spectral-radius explosion at initialization. It instead makes K_0 large in norm and strongly non-normal: $\exp(K_0) = I + K_0$ is exact, but reduced-kernel intermediates and their derivatives can still have unfavorable scaling in finite precision.

Once B moves away from zero, the nonzero eigenvalues shared by K and the skew-symmetric ambient generator are purely imaginary, and the ambient action $\exp(\xi)$ remains orthogonal. This rules out analytic norm explosion of the transformation itself. It does not rule out poor conditioning in a redundant low-rank factorization, large transient norms in a non-normal reduced kernel, or implementation-level overflow under aggressive scaling. Our use of orthonormal-column initialization for A, FP32 kernel evaluation, and generator scaling is motivated by controlling these quantities. This corrected analysis explains the observed sensitivity without attributing it to a nonexistent nonzero spectral radius at $B = 0$.

E Relation to Orthogonal Fine-Tuning and the Cayley Transform

Invariant boundary. Orthogonal multiplication is a valuable conservative prior: it preserves rank, vector norms, and the full singular spectrum. Lie-Tuning does not present rank preservation as unique to our method. Its general-linear variant deliberately relaxes only the isometry requirement, retaining invertibility and rank while allowing the spectrum to move. The orthogonal variants in Section 3.4 provide a controlled comparison under the same exponential parameterization.

Cayley versus exponential coordinates. For skew-symmetric Q, the Cayley transform $(I - Q)^{-1}(I + Q)$ is orthogonal and is evaluated by a linear solve; it omits orthogonal matrices with eigenvalue -1 from its chart. The exponential map supplies a different set of coordinates on $SO(d)$ and extends directly to non-skew generators in GL^+. Neither coordinate system is universally superior. Our reason for using the exponential is that the low-rank identity in Proposition 1 makes the exact general-linear action tractable and uses the same machinery for its orthogonal ablation.

Expressivity and cost. MExRA pays $O(dr^2 + r^3)$ to construct an exact global low-rank exponential kernel instead of an ambient $O(d^3)$ exponential. MExBA and orthogonal block methods impose locality for a different cost–expressivity trade-off. Because GL^+ is non-compact, its extra spectral freedom can also amplify activations; generator scaling, initialization, and mixed-precision kernel evaluation are therefore practical safeguards rather than consequences of the rank theorem.

F Theoretical Foundations of Initialization and Scaling in Lie-Tuning

This section records the scaling and initialization rules used in our implementation and the conditioning intuition behind them. These choices affect optimization but not the exact rank guarantee.

F.1 Scaling Low-Rank Generators with $\alpha/\sqrt{r}$

Standard LoRA commonly scales its additive update by α/r. For the exponential generator, we use $\alpha/\sqrt{r}$ to normalize the scale arising from a rank-r inner product.

Low-Rank Parameterization (MExRA). For either action side, let $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times d}$. Under the simplified model that their entries are independent, centered, and have variances σ_A^2 and σ_B^2, an entry of the unscaled product satisfies

$$\mathrm{Var}(\xi_{ij}) = \mathrm{Var}\left(\sum_{k=1}^{r} B_{ik}A_{kj}\right) = \sum_{k=1}^{r} \mathrm{Var}(B_{ik})\mathrm{Var}(A_{kj}) = r \cdot \sigma_B^2\sigma_A^2 \tag{42}$$

Thus its standard deviation scales as $\sqrt{r}$ in this model. We use

$$s_r = \frac{\alpha}{\sqrt{r}} \tag{43}$$

so that α controls the typical generator scale across rank choices. In the implementation, the reduced kernel is multiplied by s_r. When its Frobenius norm exceeds the numerical threshold κ, the kernel and the corresponding outer residual are additionally multiplied by the same clamp factor. This safeguard changes only the effective factor scaling at that forward pass and does not affect the low-rank identity in Proposition 1.

Block-Diagonal Parameterization (MExBA). For block-diagonal variants, the generator is learned directly rather than through a rank-r product. We therefore use no rank-dependent factor: the implementation uses unit scaling.

F.2 Initialization Dynamics: Kaiming vs. Orthogonal

To start exactly from the identity action, we initialize $B = 0$ for low-rank variants and $M_i = 0$ for block variants. The basis A determines the initial gradient subspace. Our implementation uses the following action- and group-dependent choices.

1. Left-Multiplied MExRA ($GL(d)$). Near initialization, $y \approx W_0 x + s_r B(AW_0 x)$. We initialize A with orthonormal rows, $AA^\top = I_r$, so this initial projection has operator norm one and does not itself amplify the base output.

2. Right-Multiplied MExRA ($GL(d)$). Here $y \approx W_0 x + s_r W_0 B(Ax)$, and A acts directly on the normalized layer input. We use Kaiming initialization to provide diverse initial input projections; this difference from the left variant is an empirical design choice motivated by the location of the action.

3. O-MExRA ($SO(d)$). The reduced kernel contains the block $-A^\top A$ (Eq. 40). At $B = 0$ the kernel is nilpotent with spectral radius zero, but its norm and non-normality still scale with $\|A^\top A\|$. Column-orthonormal initialization, $A^\top A = I_r$, makes this block dimension-independent and improves the scale of the reduced computation; Appendix D gives the corrected analysis.

Table 7: **Scaling and initialization used for Lie-Tuning variants.** These choices control generator and reduced-kernel scale; they are not required for rank preservation.

Variant	Group	Position	Scaling Factor	Trainable Init	Rationale
Left-MExRA	$GL(d)$	Post (W_0x)	$\alpha/\sqrt{r}$	$B=0$; A orthogonal	Unit-norm projection of the base output.
Right-MExRA	$GL(d)$	Pre (x)	$\alpha/\sqrt{r}$	$B=0$; A Kaiming	Diverse projections of normalized inputs.
Left-O-MExRA	$SO(d)$	Post (W_0x)	$\alpha/\sqrt{r}$	$B=0$; A orthogonal	Dimension-independent $A^\top A$ kernel block.
Right-O-MExRA	$SO(d)$	Pre (x)	$\alpha/\sqrt{r}$	$B=0$; A orthogonal	Dimension-independent $A^\top A$ kernel block.
MExBA	$GL(d)$	Post/Pre	1.0 (None)	Blocks zero	Exact identity action at initialization.
O-MExBA	$SO(d)$	Post/Pre	1.0 (None)	Blocks zero	Exact identity action at initialization.

F.3 Initialization-Preserving Asymmetric Weight Decay

For factorized MExRA and O-MExRA adapters, including their right-multiplied variants, we use asymmetric AdamW parameter groups: the reported coefficient λ is applied only to the zero-initialized factor B, while A receives no weight decay. This choice protects the deliberately structured nonzero factor after identity initialization. Because $B_0 = \mathbf{0}$, both $\xi = BA$ and $\xi = BA^\top - AB^\top$ satisfy $\nabla_A \mathcal{L}(A_0, B_0) = \mathbf{0}$, whereas $\nabla_B \mathcal{L}(A_0, B_0)$ is generally nonzero. Symmetric AdamW would nevertheless produce $A_1 = (1 - \eta\lambda)A_0$ before A receives any task gradient; for an orthonormal A_0, its Gram matrix would become $(1 - \eta\lambda)^2 I_r$ rather than I_r, while a Kaiming-initialized A_0 would similarly lose its prescribed scale. Decaying only B therefore leaves the initialized basis structure and scale intact during the zero-generator start-up and regularizes the factor that activates the first-order departure from identity. This prevents direct decay-induced contraction of A; subsequent task gradients may still update it.

G Sequential Multiplicative Composition

When task adapters are merged sequentially, additive and multiplicative parameterizations obey different composition laws. Additive adaptation gives

$$W_T = W_0 + \sum_{t=1}^{T} \Delta W_t, \tag{44}$$

which imposes no invariant on the rank of the merged map. Sequential left actions instead give

$$W_T = \exp(\xi_T) \cdots \exp(\xi_1) W_0. \tag{45}$$

Every partial product is invertible, so Eq. 45 preserves $\mathrm{rank}(W_0)$ exactly at every stage. Right actions analogously compose on the other side in their corresponding order.

The product is order-sensitive. In a neighborhood where a consistent matrix logarithm exists and the Baker–Campbell–Hausdorff (BCH) series converges, two steps can be written as $\exp(Z) = \exp(\xi_2)\exp(\xi_1)$ with

$$Z = \xi_2 + \xi_1 + \frac{1}{2}[\xi_2, \xi_1] + \frac{1}{12}[\xi_2, [\xi_2, \xi_1]] - \frac{1}{12}[\xi_1, [\xi_2, \xi_1]] + \cdots, \tag{46}$$

The commutator terms describe interactions between successive generators; BCH is a local representation of the product, not a global claim that every product has one real logarithm.

Both orthogonal and general-linear products preserve rank. The former also freeze singular values at every stage, whereas the latter may reweight and shear feature directions. That additional flexibility is a plausible mechanism for the retained-accuracy pattern in Table 5, but the algebraic composition law alone does not prove reduced forgetting. Our experiment provides evidence for one task order, as stated in Section 4.5.

H Directional Actions: Left versus Right Multiplication

The two action sides arise because W_0 maps between distinct vector spaces: an operator on its codomain post-composes W_0, while an operator on its domain pre-composes it. Non-commutativity makes their composition order-sensitive, but the domain–codomain distinction exists even before choosing a particular generator.

H.1 Algebraic Semantics: Output Space vs. Input Space

Consider a pre-trained linear layer $W_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ mapping an input feature $\mathbf{x} \in \mathbb{R}^{d_{\text{in}}}$ to an output $\mathbf{y} \in \mathbb{R}^{d_{\text{out}}}$.

Left action: post-composition on the codomain. For $G_L \in GL^+(d_{\text{out}})$,

$$y_\theta = G_L(W_0 x). \tag{47}$$

It transforms the output representation after the pretrained map. Because G_L is invertible, it leaves the null space in the input domain unchanged: $\ker(G_L W_0) = \ker(W_0)$. Its image is $G_L \,\mathrm{im}(W_0)$, so the output subspace may move while retaining its dimension.

Right action: pre-composition on the domain. For $G_R \in GL^+(d_{\text{in}})$,

$$y_\theta = W_0(G_R x). \tag{48}$$

It changes the input coordinates presented to the pretrained map. Because G_R is surjective, it leaves the output image unchanged: $\mathrm{im}(W_0 G_R) = \mathrm{im}(W_0)$. Its kernel becomes $G_R^{-1} \ker(W_0)$, so the discarded input subspace may move while retaining its dimension.

H.2 Task Hypotheses

The algebra does not determine which side is best for a task. It motivates two testable biases. An output-side action can directly recombine, scale, and shear features already produced by W_0, making it a natural candidate when the desired change is expressed in the layer output. An input-side action can align or reparameterize the representation before the frozen map acts, making it a natural candidate under input-distribution or language shifts. We use these hypotheses to organize the code-generation and cross-lingual experiments; the results should not be read as a universal prescription for all layers or tasks.

H.3 Computational and Architectural Asymmetry

For a rectangular projection, the action side also selects the ambient generator dimension. A Transformer MLP up-projection with $d_{\text{in}} = 4096$ and $d_{\text{out}} = 11008$, for example, uses an 11008-dimensional left generator or a 4096-dimensional right generator. MExBA therefore has $d_{\text{out}}b$ versus $d_{\text{in}}b$ parameters on the two sides. MExRA avoids cubic dependence on either ambient dimension because its matrix function is only $r \times r$, but it does not erase the difference: its parameters and residual projections still scale linearly as $2d_{\text{out}}r$ versus $2d_{\text{in}}r$.

H.4 Empirical Profiling vs. Theoretical Complexity

While Section 3 shows that Lie-Tuning keeps $O(dr)$ parameter scaling and residual application cost, empirical profiling reveals a notable discrepancy in hardware-level throughput. We benchmarked the training speed across different adaptation methods on the same hardware setup, measuring the throughput in samples per second.

Table 8: **Empirical Training Throughput.** Measured in samples per second during the fine-tuning of LLaMA-2-7B. The performance gaps stem from both the intrinsic algorithmic cost of multiplicative adaptation and current implementation bottlenecks.

Method	**Throughput (Samples / sec)** $\uparrow$
LoRA	16.62
OFT	8.83
O-MExBA	5.99
MExBA	5.81
O-MExRA	5.61
MExRA	5.37

As shown in Table 8, standard LoRA significantly outperforms the others in wall-clock time. We explicitly acknowledge that this performance gap is not entirely an engineering artifact; it stems from both the **intrinsic algorithmic overhead** of multiplicative adaptation and the **system-level limitations** of current unoptimized implementations.

1. Intrinsic Algorithmic Overhead (Multiplicative vs. Additive). Lie-Tuning uses a multiplicative exponential action, whereas LoRA uses an additive low-rank branch. For a left action, MExRA first obtains $h_{base} = W_0 x$ and then evaluates $\Delta h = h_{base} A^\top \Phi^\top B^\top$ in row-vector notation. The additional $r \times r$ kernel projection produces a deeper dependency chain and more operations per token than the additive baseline, even though both methods scale linearly in d for fixed r.

2. Engineering and System Bottlenecks. Beyond the algorithmic cost, our implementation invokes a general-purpose matrix-exponential routine on many compact $r \times r$ or $2r \times 2r$ kernels and upcasts the kernel computation from `bfloat16` to `float32`. Kernel-launch, batching, and casting overheads can therefore dominate these small operations. The profiling results conflate these implementation effects with MExRA's intrinsic extra projections.

Lie-Tuning therefore trades some current wall-clock speed for an invertible, spectrally flexible action. Fused kernels may reduce engineering overhead, but the additional matrix-function and projection operations remain part of the method.