TRANSFORMERS CAN IMPLEMENT PRECONDITIONED RICHARDSON ITERATION FOR IN-CONTEXT GAUSSIAN KERNEL REGRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Mechanistic studies of in-context learning (ICL) have connected transformer computations to iterative optimization algorithms for linear regression and related tasks, often using linear or ReLU attention variants. For nonlinear ICL, prior work has related softmax and kernelized attention to functional gradient descent, but it remains unclear whether a standard transformer with softmax attention can implement a convergent solver with an end-to-end prediction-error guarantee. In this paper, we study in-context kernel ridge regression (KRR) with Gaussian kernels and show, both theoretically and empirically, that a standard softmax-attention transformer can approximate the KRR predictor during its forward pass. Under bounded-data assumptions, we construct a single-head transformer whose forward pass approximately implements *preconditioned Richardson iteration* on the associated kernel system. The construction uses $\mathcal{O}(\log(1/\epsilon))$ blocks and MLP width $\mathcal{O}(\sqrt{N/\epsilon})$ to achieve ϵ-accurate prediction for prompts of length N. Our construction reveals a functional decomposition within the transformer architecture: softmax attention produces a row-normalized Gaussian-kernel operator needed for *cross-token* interactions, while MLP layers act locally to approximate the *intra-token* scalar arithmetic required by the update. Empirically, we train GPT-2-style transformers on Gaussian-process regression tasks and observe that they progressively align with the exact Gaussian KRR estimator across depth in terms of both *prediction error* and *induced weights*, with ablations further supporting this trend. Comparisons with classical KRR solvers also show that deeper layers align with later solver iterates. Together, we empirically demonstrate that the pretrained transformers exhibit progressive refinement toward exact Gaussian KRR across depth, and theoretically establish inexact preconditioned Richardson iteration as a concrete mechanism for approximating this predictor within an explicitly constructed softmax-attention transformer.

1 INTRODUCTION

In-context learning (ICL) refers to the phenomenon in which a pretrained transformer, given a sequence of input-output examples followed by a query input, produces a prediction for that query in a single forward pass, *without parameter updates* (Brown et al., 2020; Garg et al., 2022). This has emerged as a central question in the theoretical study of ICL: does the forward pass itself implement a learning algorithm on the examples in the prompt, and if so, what algorithm is being run and how is it realized by the architecture?

For linear regression and related linear tasks, prior work has made this algorithmic viewpoint precise: transformers can implement classical optimization algorithms, including gradient descent, preconditioned gradient descent, and Newton-type iterations (von Oswald et al., 2023; Akyürek et al., 2023; Ahn et al., 2023; Bai et al., 2023; Mahankali et al., 2024; Vladymyrov et al., 2024; Fu et al., 2024; Zhang et al., 2025). Transformer layers can be understood as steps of an iterative optimizer.

For nonlinear ICL, the corresponding mechanism is less understood. Kernel ridge regression (KRR) is a natural test scenario: nonlinear in the input but mathematically explicit, and on Gaussian process regression tasks its solution coincides with the Bayes posterior mean. Recent work analyzes atten-

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.

tion as a form of kernel or functional gradient descent in a reproducing kernel Hilbert space (RKHS) (Cheng et al., 2024; Han et al., 2025; Dragutinović et al., 2025; Sander & Peyré, 2024). Concrete convergence guarantees in this line, however, require attention to be *kernelized* so that each layer realizes a standard functional gradient step (Cheng et al., 2024). Standard softmax attention corresponds instead to kernel gradient descent with a context-adaptive learning rate (Cheng et al., 2024; Sander & Peyré, 2024). However, it remains unclear, empirically, to what extent pretrained GPT-style transformers align with exact KRR predictor across depth and, theoretically, whether one can explicitly construct a transformer with standard softmax attention and ReLU MLPs that realizes an iterative KRR solver with end-to-end prediction-error control.

We study both questions in the setting of in-context Gaussian KRR. Figure 1 presents our central empirical observation: GPT-style transformers trained on Gaussian-process regression tasks progressively align with exact Gaussian KRR across depth, in terms of both prediction error and output-induced weights. These output-level observations motivate our constructive analysis.

To construct a transformer with end-to-end prediction-error control, we move from the primal RKHS formulation to the dual kernel linear system: given training samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, the KRR predictor admits dual coefficients $\boldsymbol{w} \in \mathbb{R}^N$ that satisfy

$$(\boldsymbol{K} + \lambda \boldsymbol{I})\boldsymbol{w} = \boldsymbol{y}, \qquad \boldsymbol{K}_{ij} = \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j).$$

The mechanism question then shifts from *"does the transformer descend a primal RKHS loss?"* to *"does the transformer solve the dual kernel linear system?"*. For the Gaussian kernel $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\|\boldsymbol{x} - \boldsymbol{x}'\|^2/(2v^2))$, suppose the raw attention score between tokens i and j is $\boldsymbol{L}_{ij} = -\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/(2v^2)$. After softmax normalization,

$$\mathrm{softmax}(\boldsymbol{L})_{ij} = \frac{\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\sum_{j'=1}^N \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_{j'})} = (\boldsymbol{D}^{-1}\boldsymbol{K})_{ij}, \qquad \boldsymbol{D} = \mathrm{diag}(\boldsymbol{K}\boldsymbol{1}).$$

Softmax thus returns the kernel matrix itself with each row rescaled to sum to one, i.e., the row-sum preconditioner of the dual kernel system. Aggregating value vectors with these weights implements the cross-token operation in a row-sum–preconditioned Richardson step; the remaining tokenwise operations (rescaling by diagonal quantities, adding the data term, updating the current iterate) are within the reach of an MLP layer. Section 4 makes this construction precise.

We summarize our contributions as follows.

- **A realization of in-context Gaussian KRR with softmax-attention and ReLU-MLP layers** (Theorem 1; Appendix C). We construct a transformer that approximates the exact Gaussian KRR predictor without modifying the standard block architecture: softmax attention is followed by a single-hidden-layer ReLU MLP, with residual connections around both sublayers. Its forward pass approximately implements Richardson iteration with row-sum preconditioning, with softmax-attention performing the row-normalized cross-token kernel operations and ReLU-MLPs approximating the required tokenwise arithmetic. The construction neither replaces softmax with another attention operator nor directly exposes its normalizer.
- **End-to-end convergence with explicit depth–width scaling** (Theorem 1). We establish an end-to-end prediction-error guarantee by combining a convergence analysis of *inexact* preconditioned Richardson iteration with explicit ReLU approximation rates for the required scalar operations. Under bounded-data assumptions, the constructed transformer achieves prediction error at most ε using depth $\mathcal{O}(\log(1/\varepsilon))$ and MLP width $\mathcal{O}(\sqrt{N/\varepsilon})$ on prompts of length N. The depth scaling follows from the convergence rate of preconditioned Richardson, while the width scaling reflects the cost of approximating the tokenwise scalar arithmetic.
- **Empirical evidence of KRR refinement** (Section 5). We pretrain GPT-2-style transformers on Gaussian-process regression tasks for three input distributions and evaluate them across multiple context lengths. Using layer-wise linear probes, we observe progressive alignment with exact Gaussian KRR across depth in terms of both prediction error and induced-weight similarity. The probing ablations indicate that the observed trend is due to the trained transformer rather than the probes; the kernel-mismatch ablation shows that the transformer aligns more closely with KRR based on the data-generating kernel; and the noise-mismatch ablation supports the interpretation of the trained transformer as a finite-iterate KRR solver with a fixed regularization parameter.

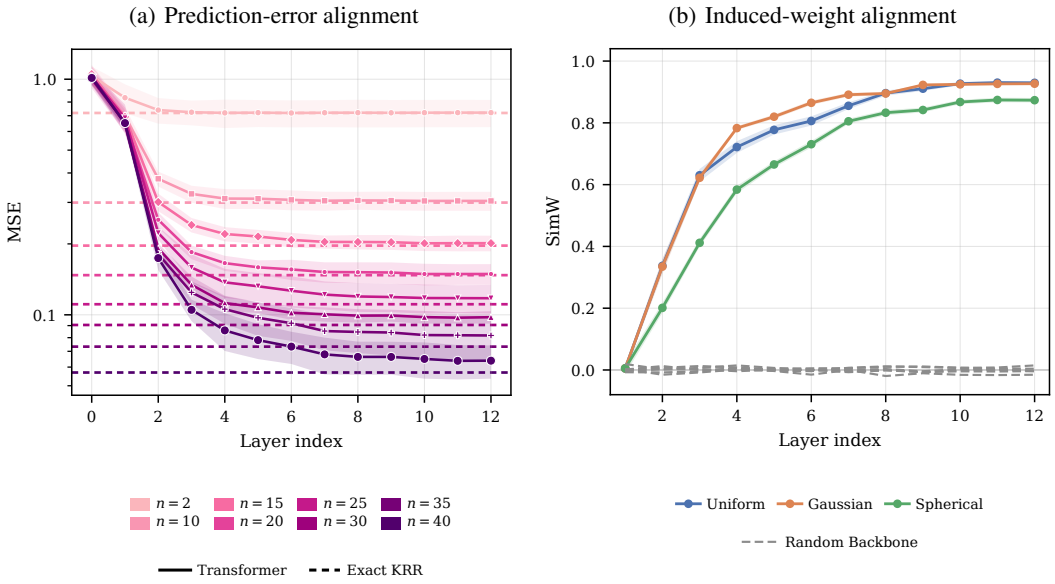


Figure 1: The transformer's in-context *prediction errors* and *induced weights* progressively align with those of exact KRR as depth increases. **(a)** Layer-wise mean-squared error (MSE) on spherical inputs; dashed horizontal lines are the corresponding exact-KRR MSEs, coloured to match their context length. *For each context length n, the transformer MSE progressively approaches the exact-KRR reference as depth increases.* Similar trends hold for Uniform and Gaussian inputs (Figure 6). **(b)** Cosine similarity (SimW) between the transformer's induced weights and the exact KRR weights at context length $n = 40$. *The induced weights become increasingly aligned with the exact KRR weights as the layer depth increases, across all three input distributions.* Grey dashed curves show random-initialized backbones, which remain near zero at every depth. In both panels, per-layer quantities are read out using linear probes trained with the backbone frozen. Results are computed over 5 evaluation seeds, each with 256 input sequences, with shaded bands indicating one standard deviation across seeds. Full details are in Section 5.

We compare the transformer with five classical KRR solvers and find that deeper layers generally best match later solver iterates. The late-layer similarities are closely clustered across solvers, consistent with their convergence to the same exact-KRR solution.

2 Related Work

Algorithmic interpretations of ICL. A major line of research frames the transformer's forward pass as the execution of a learning algorithm on the prompt. Garg et al. (2022) empirically showed that transformers can be trained to solve linear regression and decision-tree tasks in context. Subsequent theoretical work has focused predominantly on linear regression, identifying gradient descent (von Oswald et al., 2023; Akyürek et al., 2023), preconditioned gradient descent (Ahn et al., 2023; Mahankali et al., 2024; Vladymyrov et al., 2024), and Newton-type iterations (Fu et al., 2024), with parallel analyses of training dynamics (Zhang et al., 2024a; Chen et al., 2024; Lu et al., 2025; Zhang et al., 2025; Huang et al., 2024). Related works also study algorithm selection across linear tasks (Bai et al., 2023), nearest-neighbor classification (Li et al., 2024), discrete and Boolean function learning (Bhattamishra et al., 2024), and formal-language learning (Akyürek et al., 2024). Within this line of research, several studies examine how MLPs contribute to ICL in linear-attention models (Zhang et al., 2024b; Sun et al., 2025; Demir & Dogan, 2025). For noisy linear dynamical systems, Cole et al. (2026) construct deep linear transformers that implement modified Richardson iteration to approximate least-squares prediction with quantitative error guarantees.

Kernel and softmax views of nonlinear ICL. A complementary line connects ICL to kernel-based prediction. Cheng et al. (2024) show that kernel-matched attention corresponds to functional gradi-

ent descent in the RKHS. Goel & Bartlett (2024) realize Gaussian Nadaraya–Watson smoothing with softmax attention for Kalman filtering, while Collins et al. (2024) interpret pretraining single-head softmax attention as learning the bandwidth of such an estimator. Han et al. (2025) relate Bayesian inference on in-context prompts to normalized kernel regression. Shen et al. (2025) construct transformers for kernel smoothing on manifolds. Dragutinović et al. (2025) show that softmax attention implements context-adaptive kernel gradient descent for cross-entropy classification.

We relate softmax attention to preconditioned Richardson iteration for solving the dual Gaussian-KRR system. We note that independent and concurrent work by Kang et al. (2026) also connects attention normalization to this iteration for Gaussian-process posterior prediction. Their construction uses a modified transformer architecture that assumes both direct access to context-dependent attention denominators and exact division by them within each solver update. Under these architectural assumptions, their constructed layers realize each Richardson iteration *exactly*. In contrast, our construction retains standard softmax attention followed by ReLU MLPs throughout, using a dummy token to recover normalization information and MLPs to approximate the required arithmetic. We jointly analyze these approximation errors and how they propagate through the resulting *inexact* iterations. The experimental designs also fundamentally differ: their experiments evaluate posterior prediction in models pretrained with their proposed architecture. Our experiments investigate whether standard GPT-style transformers, trained from scratch without a prescribed solver, progressively approach exact KRR across depth. We assess this progression through layerwise diagnostics, ablations, and comparisons with several classical iterative solvers.

3 Preliminaries

3.1 Transformers

We consider transformer architectures that operate on an input sequence of N tokens, represented by a feature matrix $\boldsymbol{Z} = [\boldsymbol{z}_1, \dots, \boldsymbol{z}_N] \in \mathbb{R}^{D\times N}$ whose i-th column $\boldsymbol{z}_i \in \mathbb{R}^D$ is the embedding of the i-th token. We consider the following transformer architecture (Vaswani et al., 2017).

Definition 1 (Softmax attention layer)**.** A multi-head attention layer with H heads is denoted by $\mathrm{Attn}(\cdot)$. For each head $h \in [H]$, let $\boldsymbol{W}_Q^{(h)}, \boldsymbol{W}_K^{(h)}, \boldsymbol{W}_V^{(h)} \in \mathbb{R}^{D\times D}$ be the query, key, and value matrices, and let $\boldsymbol{M}^{(h)} \in \{0, -\infty\}^{N\times N}$ be a mask matrix. Given $\boldsymbol{Z} \in \mathbb{R}^{D\times N}$, the layer outputs $\widetilde{\boldsymbol{Z}} = \mathrm{Attn}(\boldsymbol{Z})$ via the residual update $\widetilde{\boldsymbol{Z}} := \boldsymbol{Z} + \sum_{h=1}^{H}(\boldsymbol{W}_V^{(h)}\boldsymbol{Z}) \cdot \sigma\big((\boldsymbol{W}_K^{(h)}\boldsymbol{Z})^\top(\boldsymbol{W}_Q^{(h)}\boldsymbol{Z}) + (\boldsymbol{M}^{(h)})^\top\big)$, where $\sigma(\cdot)$ is the column-wise softmax function. In vector form, for each $i \in [N]$, $\widetilde{\boldsymbol{z}}_i = \boldsymbol{z}_i + \sum_{h=1}^{H}\sum_{j=1}^{N} \frac{\exp\left(\left\langle \boldsymbol{W}_Q^{(h)}\boldsymbol{z}_i, \boldsymbol{W}_K^{(h)}\boldsymbol{z}_j\right\rangle + \boldsymbol{M}_{ij}^{(h)}\right)}{\sum_{j'=1}^{N}\exp\left(\left\langle \boldsymbol{W}_Q^{(h)}\boldsymbol{z}_i, \boldsymbol{W}_K^{(h)}\boldsymbol{z}_{j'}\right\rangle + \boldsymbol{M}_{ij'}^{(h)}\right)} \boldsymbol{W}_V^{(h)}\boldsymbol{z}_j$.

Definition 2 (MLP layer)**.** An MLP layer with hidden dimension D', denoted $\mathrm{MLP}(\cdot)$, has weight matrices $\boldsymbol{W}_{\mathrm{in}} \in \mathbb{R}^{D'\times D}, \boldsymbol{W}_{\mathrm{out}} \in \mathbb{R}^{D\times D'}$ and acts pointwise on each token: for $\boldsymbol{Z} \in \mathbb{R}^{D\times N}$, $\mathrm{MLP}(\boldsymbol{Z}) := \boldsymbol{Z} + \boldsymbol{W}_{\mathrm{out}}\,\sigma(\boldsymbol{W}_{\mathrm{in}}\boldsymbol{Z})$, where $\sigma(t) = \max\{t, 0\}$ is the entrywise ReLU activation.

Definition 3 (Transformer block and L-block transformer)**.** A transformer block (TFB) is the composition $\mathrm{TFB}(\boldsymbol{Z}) := \mathrm{MLP}(\mathrm{Attn}(\boldsymbol{Z}))$ of an attention layer followed by an MLP layer; we also allow an MLP-only block TFB_0 in which the attention layer acts as the identity (e.g., by setting all value matrices to zero), so $\mathrm{TFB}_0(\boldsymbol{Z}) = \mathrm{MLP}(\boldsymbol{Z})$. An L-block transformer TF is the sequential composition $\mathrm{TF}(\boldsymbol{Z}) := \big(\mathrm{TFB}^{(L)} \circ \mathrm{TFB}^{(L-1)} \circ \cdots \circ \mathrm{TFB}^{(1)}\big)(\boldsymbol{Z})$ of L blocks.

3.2 In-context Learning for Kernel Ridge Regression

Kernel ridge regression. Let $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ be a given training dataset. Here, $\{\boldsymbol{x}_i\}_{i=1}^N \subset \mathbb{R}^d$ represent the input vectors, $\{y_i\}_{i=1}^N \subset \mathbb{R}$ are the corresponding labels. We formulate the classical kernel ridge regression problem within a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ associated with a reproducing kernel $\mathcal{K}$. The objective is to minimize the regularized empirical risk

$$\min_{f\in\mathcal{H}} \left\{ \frac{1}{N}\sum_{i=1}^{N}(f(\boldsymbol{x}_i) - y_i)^2 + \lambda_0\|f\|_{\mathcal{H}}^2 \right\}, \tag{1}$$

where $\lambda_0 > 0$ is the regularization parameter. By the Representer Theorem (Schölkopf et al., 2001), the optimal solution admits the finite-dimensional form $f^*(\cdot) = \sum_{i=1}^N w_i\mathcal{K}(\boldsymbol{x}_i, \cdot)$. In the RKHS

setting considered here, optimizing over the dual coefficients $\boldsymbol{w} = [w_1, \ldots, w_N]^\top \in \mathbb{R}^N$ yields the explicit dual linear system

$$(\boldsymbol{K} + \lambda \boldsymbol{I})\boldsymbol{w} = \boldsymbol{y}, \tag{2}$$

with kernel matrix $\boldsymbol{K}_{ij} = \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)$, label vector $\boldsymbol{y} = [y_1, \ldots, y_N]^\top$, and $\lambda := \lambda_0 N$.

This formulation has a clean statistical interpretation (Rasmussen & Williams, 2006; Caponnetto & De Vito, 2007). Suppose the training labels are generated as $y_i = f(\boldsymbol{x}_i) + \epsilon_i$, where f is drawn from a zero-mean Gaussian process with covariance kernel $\mathcal{K}$, denoted $f \sim \mathrm{GP}(0, \mathcal{K})$, and $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ are i.i.d. observation noises. If the test label is the noiseless latent value $y_{N+1} = f(\boldsymbol{x}_{N+1})$, then setting $\lambda = \sigma^2$ makes the KRR predictor $f^*(\boldsymbol{x}_{N+1})$ coincide with the Bayes posterior mean.

Iterative solution via preconditioned Richardson iteration. As discussed in the introduction, the row-sum diagonal $\boldsymbol{D} = \mathrm{diag}(\boldsymbol{K}\boldsymbol{1})$ acts as a preconditioner: left-multiplying (2) by $\boldsymbol{D}^{-1}$ gives the equivalent system $\boldsymbol{D}^{-1}(\boldsymbol{K} + \lambda \boldsymbol{I})\boldsymbol{w} = \boldsymbol{D}^{-1}\boldsymbol{y}$, on which the preconditioned Richardson iteration with step size $\eta > 0$ reads

$$\boldsymbol{w}^{(\ell+1)} = \boldsymbol{w}^{(\ell)} + \eta \left(\boldsymbol{D}^{-1}\boldsymbol{y} - \boldsymbol{D}^{-1}(\boldsymbol{K} + \lambda \boldsymbol{I})\boldsymbol{w}^{(\ell)} \right). \tag{3}$$

Iteration (3) is the update that the transformer construction in Section 4 approximately implements step by step in its forward pass.

We compare plain and preconditioned Richardson iteration for solving the kernel system (2). Plain Richardson corresponds to iteration (3) with $\boldsymbol{D}$ replaced by the identity matrix. In both cases, the iteration converges if and only if $\eta \in (0, 2/\mathrm{eig}_{\max}(\boldsymbol{S}))$, where $\boldsymbol{S} = \boldsymbol{K} + \lambda \boldsymbol{I}$ for plain Richardson and $\boldsymbol{S} = \boldsymbol{D}^{-1/2}(\boldsymbol{K} + \lambda \boldsymbol{I})\boldsymbol{D}^{-1/2}$ for preconditioned Richardson. Figure 3 compares the stability threshold $2/\mathrm{eig}_{\max}(\boldsymbol{S})$ across context sizes N. For plain Richardson, this threshold decreases substantially with N, whereas for the preconditioned system it remains nearly constant, allowing a single step size across context sizes. However, this stabilization of the admissible step-size range does not lead to faster convergence in the regimes considered here. For either system, the theoretically optimal constant step size is $\eta = 2/(\mathrm{eig}_{\max}(\boldsymbol{S}) + \mathrm{eig}_{\min}(\boldsymbol{S}))$, which yields the optimal spectral convergence factor $(\kappa(\boldsymbol{S}) - 1)/(\kappa(\boldsymbol{S}) + 1)$. As shown in Figure 4, the corresponding matrices $\boldsymbol{S}$ for plain and preconditioned Richardson exhibit comparable condition numbers across N. Moreover, Figure 5 shows that plain and preconditioned Richardson require comparable numbers of iterations to reach the same relative coefficient error across context sizes N, with no systematic acceleration from the row-sum preconditioning.

In-context learning. For in-context learning, the transformer receives the training dataset $\mathcal{D}$, a new test input $\boldsymbol{x}_{N+1} \in \mathbb{R}^d$, and a set of auxiliary weights $\{w_i\}_{i=1}^N$ that act as placeholders for the intermediate iterates $\boldsymbol{w}^{(\ell)}$ in (3); the construction in Section 4 updates these weights through depth and reads off the prediction at $\boldsymbol{x}_{N+1}$.

To facilitate the required mathematical operations within the transformer's forward pass, we adopt two simple input augmentations: we include the squared norms $\|\boldsymbol{x}_j\|^2$ as a feature of each token (or, in the unit-norm setting, this feature is constant and can be omitted (Dragutinović et al., 2025)), and we prepend the sequence with a designated dummy token $\boldsymbol{x}_0 = \boldsymbol{0}$. We view these augmentations as *input preprocessing* rather than part of the core iterative-solver mechanism. This is consistent with prior constructive analyses of ICL, where input encoding, initialization, or readout operations are often separated from the main algorithmic update (von Oswald et al., 2023; Bai et al., 2023; Fu et al., 2024). Indeed, the squared-norm feature $\|\boldsymbol{x}_j\|^2$ can be approximated by an MLP-only transformer block; the dummy token $\boldsymbol{x}_0 = \boldsymbol{0}$ is also a mild input convention, analogous to special sequence tokens such as beginning-of-sequence markers in autoregressive language models (Touvron et al., 2023). We hard-code both in the input embedding to keep the iterative-solver mechanism clean.

Formally, we construct the input matrix $\boldsymbol{Z} = [\boldsymbol{z}_0, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_{N+1}] \in \mathbb{R}^{D \times (N+2)}$, with $D = d + 11$, where $\boldsymbol{z}_j = [\boldsymbol{x}_j; y_j; w_j; \|\boldsymbol{x}_j\|^2; \boldsymbol{0}_5; s_j; t_j; 1] \in \mathbb{R}^D$. Here, the five zero entries provide cached memory for intermediate computations, while the structural indicators s_j and t_j act as positional encodings that identify the dummy and test tokens, respectively: $s_0 = 1$ and $s_j = 0$ for $j \in [N+1]$, whereas $t_{N+1} = 1$ and $t_j = 0$ for $j \in \{0, \ldots, N\}$. We set the label and weight entries of the dummy and test tokens to zero. Applying the transformer gives $\widetilde{\boldsymbol{Z}} = \mathrm{TF}(\boldsymbol{Z})$, from which we extract the prediction for the test token via a readout function $y_{N+1} = \mathrm{readout}(\widetilde{\boldsymbol{Z}}) := \widetilde{\boldsymbol{Z}}_{d+1,N+1}$.

4 Main Result: An End-to-End Construction

We construct a single-head softmax-attention transformer with ReLU MLPs whose forward pass approximately implements preconditioned Richardson iteration (3) for the kernel system (2), with explicit bounds on the required depth and MLP width.

Assumptions. We impose a *bounded data* assumption: there exist $B_x, B_y > 0$ such that $\|\boldsymbol{x}_i\|_2 \le B_x$ and $|y_i| \le B_y$ for all $i \in [N+1]$. The kernel is the Gaussian kernel $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2/(2v^2))$ with bandwidth v, and the dual coefficients are initialized to zero, $\boldsymbol{w}^{(0)} = \boldsymbol{0}$.

Theorem 1 (Informal). *Fix $\lambda_0 \in (0, 1]$, $c \in (0, 1)$, $\epsilon \in (0, c)$, and a step size η satisfying (37). Let $\boldsymbol{w}^* = [w_i^* : i \in [N]]$ denote the solution to the KRR linear system (2). There exists a single-head transformer* TF *with $2L + 5$ blocks (three read-in blocks, two blocks for each of the L inexact preconditioned Richardson iterations, and two read-out blocks) for $L = \mathcal{O}(\log(1/\epsilon))$ and MLP width $W = \mathcal{O}(\sqrt{N/\epsilon})$ such that*

$$\left| \text{readout}(\text{TF}(\boldsymbol{Z})) - \sum_{i=1}^{N} w_i^* \, \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_{N+1}) \right| \le C_{\text{sys}} \, \epsilon.$$

The system constant C_{sys} depends on regularization parameter λ_0, kernel bandwidth v, step size η, data bounds B_x, B_y, and constant c, but is independent of the prompt length N.

The formal version of the above theorem appears as Theorem 11 in Appendix C.5, with explicit expressions (38)–(40) for the depth, MLP width, and the N-independent constant C_{sys}, including their parameter dependence. The proof is constructive, and we sketch the network construction here; a full overview is given in Appendix C.1. The transformer is built in three phases. The *read-in phase* (Appendix C.2) uses three blocks to prepare quantities needed by the iterative solver, including approximations of $\boldsymbol{D}^{-1}$ and $\boldsymbol{D}^{-1}\boldsymbol{y}$. The *iteration phase* (Appendix C.3) uses pairs of transformer blocks to approximately implement preconditioned Richardson updates for the coefficients $\boldsymbol{w}$. Finally, the *read-out phase* (Appendix C.4) extracts the prediction at the test token $\boldsymbol{x}_{N+1}$, yielding an approximation of $\sum_{i=1}^{N} w_i^* \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_{N+1})$.

Algorithm-Architecture correspondence. Across these phases, the two components of a standard transformer play complementary algorithmic roles. Softmax attention handles the *cross-token* operations: it produces row-normalized kernel interactions and computes products such as $\boldsymbol{D}^{-1}\boldsymbol{K}\boldsymbol{w}^{(\ell)}$. ReLU MLPs act locally at each token, approximating the *intra-token* scalar arithmetic, including elementwise operations such as $\boldsymbol{D}^{-1}\boldsymbol{y}$ and $\boldsymbol{D}^{-1}\boldsymbol{w}^{(\ell)}$. This separation of roles is central to our transformer construction: its forward pass implements an inexact preconditioned Richardson iteration within the standard softmax-attention/MLP architecture.

Scaling of depth and width. The depth and width scalings in Theorem 1 come from two different sources. The depth $\mathcal{O}(\log(1/\epsilon))$ is an *algorithmic count* obtained from the error analysis of the inexact preconditioned Richardson iteration (Appendix A). With a suitable choice of step size η, the error-propagation matrix of the underlying iteration is contractive. Consequently, a logarithmic number of inexact iteration steps suffices to reach the desired prediction accuracy. In contrast, the MLP width scaling is an *approximation count*: ReLU MLPs are used to approximate the token-wise scalar arithmetic operations required by the construction (Appendix B). Common single-layer ReLU approximation results used in the ICL literature are input-dimension-free, thereby avoiding the curse of dimensionality (Bach, 2017; Bai et al., 2023). In our construction, however, the MLPs only need to approximate a few specific one-dimensional smooth functions, such as $x/(1-x)$, x^2, and $1/x$, on intervals of interest. We can therefore use explicit linear spline approximations, which are representable by single-layer ReLU networks and achieve sharper interpolation rates (De Boor, 1978; DeVore, 1998; DeVore et al., 2021). By carefully estimating the relevant approximation intervals and error tolerances, we obtain the required MLP width scaling $\mathcal{O}(\sqrt{N/\epsilon})$ for each scalar approximation. Together, the depth scaling reflects the convergence rate of the inexact preconditioned Richardson iteration, while the width scaling reflects the cost of approximating the local scalar arithmetic.

Variation of the regularization parameter. Theorem 1 assumes that λ_0 is a constant independent of the prompt length N. Recall that the regularization parameter in KRR is related to λ_0 through

$\lambda = \lambda_0 N$. Thus, if λ is held fixed as N varies, then $\lambda_0 = \lambda/N$ decreases with the prompt length. Substituting $\lambda_0 = \lambda/N$ into the explicit expressions for the system constant, depth, and MLP width in (38), (39), and (40) shows that, to guarantee a final prediction error of at most $\tilde{\epsilon}$, it suffices to use depth $L = \mathcal{O}(N \log(N/\tilde{\epsilon}))$, MLP width $W = \mathcal{O}\left(N^{11/4}/\sqrt{\tilde{\epsilon}}\right)$. Thus, unlike the constant-λ_0 regime, fixing the KRR regularization parameter λ, our sufficient depth bound increases with the number N of in-context examples. As we will see in the numerical experiments, this prediction is qualitatively consistent with the observation that more transformer blocks are required to implement Gaussian KRR for longer prompts when λ is fixed at the training noise variance.

5 Experiments

We empirically show that trained transformers progressively approach exact KRR estimator across depth and further investigate how this refinement relates to classical iterative KRR solvers.

Data generation. We consider dynamically generated Gaussian process (GP) regression tasks. For each training and evaluation sequence, we sample inputs $\boldsymbol{x}_i \overset{\text{i.i.d.}}{\sim} \mathcal{P}$, $i \in [N+1]$, and $f \sim \mathrm{GP}(0, \mathcal{K})$ with Gaussian kernel bandwidth v. Context labels are $y_i = f(\boldsymbol{x}_i) + \epsilon_i$, with $\epsilon_i \sim \mathcal{N}(0, \sigma^2_{\text{train}})$ for $i \in [N]$. We set $N = 40$, $d = 5$, $\sigma_{\text{train}} = 0.05$, and $v = 1$. We use three input distributions $\mathcal{P}$: Uniform $\mathrm{Unif}([-1,1]^d)$, Gaussian $\mathcal{N}(0, 0.6^2 \boldsymbol{I}_d)$, and Spherical $\mathrm{Unif}(\mathbb{S}^{d-1})$.

Transformer architecture and pretraining. For each input distribution, we train a GPT-2-style decoder-only transformer, with 12 layers and 8 softmax-attention heads, from scratch (Radford et al., 2019; Garg et al., 2022), minimizing noisy-label MSE at every in-context position $n \in [N]$. Pretraining runs for 500,000 optimizer steps with batch size 64. All methods are evaluated on the same 256 fresh GP sequences per seed across 5 fixed evaluation seeds. Full details are in Appendix E.

5.1 Progressive alignment with exact KRR across depth

Prediction-error alignment. Figure 1(a) reports the transformer's layer-wise prediction MSE on spherical inputs for selected context lengths $n \in \{2, 10, 15, 20, 25, 30, 35, 40\}$, where $N = 40$ is the maximum context length. The exact-KRR predictor is obtained by directly solving the kernel linear system in (2). For a fixed context length n, let $\mathcal{A}_n^{(s,i)}$ denote the predictor produced using the first n labeled context examples $\{\boldsymbol{x}_j^{(s,i)}, y_j^{(s,i)}\}_{j=1}^n$ of sequence i under evaluation seed s, where $\mathcal{A} = \mathrm{TF}^{(\ell)}$ denotes the predictor from transformer block ℓ and $\mathcal{A} = \mathrm{KRR}$ denotes the exact-KRR predictor. We compute the mean-squared error as $\mathrm{MSE}_{\mathcal{A}}(n) = \underset{s\in[5], i\in[256]}{\mathbb{E}}\left[\left(\mathcal{A}_n^{(s,i)}(\boldsymbol{x}_{n+1}^{(s,i)}) - y_{n+1}^{(s,i)}\right)^2\right]$, where the expectation denotes the empirical average over the 5 evaluation seeds and 256 held-out sequences per seed. Solid curves show transformer MSEs and horizontal dashed lines the exact-KRR references, evaluated on the same prediction problems. The transformer's MSE decreases overall with depth, most sharply in the first few layers. Shorter contexts approach their KRR references sooner, while longer contexts benefit from additional depth. By the final layers, transformer MSEs approach the corresponding exact-KRR values across all displayed context lengths.

Induced-weight alignment. The exact-KRR predictor admits an explicit representation through its dual coefficient vector, whereas a transformer layer exposes only scalar predictions on query inputs. To compare them in the same dual coefficient space, for each regression problem defined by a given context length, we view each transformer block as a kernel machine, and interpret its dual coefficients as the block's *induced weights* at that context. Following prior work on output-based weight recovery in linear-regression ICL (Akyürek et al., 2023; Fu et al., 2024), we adapt this procedure to KRR. For each sequence and fixed context length n, we sample $P = 200$ independent probe queries $\boldsymbol{q}_r \in \mathbb{R}^d$, $r \in [P]$, from the same input distribution and construct the probe matrix $\boldsymbol{\Phi} \in \mathbb{R}^{P\times n}$, with $\boldsymbol{\Phi}_{rj} := \mathcal{K}(\boldsymbol{q}_r, \boldsymbol{x}_j)$. Applying the trained block-wise linear probe to the query-position hidden states produces the prediction vector $\boldsymbol{F} \in \mathbb{R}^P$ at block ℓ. Then the transformer-induced weight vector can be computed by $\underset{\boldsymbol{w}\in\mathbb{R}^n}{\mathrm{argmin}}\left\{\|\boldsymbol{F} - \boldsymbol{\Phi}\boldsymbol{w}\|_2^2 + \tau\|\boldsymbol{w}\|_2^2\right\} = \left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \tau\boldsymbol{I}\right)^{-1}\boldsymbol{\Phi}^\top\boldsymbol{F}$, with $\tau = 10^{-5}$. For sequence i under evaluation seed s, let $\boldsymbol{w}^{(s,i)}_{\mathrm{TF}^{(\ell)},n}$ denote the transformer-induced

weights and let $\boldsymbol{w}_{\mathrm{KRR},n}^{(s,i)}$ denote the exact-KRR dual coefficients for the same context of length n. We define their similarity as $\mathrm{SimW}\left(\mathrm{TF}^{(\ell)}, \mathrm{KRR}\,;\, n\right) = \mathop{\mathbb{E}}_{s\in[5],i\in[256]} \mathcal{C}\left(\boldsymbol{w}_{\mathrm{TF}^{(\ell)},n}^{(s,i)}, \boldsymbol{w}_{\mathrm{KRR},n}^{(s,i)}\right)$, where $\mathcal{C}(\boldsymbol{u},\boldsymbol{v}) := \frac{\langle \boldsymbol{u},\boldsymbol{v}\rangle}{\|\boldsymbol{u}\|_2\|\boldsymbol{v}\|_2}$ is the cosine similarity. Figure 1(b) reports SimW at the maximal considered context length, $n = 40$, showing that the transformer's induced weights become progressively more aligned with the exact-KRR coefficients as depth increases across all three input distributions. Figure 7 extends this comparison across context lengths for all three input distributions and five random seeds, showing that longer contexts generally require greater depth to reach comparable levels of alignment. Our experiments use a fixed KRR regularization parameter λ, determined by the noise variance. As discussed after Theorem 1, fixing λ implies $\lambda_0 = \lambda/N$, and our theoretical construction therefore predicts that the depth increases with the number of in-context examples. The empirical dependence on context length is qualitatively consistent with this prediction. We finally remark that the induced weights used to compute SimW are externally recovered dual coefficients from the transformer's predictions; they are not transformer parameters, attention weights, or directly observed internal states.

5.2 Ablation studies

Robustness to linear probing. Our layer-wise predictions are obtained using linear probes trained with the transformer backbone frozen, raising the question whether the trends in Figure 1 are induced by the probes rather than by the backbone itself. We first test this using two ablations that replace the trained backbone with a randomly initialized one: one inherits the probes trained on the trained backbone, while the other retrains the probes on the random backbone. Figure 8 shows that, at $n = 40$, neither control reproduces the trained model's behavior: SimW remains near zero across all layers, while the MSE stays close to its initial-layer value. As a complementary control, we remove the separately trained probes and read each intermediate hidden state using the trained transformer's native final LayerNorm and prediction head. At depth ℓ, we apply this native readout directly to the hidden states and treat the resulting scalar as the layer-ℓ prediction. Figure 9 compares this readout with the per-layer linear probes. Although the native readout yields a noisier MSE trajectory at intermediate depths, its predictions progressively approach those of the probes and agree closely in the final layers. The layer-wise SimW trend seems robust to the choice of readout. These results suggest that the probes mainly *calibrate* intermediate hidden states into scalar predictions. Together, we conclude that the linear probe does not manufacture the observed trend; rather, the transformer backbone progressively refines its predictions toward the KRR predictor across depth.

Mismatched estimator kernel. We examine whether the observed alignment is specific to the Gaussian kernel used to generate the training data. To this end, we keep the model, spherical inputs, and Gaussian labels fixed, but replace the Gaussian estimator kernel with the arc-cosine kernel (Cho & Saul, 2009) in both the exact-KRR and the induced-weight recovery comparison. Figure 10 shows substantially lower SimW with arc-cosine KRR: alignment initially increases but declines at later layers, while alignment with Gaussian KRR continues to improve. The prediction error shows a similar separation: the transformer remains close to the Gaussian KRR error but clearly separated from that of arc-cosine KRR. These results suggest that the deeper layers progressively refine the prediction toward Gaussian KRR specifically, rather than toward KRR solutions based on arbitrary kernels.

Noise-level ablation. We examine whether a Transformer trained at noise level σ_{train} behaves like KRR with λ being $\sigma_{\mathrm{train}}^2$. To this end, we vary the noise level σ_{test}, measure the error against the noiseless labels, and report the ratio $\mathrm{MSE}_{\mathrm{TF}}/\mathrm{MSE}_{\mathrm{KRR}}$, where KRR uses $\lambda = \sigma_{\mathrm{train}}^2$. We report the results in Figure 11. We find that, across all three input distributions, the ratio generally moves closer to one in the later layers. Whether the late-layer ratio lies above or below one depends on the test noise: it remains above one for small σ_{test} but falls below one for large σ_{test}. Both behaviors may be explained by finite depth (early stopping) acting as additional regularization, which is harmful when the labels are nearly clean but beneficial when they are noisy.

5.3 Comparison with classical KRR solvers

Classical baselines. We compare the per-layer transformer predictions against the exact KRR solution and five classical iterative methods for KRR. The exact KRR solution is $\boldsymbol{w}^\star = (\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}$,

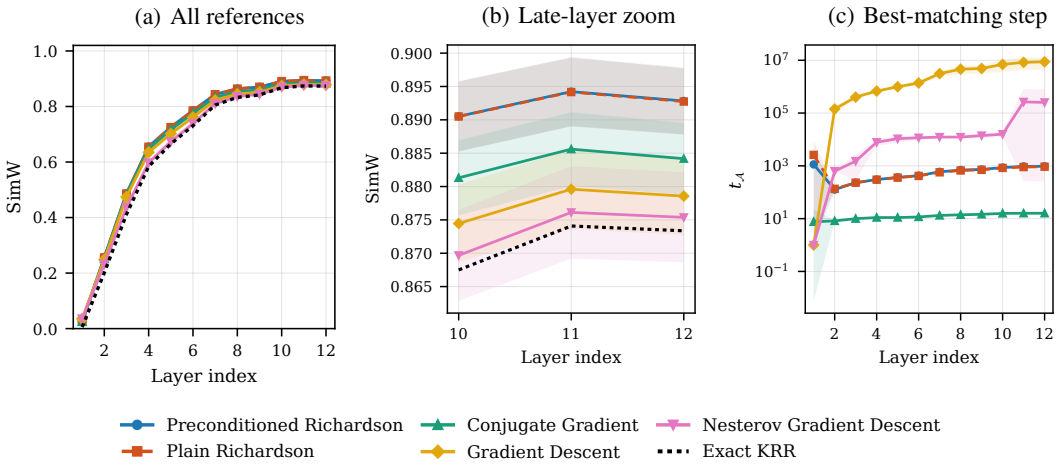


Figure 2: **Best-match SimW and the corresponding solver iterations for spherical inputs** at $n = 40$. SimW generally increases with depth, while its late-layer values are closely clustered across the five classical solvers. These trends are consistent with progressive refinement toward their shared exact KRR solution.

which is also the common limiting solution of the five iterative methods considered below. Three are linear-system solvers applied to the kernel system (2): *plain Richardson iteration*, *preconditioned Richardson iteration*, and *conjugate gradient*. The other two are loss-function optimizers applied to the RKHS-regularized loss $L(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{K}\boldsymbol{w} - \boldsymbol{y}\|^2 + \frac{1}{2}\lambda \boldsymbol{w}^\top \boldsymbol{K}\boldsymbol{w}$: *gradient descent* and *Nesterov gradient descent*. Full details for each method are provided in Appendix D.

Best-matching step. We explore which iterate of each classical solver is most similar to the transformer-induced weights. For a solver $\mathcal{A}$, let $\boldsymbol{w}^{(s,i)}_{\mathcal{A}^{(t)},n}$ denote its coefficient vector after iteration t on sequence i under evaluation seed s. We define the best-matching iteration as $t_{\mathcal{A}}(\ell; n) := \mathop{\mathbb{E}}_{s\in[5]}\left[\operatorname{argmax}_{t\in[T]} \mathop{\mathbb{E}}_{i\in[256]} \mathcal{C}\left(\boldsymbol{w}^{(s,i)}_{\mathrm{TF}^{(\ell)},n}, \boldsymbol{w}^{(s,i)}_{\mathcal{A}^{(t)},n}\right)\right]$, where T is chosen sufficiently large to provide enough iterations for each solver to converge.

Results. Figure 2(a) reports the best-match SimW at $n = 40$ for spherical inputs, Figure 2(b) enlarges the late-layer values, and Figure 2(c) shows the corresponding best-matching steps. The best-match SimW generally increases with depth for all five classical methods, while the late-layer values remain closely clustered. This is consistent with the fact that all five methods converge to the same exact KRR solution: as the transformer-induced weights approach this common endpoint, they become simultaneously similar to late iterates of multiple solvers. Thus, SimW supports progressive refinement toward the KRR solution but does not, by itself, identify the underlying iterative algorithm. Figure 2(c) primarily reflects the relative convergence speeds of the solvers; in particular, plain and preconditioned Richardson converge at nearly the same rate in this setting. The role of preconditioned Richardson in our analysis instead comes from the constructive theory, through the correspondence between softmax attention and row-sum preconditioning. We finally remark that the same pattern holds for Gaussian and Uniform inputs in Figures 12 and 13.

6 Discussion

Our construction establishes inexact preconditioned Richardson iteration as a mechanism by which standard softmax-attention transformers can approximate in-context Gaussian KRR. Our experiments support progressive refinement toward KRR, but may not conclusively establish that trained transformers internally implement this particular solver. Since our analysis relies on the RKHS representer theorem, one possible direction is to investigate whether analogous solver constructions extend to reproducing kernel Banach spaces (Parhi & Nowak, 2021; Bartolucci et al., 2023; Wang et al., 2024; 2025).

AI USE STATEMENT

In this work, we used generative AI tools to polish our original writing, identify additional relevant literature, and assist with debugging code and configuring computational workflows. We have reviewed all AI-assisted work, including revisions to the text, suggested references, and changes to the code. We take full responsibility for the final content of this work, including any text, claims, code, or other artifacts produced with the assistance of generative AI.

REFERENCES

Kwangjun Ahn, Xiang Cheng, Hadi Daneshmand, and Suvrit Sra. Transformers learn to implement preconditioned gradient descent for in-context learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 45614–45650, 2023.

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. In *The Eleventh International Conference on Learning Representations*, 2023.

Ekin Akyürek, Bailin Wang, Yoon Kim, and Jacob Andreas. In-context language learning: Architectures and algorithms. In *International Conference on Machine Learning*. PMLR, 2024.

Francis Bach. Breaking the curse of dimensionality with convex neural networks. *Journal of Machine Learning Research*, 18(19):1–53, 2017.

Francis Bach. *Learning Theory from First Principles*. MIT Press, 2024.

Yu Bai, Fan Chen, Huan Wang, Caiming Xiong, and Song Mei. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. In *Advances in Neural Information Processing Systems*, volume 36, pp. 57125–57211, 2023.

Francesca Bartolucci, Ernesto De Vito, Lorenzo Rosasco, and Stefano Vigogna. Understanding neural networks with reproducing kernel Banach spaces. *Applied and Computational Harmonic Analysis*, 62:194–236, 2023.

Satwik Bhattamishra, Arkil Patel, Phil Blunsom, and Varun Kanade. Understanding in-context learning in transformers and LLMs by learning to learn discrete functions. In *International Conference on Learning Representations*, 2024.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020.

Andrea Caponnetto and Ernesto De Vito. Optimal rates for the regularized least-squares algorithm. *Foundations of Computational Mathematics*, 7(3):331–368, 2007.

Siyu Chen, Heejune Sheen, Tianhao Wang, and Zhuoran Yang. Training dynamics of multi-head softmax attention for in-context learning: Emergence, convergence, and optimality. In *The Thirty Seventh Annual Conference on Learning Theory*, volume 247, pp. 4573–4573. PMLR, 2024.

Xiang Cheng, Yuxin Chen, and Suvrit Sra. Transformers implement functional gradient descent to learn non-linear functions in context. In *International Conference on Machine Learning*, volume 235, pp. 8002–8037. PMLR, 2024.

Youngmin Cho and Lawrence K. Saul. Kernel methods for deep learning. In *Advances in Neural Information Processing Systems 22*, 2009.

Frank Cole, Yuxuan Zhao, Yulong Lu, and Tianhao Zhang. In-context learning of linear dynamical systems with transformers: Approximation bounds and depth-separation. *Advances in Neural Information Processing Systems*, 38:169441–169479, 2026.

Liam Collins, Advait Parulekar, Aryan Mokhtari, Sujay Sanghavi, and Sanjay Shakkottai. In-context learning with transformers: Softmax attention adapts to function lipschitzness. In *Advances in Neural Information Processing Systems*, volume 37, pp. 92638–92696, 2024.

Carl De Boor. *A practical guide to splines*, volume 27. springer New York, 1978.

Samet Demir and Zafer Dogan. Asymptotic study of in-context learning with random transformers through equivalent models, 2025.

Ronald DeVore. Nonlinear approximation. *Acta numerica*, 7:51–150, 1998.

Ronald DeVore, Boris Hanin, and Guergana Petrova. Neural network approximation. *Acta Numerica*, 30:327–444, 2021.

Sara Dragutinović, Andrew M. Saxe, and Aaditya K. Singh. Softmax $\geq$ linear: Transformers may learn to classify in-context by kernel gradient descent. *arXiv preprint arXiv:2510.10425*, 2025.

Deqing Fu, Tian-Qi Chen, Robin Jia, and Vatsal Sharan. Transformers learn to achieve second-order convergence rates for in-context linear regression. In *Advances in Neural Information Processing Systems*, volume 37, pp. 98675–98716, 2024.

Shivam Garg, Dimitris Tsipras, Percy S. Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. In *Advances in Neural Information Processing Systems*, volume 35, pp. 30583–30598, 2022.

Gautam Goel and Peter Bartlett. Can a transformer represent a Kalman filter? In *6th Annual Learning for Dynamics & Control Conference*, pp. 1502–1512. PMLR, 2024.

Chi Han, Ziqi Wang, Han Zhao, and Heng Ji. Understanding emergent in-context learning from a kernel regression perspective. *Transactions on Machine Learning Research*, 2025.

Yu Huang, Yuan Cheng, and Yingbin Liang. In-context convergence of transformers. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 19660–19722. PMLR, 2024.

Gyeonghun Kang, Changwoo J Lee, and Xiang Cheng. Transformers can learn posterior predictive distributions in-context. *arXiv preprint arXiv:2605.26713*, 2026.

Zihao Li, Yuan Cao, Cheng Gao, Yihan He, Han Liu, Jason M Klusowski, Jianqing Fan, and Mengdi Wang. One-layer transformer provably learns one-nearest neighbor in context. *Advances in Neural Information Processing Systems*, 37:82166–82204, 2024.

Yue M. Lu, Mary I. Letey, Jacob A. Zavatone-Veth, Anindita Maiti, and Cengiz Pehlevan. Asymptotic theory of in-context learning by linear attention. *Proceedings of the National Academy of Sciences*, 122(28):e2502599122, 2025.

Arvind V. Mahankali, Tatsunori B. Hashimoto, and Tengyu Ma. One step of gradient descent is provably the optimal in-context learner with one layer of linear self-attention. In *International Conference on Learning Representations*, 2024.

Rahul Parhi and Robert D Nowak. Banach space representer theorems for neural networks and ridge splines. *Journal of Machine Learning Research*, 22(43):1–40, 2021.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Victor Ryaben'kii and Semyon Tsynkov. *A Theoretical Introduction to Numerical Analysis*. Chapman & Hall, 2006.

Michael E Sander and Gabriel Peyré. Towards understanding the universality of transformers for next-token prediction. *arXiv preprint arXiv:2410.03011*, 2024.

Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola. A generalized representer theorem. In *International conference on computational learning theory*, pp. 416–426. Springer, 2001.

Zhaiming Shen, Alexander Hsu, Rongjie Lai, and Wenjing Liao. Understanding in-context learning on structured manifolds: Bridging attention to kernel methods. *arXiv preprint arXiv:2506.10959*, 2025.

Jonathan R Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. 1994.

Haoyuan Sun, Ali Jadbabaie, and Navid Azizan. On the role of transformer feed-forward layers in nonlinear in-context learning, 2025.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Max Vladymyrov, Johannes von Oswald, Mark Sandler, and Rong Ge. Linear transformers are versatile in-context learners. In *Advances in Neural Information Processing Systems*, volume 37, pp. 48784–48809, 2024.

Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pp. 35151–35174. PMLR, 2023.

Rui Wang, Yuesheng Xu, and Mingsong Yan. Sparse representer theorems for learning in reproducing kernel Banach spaces. *Journal of Machine Learning Research*, 25(93):1–45, 2024.

Rui Wang, Yuesheng Xu, and Mingsong Yan. Hypothesis spaces for deep learning. *Neural Networks*, pp. 107995, 2025.

Ruiqi Zhang, Spencer Frei, and Peter L. Bartlett. Trained transformers learn linear models in-context. *Journal of Machine Learning Research*, 25(49):1–55, 2024a.

Ruiqi Zhang, Jingfeng Wu, and Peter L. Bartlett. In-context learning of a linear transformer block: Benefits of the MLP component and one-step GD initialization. In *Advances in Neural Information Processing Systems*, volume 37, pp. 18310–18361, 2024b.

Yedi Zhang, Aaditya K Singh, Peter E Latham, and Andrew Saxe. Training dynamics of in-context learning in linear attention. *arXiv preprint arXiv:2501.16265*, 2025.

A Error Analysis of Inexact Preconditioned Richardson Iterations

In this section, we present an error analysis of the inexact preconditioned Richardson iteration, an algorithm we will subsequently implement via a specifically constructed transformer. We proceed with a helpful lemma.

Lemma 2. *Let $\lambda_0 > 0$ and $\lambda = \lambda_0 N$. Suppose that $\boldsymbol{w}^* = [w_i^* : i \in [N]]$ is the solution of the linear system (2) with the bounded data assumption (i.e., $\|\boldsymbol{y}\|_\infty \le B_y$). Then it holds that*

$$\|\boldsymbol{K}\boldsymbol{w}^*\|_\infty \le \frac{B_y}{\sqrt{\lambda_0}}, \quad \|\boldsymbol{w}^*\|_\infty \le \frac{\left(\frac{1}{\sqrt{\lambda_0}}+1\right) B_y}{\lambda_0}\frac{1}{N}.$$

Proof. It is known by representer theorem that $f^*(x) = \sum_{i=1}^N w_i^* \mathcal{K}(\boldsymbol{x}_i, \cdot)$ is the optimal solution of (1). We denote $\mathcal{L}(f) := \frac{1}{N}\sum_{i\in[N]} \left(f(\boldsymbol{x}_i) - y_i\right)^2 + \lambda_0 \|f\|_{\mathcal{H}}^2$. The optimality of f^* implies

$$\lambda_0 \|f^*\|_{\mathcal{H}}^2 \le \mathcal{L}(f^*) \le \mathcal{L}(0) = \frac{1}{N}\sum_{i=1}^N y_i^2 \le B_y^2,$$

and hence $\|f^*\|_{\mathcal{H}} \le \frac{B_y}{\sqrt{\lambda_0}}$. By the reproducing property in RKHS, i.e., $f^*(\boldsymbol{x}) = \langle f^*, \mathcal{K}(\boldsymbol{x}, \cdot)\rangle_{\mathcal{H}}$, we get that

$$|f^*(\boldsymbol{x})| \le \|f^*\|_{\mathcal{H}} \|\mathcal{K}(\boldsymbol{x}, \cdot)\|_{\mathcal{H}} = \|f^*\|_{\mathcal{H}} \sqrt{\mathcal{K}(\boldsymbol{x}, \boldsymbol{x})} \le \frac{B_y}{\sqrt{\lambda_0}}, \quad \text{for all } \boldsymbol{x} \in \mathbb{R}^d.$$

Note that $\boldsymbol{K}\boldsymbol{w}^* = [f^*(\boldsymbol{x}_i) : i \in [N]]$. Consequently, we obtain from the previous estimate that

$$\|\boldsymbol{K}\boldsymbol{w}^*\|_\infty = \max_{i\in[N]} |f^*(\boldsymbol{x}_i)| \le \frac{B_y}{\sqrt{\lambda_0}}.$$

Moreover, from the linear system (2), we get

$$\|\boldsymbol{w}^*\|_\infty = \frac{\|\boldsymbol{K}\boldsymbol{w}^* - \boldsymbol{y}\|_\infty}{\lambda} \le \frac{\|\boldsymbol{K}\boldsymbol{w}^*\|_\infty + \|\boldsymbol{y}\|_\infty}{\lambda} \le \frac{\left(\frac{1}{\sqrt{\lambda_0}}+1\right) B_y}{\lambda_0}\frac{1}{N}.$$

This completes the proof. □

We recall that the bounded data assumption (i.e., $\|\boldsymbol{x}_i\|_2 \le B_x$ for all samples $\boldsymbol{x}_i$) and boundedness of the Gaussian kernel imply that for all $i \in [N]$,

$$N\kappa_{\min} \le \sum_{j\in[N]} \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) \le N \tag{4}$$

where $\kappa_{\min} := \exp\left(-\frac{2B_x^2}{v^2}\right)$.

Theorem 3 (Error Analysis for Inexact Preconditioned Richardson Iterations)**.** *Suppose that the bounded data assumption holds with constants B_x and B_y. Let initial value $\boldsymbol{w}^{(0)} = \boldsymbol{0}$. Let $\epsilon_{\text{flip}}, \epsilon_{\text{sq}}, \widetilde{\epsilon}_{\text{sq}} \in (0,1)$. Let $\lambda_0, \eta > 0$ and $\lambda = \lambda_0 N$. Consider inexact preconditioned Richardson iteration*

$$\boldsymbol{w}^{(\ell+1)} = \boldsymbol{w}^{(\ell)} + \eta\left(\boldsymbol{D}^{-1}\boldsymbol{y} - \boldsymbol{D}^{-1}\left(\boldsymbol{K} + \lambda\boldsymbol{I}\right)\boldsymbol{w}^{(\ell)}\right) + \eta\left(\left(\boldsymbol{y} - \lambda\boldsymbol{w}^{(\ell)}\right) \odot \boldsymbol{r} + \frac{\boldsymbol{\tau}_+ - \boldsymbol{\tau}_- - \lambda\widetilde{\boldsymbol{\tau}}_+^{(\ell)} + \lambda\widetilde{\boldsymbol{\tau}}_-^{(\ell)}}{4}\right), \tag{5}$$

with

$$\|\boldsymbol{r}\|_\infty \le \frac{\epsilon_{\text{flip}}}{N}, \quad \|\boldsymbol{\tau}_+\|_\infty, \|\boldsymbol{\tau}_-\|_\infty \le \frac{\epsilon_{\text{sq}}}{N}, \quad \left\|\widetilde{\boldsymbol{\tau}}_+^{(\ell)}\right\|_\infty, \left\|\widetilde{\boldsymbol{\tau}}_-^{(\ell)}\right\|_\infty \le \frac{\widetilde{\epsilon}_{\text{sq}}}{N^2}. \tag{6}$$

Define

$$\boldsymbol{A} := \boldsymbol{I} - \eta\left(\lambda\,\mathrm{diag}(\boldsymbol{r}) + \boldsymbol{D}^{-1/2}\left(\boldsymbol{K} + \lambda\boldsymbol{I}\right)\boldsymbol{D}^{-1/2}\right). \tag{7}$$

If the step size η satisfies

$$0 < \eta < \frac{1}{\lambda_0 \epsilon_{\text{flip}} + 1 + \lambda_0/\kappa_{\min}}, \tag{8}$$

then it holds that

$$\|\boldsymbol{A}\|_{2\to 2} \le 1 - \eta\lambda_0 (1 - \epsilon_{\text{flip}}) \in (0, 1), \tag{9}$$

and

$$\sqrt{N} \left\| \boldsymbol{w}^{(\ell)} - \boldsymbol{w}^* \right\|_2 \le \|\boldsymbol{A}\|_{2\to 2}^{\ell} \frac{\left(\frac{1}{\sqrt{\lambda_0}} + 1\right) B_y}{\lambda_0 \sqrt{\kappa_{\min}}} + \frac{\eta \left(\frac{B_y}{\sqrt{\lambda_0}} \epsilon_{\text{flip}} + (1 + \lambda_0)(\epsilon_{\text{sq}} + \widetilde{\epsilon}_{\text{sq}}) \right)}{(1 - \|\boldsymbol{A}\|_{2\to 2}) \sqrt{\kappa_{\min}}}. \tag{10}$$

Proof. We first prove (9). We denote $\boldsymbol{K}_\lambda := \lambda \text{diag}(\boldsymbol{r}) + \boldsymbol{D}^{-1/2} (\boldsymbol{K} + \lambda \boldsymbol{I}) \boldsymbol{D}^{-1/2}$. Then the matrix $\boldsymbol{A}$ can be rewritten as $\boldsymbol{A} = \boldsymbol{I} - \eta \boldsymbol{K}_\lambda$. Since $\boldsymbol{A}$ is symmetric, $\|\boldsymbol{A}\|_{2\to 2}$ equals its maximum absolute eigenvalue. Note that the eigenvalues of $\boldsymbol{A}$ satisfy

$$\text{eig}(\boldsymbol{A}) = 1 - \eta \, \text{eig}(\boldsymbol{K}_\lambda).$$

By Weyl's inequality, the maximum eigenvalue of $\boldsymbol{K}_\lambda$ is bounded by

$$\begin{aligned} \text{eig}_{\max}(\boldsymbol{K}_\lambda) &\le \text{eig}_{\max}(\lambda \text{diag}(\boldsymbol{r})) + \text{eig}_{\max}(\boldsymbol{D}^{-1/2} \boldsymbol{K} \boldsymbol{D}^{-1/2}) + \text{eig}_{\max}(\lambda \boldsymbol{D}^{-1}) \\ &\le \lambda_0 \epsilon_{\text{flip}} + 1 + \frac{\lambda_0}{\kappa_{\min}}, \end{aligned} \tag{11}$$

where we used assumption (6), relation (4), $\lambda = \lambda_0 N$ and $\text{eig}_{\max}(\boldsymbol{D}^{-1/2} \boldsymbol{K} \boldsymbol{D}^{-1/2}) \le 1$. Assumption (8) of η guarantees $\eta \, \text{eig}_{\max}(\boldsymbol{K}_\lambda) < 1$. Consequently, all eigenvalues of $\boldsymbol{A}$ are strictly positive, i.e.,

$$\text{eig}_{\min}(\boldsymbol{A}) = 1 - \eta \, \text{eig}_{\max}(\boldsymbol{K}_\lambda) > 0.$$

It follows that the operator norm of $\boldsymbol{A}$ is exactly its maximum eigenvalue. Hence,

$$\|\boldsymbol{A}\|_{2\to 2} = \text{eig}_{\max}(\boldsymbol{A}) = 1 - \eta \, \text{eig}_{\min}(\boldsymbol{K}_\lambda).$$

Applying Weyl's inequality to the minimum eigenvalue gives

$$\text{eig}_{\min}(\boldsymbol{K}_\lambda) \ge \text{eig}_{\min}(\lambda \text{diag}(\boldsymbol{r})) + \text{eig}_{\min}(\boldsymbol{D}^{-1/2} \boldsymbol{K} \boldsymbol{D}^{-1/2}) + \text{eig}_{\min}(\lambda \boldsymbol{D}^{-1}) \ge -\lambda_0 \epsilon_{\text{flip}} + \lambda_0, \tag{12}$$

where we used assumption (6), relation (4), $\lambda = \lambda_0 N$ and the fact that matrix $\boldsymbol{D}^{-1/2} \boldsymbol{K} \boldsymbol{D}^{-1/2}$ is positive semi-definite. Therefore, the operator norm is bounded as

$$\|\boldsymbol{A}\|_{2\to 2} \le 1 - \eta\lambda_0 (1 - \epsilon_{\text{flip}}).$$

We finally verify the upper bound $1 - \eta\lambda_0 (1 - \epsilon_{\text{flip}})$ is in $(0, 1)$. It suffices to show that

$$\eta\lambda_0 (1 - \epsilon_{\text{flip}}) \in (0, 1). \tag{13}$$

We obtain from (11) and (12) that $\lambda_0(1 - \epsilon_{\text{flip}}) \le \lambda_0 \epsilon_{\text{flip}} + 1 + \frac{\lambda_0}{\kappa_{\min}}$, and hence (13) is clearly true due to assumption (8) for η. This completes the proof of (9).

In the following, we present the proof of (10). We subtract $\boldsymbol{w}^*$ for both sides of (5), and note that the exact solution $\boldsymbol{w}^*$ satisfies $(\boldsymbol{K} + \lambda \boldsymbol{I})\boldsymbol{w}^* = \boldsymbol{y}$. Then

$$\begin{aligned} \boldsymbol{w}^{(\ell+1)} - \boldsymbol{w}^* &= \boldsymbol{w}^{(\ell)} - \boldsymbol{w}^* + \eta \boldsymbol{D}^{-1} (\boldsymbol{K} + \lambda \boldsymbol{I}) \left(\boldsymbol{w}^* - \boldsymbol{w}^{(\ell)} \right) + \eta \left(\left((\boldsymbol{K} + \lambda \boldsymbol{I}) \boldsymbol{w}^* - \lambda \boldsymbol{w}^{(\ell)} \right) \odot \boldsymbol{r} + \boldsymbol{\Delta}^{(\ell)} \right) \\ &= \left(\boldsymbol{I} - \eta \left(\lambda \text{diag}(\boldsymbol{r}) + \boldsymbol{D}^{-1} (\boldsymbol{K} + \lambda \boldsymbol{I}) \right) \right) \left(\boldsymbol{w}^{(\ell)} - \boldsymbol{w}^* \right) + \eta \left((\boldsymbol{K} \boldsymbol{w}^*) \odot \boldsymbol{r} + \boldsymbol{\Delta}^{(\ell)} \right) \end{aligned} \tag{14}$$

where

$$\boldsymbol{\Delta}^{(\ell)} := \frac{\boldsymbol{\tau}_+ - \boldsymbol{\tau}_- - \lambda \widetilde{\boldsymbol{\tau}}_+^{(\ell)} + \lambda \widetilde{\boldsymbol{\tau}}_-^{(\ell)}}{4}.$$

Note that

$$\begin{aligned} \boldsymbol{I} - \eta \left(\lambda \text{diag}(\boldsymbol{r}) + \boldsymbol{D}^{-1} (\boldsymbol{K} + \lambda \boldsymbol{I}) \right) &= \boldsymbol{D}^{-1/2} \left(\boldsymbol{I} - \eta \left(\lambda \text{diag}(\boldsymbol{r}) + \boldsymbol{D}^{-1/2} (\boldsymbol{K} + \lambda \boldsymbol{I}) \boldsymbol{D}^{-1/2} \right) \right) \boldsymbol{D}^{1/2}, \\ &= \boldsymbol{D}^{-1/2} \boldsymbol{A} \boldsymbol{D}^{1/2}. \end{aligned}$$

This, combined with (14), implies

$$\boldsymbol{D}^{1/2}\left(\boldsymbol{w}^{(\ell+1)}-\boldsymbol{w}^*\right)=\boldsymbol{A}\underbrace{\boldsymbol{D}^{1/2}\left(\boldsymbol{w}^{(\ell)}-\boldsymbol{w}^*\right)}_{\text{denoted by }\boldsymbol{e}^{(\ell)}}+\underbrace{\boldsymbol{D}^{1/2}\eta\left((\boldsymbol{K}\boldsymbol{w}^*)\odot\boldsymbol{r}+\boldsymbol{\Delta}^{(\ell)}\right)}_{\text{denoted by }\boldsymbol{\epsilon}^{(\ell)}}.$$

We rewrite the above equation as $\boldsymbol{e}^{(\ell+1)}=\boldsymbol{A}\boldsymbol{e}^{(\ell)}+\boldsymbol{\epsilon}^{(\ell)}$, which with an argument of recursion gives

$$\boldsymbol{e}^{(\ell+1)}=\boldsymbol{A}^{\ell+1}\boldsymbol{e}^{(0)}+\sum_{j=0}^{\ell}\boldsymbol{A}^j\boldsymbol{\epsilon}^{(\ell-j)}. \tag{15}$$

By definition of $\boldsymbol{\epsilon}^{(\ell)}$, we have

$$\left\|\boldsymbol{\epsilon}^{(\ell)}\right\|_2\le\sqrt{N}\left\|\boldsymbol{\epsilon}^{(\ell)}\right\|_\infty\le\sqrt{N}\eta\left\|\mathrm{diag}(\boldsymbol{D}^{1/2})\right\|_\infty\left(\|\boldsymbol{K}\boldsymbol{w}^*\|_\infty\|\boldsymbol{r}\|_\infty+\left\|\boldsymbol{\Delta}^{(\ell)}\right\|_\infty\right).$$

Recall that (4) implies $\left\|\mathrm{diag}(\boldsymbol{D}^{1/2})\right\|_\infty\le\sqrt{N}$; Lemma 2 yields $\|\boldsymbol{K}\boldsymbol{w}^*\|_\infty\le B_y/\sqrt{\lambda_0}$; assumption (6) yields $\|\boldsymbol{r}\|_\infty\le\epsilon_{\text{flip}}/N$ and

$$\left\|\boldsymbol{\Delta}^{(\ell)}\right\|_\infty\le\frac{\|\boldsymbol{\tau}_+\|_\infty+\|\boldsymbol{\tau}_-\|_\infty+\lambda\left(\|\widetilde{\boldsymbol{\tau}}_+\|_\infty+\|\widetilde{\boldsymbol{\tau}}_-\|_\infty\right)}{4}\le\frac{\epsilon_{\text{sq}}+\lambda_0\widetilde{\epsilon}_{\text{sq}}}{2N}\le\frac{(1+\lambda_0)\left(\epsilon_{\text{sq}}+\widetilde{\epsilon}_{\text{sq}}\right)}{N}.$$

Consequently, we have

$$\left\|\boldsymbol{\epsilon}^{(\ell)}\right\|_2\le\eta\left(\frac{B_y}{\sqrt{\lambda_0}}\epsilon_{\text{flip}}+(1+\lambda_0)\left(\epsilon_{\text{sq}}+\widetilde{\epsilon}_{\text{sq}}\right)\right)\equiv B_\epsilon.$$

Therefore, for all ℓ, we obtain from (15) that

$$\left\|\boldsymbol{e}^{(\ell)}\right\|_2\le\|\boldsymbol{A}\|_{2\to2}^{\ell}\left\|\boldsymbol{e}^{(0)}\right\|_2+B_\epsilon\sum_{j=0}^{\ell-1}\|\boldsymbol{A}\|_{2\to2}^{j}\le\|\boldsymbol{A}\|_{2\to2}^{\ell}\left\|\boldsymbol{e}^{(0)}\right\|_2+\frac{B_\epsilon}{1-\|\boldsymbol{A}\|_{2\to2}},$$

where the last inequality follows from (9), i.e., $\|\boldsymbol{A}\|_{2\to2}\in(0,1)$. Note that by (4), we have

$$\left\|\boldsymbol{e}^{(\ell)}\right\|_2=\left\|\boldsymbol{D}^{1/2}\left(\boldsymbol{w}^{(\ell)}-\boldsymbol{w}^*\right)\right\|_2\ge\sqrt{\kappa_{\min}N}\left\|\boldsymbol{w}^{(\ell)}-\boldsymbol{w}^*\right\|_2,$$

and by Lemma 2, we have

$$\left\|\boldsymbol{e}^{(0)}\right\|_2=\left\|\boldsymbol{D}^{1/2}\left(\boldsymbol{w}^{(0)}-\boldsymbol{w}^*\right)\right\|_2=\left\|\boldsymbol{D}^{1/2}\boldsymbol{w}^*\right\|_2\le N\|\boldsymbol{w}^*\|_\infty\le\frac{\left(\frac{1}{\sqrt{\lambda_0}}+1\right)B_y}{\lambda_0}$$

which combining with the previous estimate yields (10). □

Theorem 4 (Entry-wise boundedness of $\boldsymbol{w}^{(\ell)}$)**.** *Suppose that the bounded data assumption holds with constants B_x and B_y. Let initial value $\boldsymbol{w}^{(0)}=\boldsymbol{0}$. Let $\epsilon_{\text{sq}},\widetilde{\epsilon}_{\text{sq}}\in(0,1)$ and $\epsilon_{\text{flip}}\in(0,c)$ for a constant $c\in(0,1)$. Let $\lambda_0>0$ and η satisfy (8). Let $\boldsymbol{w}^{(\ell)}$ be generated from inexact preconditioned Richardson iteration (5) with condition (6). Then for all $\ell\in\mathbb{N}$,*

$$\left\|\boldsymbol{w}^{(\ell)}\right\|_\infty\le B_w:=\frac{1}{\sqrt{N}}\left(\frac{\left(\frac{1}{\sqrt{\lambda_0}}+1\right)B_y}{\lambda_0\sqrt{\kappa_{\min}}}+\frac{\frac{B_y}{\sqrt{\lambda_0}}+2(1+\lambda_0)}{\lambda_0(1-c)\sqrt{\kappa_{\min}}}\right)+\frac{1}{N}\left(\frac{\left(\frac{1}{\sqrt{\lambda_0}}+1\right)B_y}{\lambda_0}\right). \tag{16}$$

Proof. We obtain from (10) in Theorem 3 that

$$\left\|\boldsymbol{w}^{(\ell)}-\boldsymbol{w}^*\right\|_\infty\le\left\|\boldsymbol{w}^{(\ell)}-\boldsymbol{w}^*\right\|_2\le\frac{1}{\sqrt{N}}\left(\|\boldsymbol{A}\|_{2\to2}^{\ell}\frac{\left(\frac{1}{\sqrt{\lambda_0}}+1\right)B_y}{\lambda_0\sqrt{\kappa_{\min}}}+\frac{\eta\left(\frac{B_y}{\sqrt{\lambda_0}}+2(1+\lambda_0)\right)}{(1-\|\boldsymbol{A}\|_{2\to2})\sqrt{\kappa_{\min}}}\right),$$

where we used assumptions of $\epsilon_{\text{flip}},\epsilon_{\text{sq}},\widetilde{\epsilon}_{\text{sq}}<1$. Furthermore, by (9) in Theorem 3, we apply the estimate of $\|\boldsymbol{A}\|_{2\to2}$ and get

$$\left\|\boldsymbol{w}^{(\ell)}-\boldsymbol{w}^*\right\|_\infty\le\frac{1}{\sqrt{N}}\left(\frac{\left(\frac{1}{\sqrt{\lambda_0}}+1\right)B_y}{\lambda_0\sqrt{\kappa_{\min}}}+\frac{\frac{B_y}{\sqrt{\lambda_0}}+2(1+\lambda_0)}{\lambda_0(1-c)\sqrt{\kappa_{\min}}}\right),$$

where we used the assumption of $\epsilon_{\text{flip}}<c$. The desired result (16) can be immediately obtained by the triangle inequality $\left\|\boldsymbol{w}^{(\ell)}\right\|_\infty\le\left\|\boldsymbol{w}^{(\ell)}-\boldsymbol{w}^*\right\|_\infty+\|\boldsymbol{w}^*\|_\infty$ and estimate of $\|\boldsymbol{w}^*\|_\infty$ established in Lemma 2. □

Proposition 5. *Suppose that assumptions of Theorem 4 hold and* $\boldsymbol{w}^* = [w_i^* : i \in [N]]$ *is the solution of linear system (2). Then for all* $\ell \in \mathbb{N}$*, it holds that*

$$\left|\sum_{i=1}^{N}\left(w_i^{(\ell)} - w_i^*\right)\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_{N+1})\right| \leq (1-\eta\lambda_0(1-c))^{\ell}\frac{\left(\frac{1}{\sqrt{\lambda_0}}+1\right)B_y}{\lambda_0\sqrt{\kappa_{\min}}} + \frac{\frac{B_y}{\sqrt{\lambda_0}}\epsilon_{\text{flip}} + (1+\lambda_0)(\epsilon_{\text{sq}} + \widetilde{\epsilon}_{\text{sq}})}{\lambda_0(1-c)\sqrt{\kappa_{\min}}}.$$

Proof. By the Cauchy-Schwarz inequality, we have

$$\left|\sum_{i=1}^{N}\left(w_i^{(\ell)} - w_i^*\right)\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_{N+1})\right| \leq \left\|\boldsymbol{w}^{(\ell)} - \boldsymbol{w}^*\right\|_2 \left(\sum_{i=1}^{N}\left(\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_{N+1})\right)^2\right)^{\frac{1}{2}} \leq \sqrt{N}\left\|\boldsymbol{w}^{(\ell)} - \boldsymbol{w}^*\right\|_2.$$

Moreover, it follows from (9) and (10) in Theorem 3 that

$$\sqrt{N}\left\|\boldsymbol{w}^{(\ell)} - \boldsymbol{w}^*\right\|_2 \leq (1-\eta\lambda_0(1-c))^{\ell}\frac{\left(\frac{1}{\sqrt{\lambda_0}}+1\right)B_y}{\lambda_0\sqrt{\kappa_{\min}}} + \frac{\frac{B_y}{\sqrt{\lambda_0}}\epsilon_{\text{flip}} + (1+\lambda_0)(\epsilon_{\text{sq}} + \widetilde{\epsilon}_{\text{sq}})}{\lambda_0(1-c)\sqrt{\kappa_{\min}}}.$$

The desired result immediately follows from the above estimates. □

B Approximation Power of Single-Layer ReLU Networks

In the construction of Transformers for in-context learning, we typically use the MLP layer (a single-layer ReLU neural network) to approximate necessary arithmetic operations. For general dimension-free approximation (which holds for arbitrary input dimensions), achieving a target accuracy of ϵ requires a network width that scales as $\mathcal{O}(\log(1/\epsilon)/\epsilon^2)$ (Bach, 2017; Bai et al., 2023). While broadly applicable, these existential guarantees are inherently non-constructive and suffer from a relatively loose dependence on the error tolerance. However, because the specific operations we need to approximate are strictly one-dimensional smooth functions, we can use continuous piecewise-linear splines to reduce this width requirement to $\mathcal{O}(1/\sqrt{\epsilon})$ (De Boor, 1978; DeVore et al., 2021; DeVore, 1998). Furthermore, unlike dimension-free approaches, spline-based methods provide an explicit mathematical recipe for constructing the exact neural parameters. Therefore, in this section, we adopt these constructive spline arguments to establish sharper, explicitly realizable network width bounds.

To explicitly control the approximation error, we present the following constructive lemma. This result follows directly from standard error analysis of piecewise linear interpolation and well-known ReLU representations of splines (De Boor, 1978; DeVore et al., 2021). We state it here in the form required for our theoretical constructions and omit the standard proof.

Lemma 6 (Constructive Approximation of Single-layer ReLU network)**.** *Let* $f \in C^2[M_1, M_2]$ *and let* $M_1 = x_0 < x_1 < \cdots < x_n = M_2$ *be an arbitrary partition of* $[M_1, M_2]$ *with* $n \in \mathbb{N}$*. We define the network parameters explicitly as follows:*

$$d = f(M_1), \quad a_s = 1, \quad b_s = -x_{s-1}, \quad \text{for } s = 1, \ldots, n,$$

$$c_s = \begin{cases} \dfrac{f(x_1) - f(x_0)}{x_1 - x_0}, & s = 1, \\ \dfrac{f(x_s) - f(x_{s-1})}{x_s - x_{s-1}} - \dfrac{f(x_{s-1}) - f(x_{s-2})}{x_{s-1} - x_{s-2}}, & s = 2, \ldots, n. \end{cases}$$

Then the neural network $\phi(x) = \sum_{s=1}^{n} c_s \mathrm{ReLU}(a_s x + b_s) + d$ *satisfies the uniform approximation bound*

$$\sup_{x \in [M_1, M_2]} |f(x) - \phi(x)| \leq \frac{1}{8}\max_{k \in [n]}\left((x_k - x_{k-1})^2 \sup_{x \in [x_{k-1}, x_k]} |f''(x)|\right).$$

We now apply the general bound from Lemma 6 to derive explicit network widths for approximating specific one-dimensional smooth functions. Because our transformer blocks rely on these functions to compute intermediate variables, we formulate their approximation bounds as direct corollaries below.

Corollary 7 (Approximation of x^2 on $[-\delta, \delta]$)**.** *Let $\delta > 0$ and target accuracy $\epsilon > 0$. If the network width $n \in \mathbb{N}$ satisfies $n \geq \delta/\sqrt{\epsilon}$, then there exist parameters $a_s, b_s, c_s \in \mathbb{R}$ for $s \in [n]$, and $d \in \mathbb{R}$, such that the neural network $\phi(x) = \sum_{s=1}^{n} c_s \mathrm{ReLU}(a_s x + b_s) + d$ satisfies the uniform bound $\sup_{x \in [-\delta,\delta]} |x^2 - \phi(x)| \leq \epsilon$.*

Proof. We apply Lemma 6 using a uniform partition $x_k = -\delta + 2\delta k/n$. The interval width is constantly $2\delta/n$. By assumption on n and the fact of $f''(x) = 2$, we have

$$\frac{1}{8}(x_k - x_{k-1})^2 \sup_{x \in [x_{k-1}, x_k]} |f''(x)| = \frac{1}{8}(4\delta^2/n^2)(2) = \delta^2/n^2 \leq \epsilon.$$

□

Corollary 8 (Approximation of $x/(1-x)$ on $[0, \delta]$)**.** *Let $\delta \in (0, 1)$ and target accuracy $\epsilon \in (0, 1)$. If the network width $n \in \mathbb{N}$ satisfies*

$$n \geq \frac{2\left((1-\delta)^{-1/2} - 1\right)}{\sqrt{\epsilon}} = \mathcal{O}(1/\sqrt{\epsilon(1-\delta)}),$$

then there exist parameters $a_s, b_s, c_s \in \mathbb{R}$ for $s \in [n]$, and $d \in \mathbb{R}$, such that the neural network $\phi(x) = \sum_{s=1}^{n} c_s \mathrm{ReLU}(a_s x + b_s) + d$ satisfies $\sup_{x \in [0,\delta]} \left| \frac{x}{1-x} - \phi(x) \right| \leq \epsilon$.

Proof. Apply Lemma 6 to $f(x) = x/(1-x)$ on $[0, \delta]$. Following de Boor's equidistribution principle (De Boor, 1978), we select the adaptive grid nodes $x_k = 1 - y_k^{-2}$ where $y_k = 1 + \frac{k}{n}((1-\delta)^{-1/2} - 1)$. Then it follows from the assumption on n that

$$\frac{1}{8}(x_k - x_{k-1})^2 \sup_{x \in [x_{k-1}, x_k]} |f''(x)| \leq \frac{4}{n^2}\left((1-\delta)^{-1/2} - 1\right)^2 \leq \epsilon.$$

□

Corollary 9 (Approximation of $1/x$ on $[\delta, 1]$)**.** *Let $\delta \in (0, 1)$ and target accuracy $\epsilon \in (0, 1)$. If the network width $n \in \mathbb{N}$ satisfies*

$$n \geq 3/\sqrt{\delta\epsilon} = \mathcal{O}(1/\sqrt{\delta\epsilon}),$$

then there exist parameters $a_s, b_s, c_s \in \mathbb{R}$ for $s \in [n]$, and $d \in \mathbb{R}$, such that the neural network $\phi(x) = \sum_{s=1}^{n} c_s \mathrm{ReLU}(a_s x + b_s) + d$ satisfies $\sup_{x \in [\delta,1]} |1/x - \phi(x)| \leq \epsilon$.

Proof. Apply Lemma 6 to $f(x) = 1/x$ on $[\delta, 1]$. Following de Boor's equidistribution principle (De Boor, 1978), we select the adaptive grid nodes $x_k = y_k^{-2}$ where $y_k = \delta^{-1/2} + \frac{k}{n}(1 - \delta^{-1/2})$. Then by the assumption on n, we have

$$\frac{1}{8}(x_k - x_{k-1})^2 \sup_{x \in [x_{k-1}, x_k]} |f''(x)| \leq \frac{9}{\delta n^2} \leq \epsilon.$$

□

C Construction of Transformers

C.1 Overview of the Construction

We provide an overview of the transformer construction designed to implement the inexact preconditioned Richardson iteration. The construction includes three phases: read-in phase, iteration phase and read-out phase, where their detailed construction can be found in Appendices C.2, C.3, and C.4, respectively.

Read-in Phase. This phase serves as a preparatory step for the subsequent inexact Richardson iteration. Across three blocks, it approximates the preconditioner $\boldsymbol{D}^{-1}$ and $\boldsymbol{D}^{-1}\boldsymbol{y}$:

- One transformer block approximates $\alpha_i \approx \boldsymbol{D}_{ii}^{-1}$. Specifically, its single-head attention layer exactly computes $k_i \equiv 1/(1+\sum_{j'=1}^{N}\mathcal{K}(\boldsymbol{x}_i,\boldsymbol{x}_{j'}))$, and its MLP layer approximates $\alpha_i \approx k_i/(1-k_i) = \boldsymbol{D}_{ii}^{-1}$.
- One MLP-only transformer block zeros out α_{N+1}.
- One MLP-only transformer block approximates $\beta_i \approx \eta y_i \alpha_i$.

Iteration Phase. The core optimization executes the iterative updates on $\boldsymbol{w}$:

- One transformer block exactly computes $-\boldsymbol{D}^{-1}\boldsymbol{K}\boldsymbol{w}$. Its single-head attention layer computes $p_i \equiv -\boldsymbol{D}_{ii}^{-1}\sum_{j=1}^{N}\mathcal{K}(\boldsymbol{x}_i,\boldsymbol{x}_j)w_j$, while its MLP layer zeros out p_{N+1}.
- One MLP-only transformer block implements the one-step inexact iteration update

$$\boldsymbol{w}^{(\ell+1)} = \boldsymbol{w}^{(\ell)} + \eta\left(\boldsymbol{D}^{-1}\boldsymbol{y} - \boldsymbol{D}^{-1}\left(\boldsymbol{K}+\lambda\boldsymbol{I}\right)\boldsymbol{w}^{(\ell)}\right) + \boldsymbol{e}^{(\ell)},$$

  where $\boldsymbol{e}^{(\ell)}$ is the approximation error vector. This block also clears the p_i coordinate to reset the cache for future iterations.

Read-out Phase. The final blocks extract the prediction for the test token:

- One transformer block computes the normalized prediction and its inverse scaling factor for the test token. Its single-head attention layer exactly computes $\widehat{p}_{N+1} \equiv k_{N+1}\sum_{j=1}^{N}\mathcal{K}(\boldsymbol{x}_{N+1},\boldsymbol{x}_j)w_j$, and its MLP layer approximates $\widehat{k}_{N+1} \approx 1/k_{N+1}$.
- One MLP-only transformer block approximates the product $\widehat{p}_{N+1}\widehat{k}_{N+1}$ and writes the result to the target coordinate as the approximation for the Bayes optimal prediction $\sum_{j=1}^{N} w_j^*\mathcal{K}(\boldsymbol{x}_{N+1},\boldsymbol{x}_j)$.

As detailed across the three phases above, each token $\boldsymbol{z}_i$ is explicitly augmented to track these intermediate coordinates, structured as follows

$$\boldsymbol{z}_i = [\boldsymbol{x}_i^\top, y_i, w_i, \|\boldsymbol{x}_i\|_2^2, k_i, \alpha_i, \beta_i, p_i \text{ (or } \widehat{p}_i), \widehat{k}_i, s_i, t_i, 1]^\top.$$

C.2 Read-in Phase

One transformer block approximates $\boldsymbol{D}^{-1}$. We first construct a single-head attention layer. Set $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V \in \mathbb{R}^{D\times D}$ such that

$$\boldsymbol{W}_Q\boldsymbol{z}_i = [\boldsymbol{x}_i/v; \|\boldsymbol{x}_i\|^2/v; -(1-s_i)/(2v); \boldsymbol{0}],\ \boldsymbol{W}_K\boldsymbol{z}_j = [\boldsymbol{x}_j/v; -(1-s_j)/(2v); \|\boldsymbol{x}_j\|^2/v; \boldsymbol{0}],\ \boldsymbol{W}_V\boldsymbol{z}_j = s_j\boldsymbol{e}_{d+4}.$$

Note that for $i,j \in \mathbb{Z}_{N+2}$,

$$\langle \boldsymbol{W}_Q\boldsymbol{z}_i, \boldsymbol{W}_K\boldsymbol{z}_j\rangle = \begin{cases} -\frac{\|\boldsymbol{x}_i-\boldsymbol{x}_j\|^2}{2v^2}, & \text{if } i,j\in[N+1],\\ 0, & \text{if } i=0 \text{ or } j=0,\end{cases} \qquad \boldsymbol{W}_V\boldsymbol{z}_j = \begin{cases}\boldsymbol{0}, & \text{if } j\in[N+1],\\ \boldsymbol{e}_{d+4}, & \text{if } j=0.\end{cases}$$

We mask out the test token, i.e., setting mask matrix $\boldsymbol{M}\in\mathbb{R}^{(N+2)\times(N+2)}$ to be

$$\boldsymbol{M}_{ij} = \begin{cases}-\infty, & \text{if } j = N+1,\\ 0, & \text{otherwise.}\end{cases}$$

It follows that for all $i \in [N]$,

$$\begin{aligned}
\widetilde{\boldsymbol{z}}_i &= \boldsymbol{z}_i + \sum_{j=0}^{N+1} \frac{\exp\left(\langle \boldsymbol{W}_Q \boldsymbol{z}_i, \boldsymbol{W}_K \boldsymbol{z}_j \rangle + \boldsymbol{M}_{ij}\right)}{\sum_{j'=0}^{N+1} \exp\left(\langle \boldsymbol{W}_Q \boldsymbol{z}_i, \boldsymbol{W}_K \boldsymbol{z}_{j'} \rangle + \boldsymbol{M}_{ij'}\right)} \boldsymbol{W}_V \boldsymbol{z}_j \\
&= \boldsymbol{z}_i + \sum_{j=0}^{N} \frac{\exp\left(\langle \boldsymbol{W}_Q \boldsymbol{z}_i, \boldsymbol{W}_K \boldsymbol{z}_j \rangle\right)}{\sum_{j'=0}^{N} \exp\left(\langle \boldsymbol{W}_Q \boldsymbol{z}_i, \boldsymbol{W}_K \boldsymbol{z}_{j'} \rangle\right)} \boldsymbol{W}_V \boldsymbol{z}_j \\
&= \boldsymbol{z}_i + \frac{\exp\left(\langle \boldsymbol{W}_Q \boldsymbol{z}_i, \boldsymbol{W}_K \boldsymbol{z}_0 \rangle\right)}{\sum_{j'=0}^{N} \exp\left(\langle \boldsymbol{W}_Q \boldsymbol{z}_i, \boldsymbol{W}_K \boldsymbol{z}_{j'} \rangle\right)} \boldsymbol{W}_V \boldsymbol{z}_0 + \sum_{j=1}^{N} \frac{\exp\left(\langle \boldsymbol{W}_Q \boldsymbol{z}_i, \boldsymbol{W}_K \boldsymbol{z}_j \rangle\right)}{\sum_{j'=0}^{N} \exp\left(\langle \boldsymbol{W}_Q \boldsymbol{z}_i, \boldsymbol{W}_K \boldsymbol{z}_{j'} \rangle\right)} \boldsymbol{W}_V \boldsymbol{z}_j \\
&= \boldsymbol{z}_i + \frac{1}{1 + \sum_{j'=1}^{N} \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_{j'})} \boldsymbol{e}_{d+4}
\end{aligned}$$

and when $i = 0$,

$$\begin{aligned}
\widetilde{\boldsymbol{z}}_0 &= \boldsymbol{z}_0 + \sum_{j=0}^{N+1} \frac{\exp\left(\langle \boldsymbol{W}_Q \boldsymbol{z}_0, \boldsymbol{W}_K \boldsymbol{z}_j \rangle + \boldsymbol{M}_{0j}\right)}{\sum_{j'=0}^{N+1} \exp\left(\langle \boldsymbol{W}_Q \boldsymbol{z}_0, \boldsymbol{W}_K \boldsymbol{z}_{j'} \rangle + \boldsymbol{M}_{0j'}\right)} \boldsymbol{W}_V \boldsymbol{z}_j \\
&= \boldsymbol{z}_0 + \frac{1}{\sum_{j'=0}^{N} \exp\left(\langle \boldsymbol{W}_Q \boldsymbol{z}_0, \boldsymbol{W}_K \boldsymbol{z}_{j'} \rangle\right)} \boldsymbol{e}_{d+4} = \boldsymbol{z}_0 + \frac{1}{1+N} \boldsymbol{e}_{d+4}
\end{aligned}$$

and when $i = N + 1$,

$$\begin{aligned}
\widetilde{\boldsymbol{z}}_{N+1} &= \boldsymbol{z}_{N+1} + \sum_{j=0}^{N+1} \frac{\exp\left(\langle \boldsymbol{W}_Q \boldsymbol{z}_{N+1}, \boldsymbol{W}_K \boldsymbol{z}_j \rangle + \boldsymbol{M}_{N+1,j}\right)}{\sum_{j'=0}^{N+1} \exp\left(\langle \boldsymbol{W}_Q \boldsymbol{z}_{N+1}, \boldsymbol{W}_K \boldsymbol{z}_{j'} \rangle + \boldsymbol{M}_{N+1,j'}\right)} \boldsymbol{W}_V \boldsymbol{z}_j \\
&= \boldsymbol{z}_{N+1} + \frac{1}{\sum_{j'=0}^{N} \exp\left(\langle \boldsymbol{W}_Q \boldsymbol{z}_{N+1}, \boldsymbol{W}_K \boldsymbol{z}_{j'} \rangle\right)} \boldsymbol{e}_{d+4} = \boldsymbol{z}_{N+1} + \frac{1}{1 + \sum_{j'=1}^{N} \mathcal{K}(\boldsymbol{x}_{N+1}, \boldsymbol{x}_{j'})} \boldsymbol{e}_{d+4}
\end{aligned}$$

Let

$$k_i := \begin{cases} \dfrac{1}{1+N}, & \text{if } i = 0, \\ \dfrac{1}{1 + \sum_{j'=1}^{N} \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_{j'})}, & \text{if } i \in [N+1]. \end{cases} \tag{17}$$

We remark that the value of k_0 is not important in the later analysis, and hence we replace this value by $*$. Then, after this attention layer, we have output[1]

$$\begin{bmatrix}
\boldsymbol{x}_0 & \boldsymbol{x}_1 & \boldsymbol{x}_2 & \dots & \boldsymbol{x}_N & \boldsymbol{x}_{N+1} \\
0 & y_1 & y_2 & \dots & y_N & 0 \\
0 & w_1 & w_2 & \dots & w_N & 0 \\
0 & \|\boldsymbol{x}_1\|^2 & \|\boldsymbol{x}_2\|^2 & \dots & \|\boldsymbol{x}_N\|^2 & \|\boldsymbol{x}_{N+1}\|^2 \\
\boxed{*} & \boxed{k_1} & \boxed{k_2} & \dots & \boxed{k_N} & \boxed{k_{N+1}} \\
\boldsymbol{0}_4 & \boldsymbol{0}_4 & \boldsymbol{0}_4 & \dots & \boldsymbol{0}_4 & \boldsymbol{0}_4 \\
s_0 & s_1 & s_2 & \dots & s_N & s_{N+1} \\
t_0 & t_1 & t_2 & \dots & t_N & t_{N+1} \\
1 & 1 & 1 & \dots & 1 & 1
\end{bmatrix}.$$

We next construct the MLP layer in this transformer block. Let $\epsilon_{\text{flip}} \in (0, 1)$. According to Corollary 8 with $\delta = 1/(1 + N\kappa_{\min})$, there exist $a_s, b_s, c_s \in \mathbb{R}$ for $s \in [n_{\text{flip}}]$ with $n_{\text{flip}} := \left\lceil \frac{2\left((1 - \frac{1}{1+N\kappa_{\min}})^{-1/2} - 1\right)}{\sqrt{\epsilon_{\text{flip}}/N}} \right\rceil$ such that

$$\sup_{x \in \left[0, \frac{1}{1+N\kappa_{\min}}\right]} \left| \frac{x}{1-x} - \phi_{\text{flip}}(x) \right| < \frac{\epsilon_{\text{flip}}}{N} \tag{18}$$

[1] Here, the entries wrapped in a box indicate the values that have been updated compared to the input.

where $\phi_{\text{flip}}(x) := \sum_{s=1}^{n_{\text{flip}}} c_s \text{ReLU}\left(a_s x + b_s\right)$. Set $\boldsymbol{W}_{\text{in}} \in \mathbb{R}^{n_{\text{flip}} \times D}$ and $\boldsymbol{W}_{\text{out}} \in \mathbb{R}^{D \times n_{\text{flip}}}$ such that

$$\boldsymbol{W}_{\text{in}} \boldsymbol{z}_i = \begin{bmatrix} a_1 k_i + b_1 \\ a_2 k_i + b_2 \\ \vdots \\ a_{n_{\text{flip}}} k_i + b_{n_{\text{flip}}} \end{bmatrix}, \quad [\boldsymbol{W}_{\text{out}}]_{j,:} = \begin{cases} [c_1, c_2, \ldots, c_{n_{\text{flip}}}], & \text{if } j = d+5, \\ \boldsymbol{0}, & \text{otherwise.} \end{cases}$$

For all $i \in \mathbb{Z}_{N+2}$, let $\alpha_i := \phi_{\text{flip}}(k_i)$. We ignore the estimate for $i = 0$ as this value is not essential. Therefore, after this MLP layer, we have output

$$\begin{bmatrix} \boldsymbol{x}_0 & \boldsymbol{x}_1 & \boldsymbol{x}_2 & \ldots & \boldsymbol{x}_N & \boldsymbol{x}_{N+1} \\ 0 & y_1 & y_2 & \ldots & y_N & 0 \\ 0 & w_1 & w_2 & \ldots & w_N & 0 \\ 0 & \|\boldsymbol{x}_1\|^2 & \|\boldsymbol{x}_2\|^2 & \ldots & \|\boldsymbol{x}_N\|^2 & \|\boldsymbol{x}_{N+1}\|^2 \\ * & k_1 & k_2 & \ldots & k_N & k_{N+1} \\ \boxed{*} & \boxed{\alpha_1} & \boxed{\alpha_2} & \ldots & \boxed{\alpha_N} & \boxed{\alpha_{N+1}} \\ \boldsymbol{0}_3 & \boldsymbol{0}_3 & \boldsymbol{0}_3 & \ldots & \boldsymbol{0}_3 & \boldsymbol{0}_3 \\ s_0 & s_1 & s_2 & \ldots & s_N & s_{N+1} \\ t_0 & t_1 & t_2 & \ldots & t_N & t_{N+1} \\ 1 & 1 & 1 & \ldots & 1 & 1 \end{bmatrix}.$$

As a remark at the end of this block, we let

$$r_i := \alpha_i - \frac{k_i}{1 - k_i}, \qquad i \in [N+1]. \tag{19}$$

For $i \in [N]$, this agrees with $r_i = \alpha_i - \boldsymbol{D}_{ii}^{-1}$. Since $0 < k_i < \frac{1}{1+N\kappa_{\min}}$, we obtain from (18) that

$$|r_i| = \left| \frac{k_i}{1 - k_i} - \alpha_i \right| < \frac{\epsilon_{\text{flip}}}{N}, \quad i \in [N+1]. \tag{20}$$

One MLP-only transformer block zeros out α_{N+1}. It follows from (20) and the triangle inequality that for all $i \in [N+1]$,

$$|\alpha_i| < \frac{1}{\sum_{j'=1}^{N} \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_{j'})} + \frac{\epsilon_{\text{flip}}}{N} < \frac{1}{N\kappa_{\min}} + \frac{1}{N} \equiv B_\alpha \tag{21}$$

Then we set $\boldsymbol{W}_{\text{in}} \in \mathbb{R}^{2 \times D}$ and $\boldsymbol{W}_{\text{out}} \in \mathbb{R}^{D \times 2}$ such that

$$\boldsymbol{W}_{\text{in}} \boldsymbol{z}_i = \begin{bmatrix} -\alpha_i - B_\alpha(1 - t_i) \\ \alpha_i - B_\alpha(1 - t_i) \end{bmatrix}, \quad [\boldsymbol{W}_{\text{out}}]_{j,:} = \begin{cases} [1, -1], & \text{if } j = d+5, \\ \boldsymbol{0}, & \text{otherwise.} \end{cases}$$

Recalling the definition of t_i, when $i \in [N]$, we have $t_i = 0$, and hence

$$\boldsymbol{z}_i + \boldsymbol{W}_{\text{out}} \text{ReLU}(\boldsymbol{W}_{\text{in}} \boldsymbol{z}_i) = \boldsymbol{z}_i + \text{ReLU}(-\alpha_i - B_\alpha)\boldsymbol{e}_{d+5} - \text{ReLU}(\alpha_i - B_\alpha)\boldsymbol{e}_{d+5} = \boldsymbol{z}_i;$$

moreover, when $i = N+1$, we have $t_{N+1} = 1$, which leads to

$$\boldsymbol{z}_{N+1} + \boldsymbol{W}_{\text{out}} \text{ReLU}(\boldsymbol{W}_{\text{in}} \boldsymbol{z}_{N+1}) = \boldsymbol{z}_{N+1} + \text{ReLU}(-\alpha_{N+1})\boldsymbol{e}_{d+5} - \text{ReLU}(\alpha_{N+1})\boldsymbol{e}_{d+5} = \boldsymbol{z}_{N+1} - \alpha_{N+1}\boldsymbol{e}_{d+5}.$$

So, after this MLP layer, we have output

$$\begin{bmatrix} \boldsymbol{x}_0 & \boldsymbol{x}_1 & \boldsymbol{x}_2 & \ldots & \boldsymbol{x}_N & \boldsymbol{x}_{N+1} \\ 0 & y_1 & y_2 & \ldots & y_N & 0 \\ 0 & w_1 & w_2 & \ldots & w_N & 0 \\ 0 & \|\boldsymbol{x}_1\|^2 & \|\boldsymbol{x}_2\|^2 & \ldots & \|\boldsymbol{x}_N\|^2 & \|\boldsymbol{x}_{N+1}\|^2 \\ * & k_1 & k_2 & \ldots & k_N & k_{N+1} \\ * & \alpha_1 & \alpha_2 & \ldots & \alpha_N & \boxed{0} \\ \boldsymbol{0}_3 & \boldsymbol{0}_3 & \boldsymbol{0}_3 & \ldots & \boldsymbol{0}_3 & \boldsymbol{0}_3 \\ s_0 & s_1 & s_2 & \ldots & s_N & s_{N+1} \\ t_0 & t_1 & t_2 & \ldots & t_N & t_{N+1} \\ 1 & 1 & 1 & \ldots & 1 & 1 \end{bmatrix}.$$

One MLP-only transformer block approximates $\eta y_i \alpha_i$. Let $\epsilon_{\text{sq}} \in (0,1)$. According to Corollary 7 with $\delta = B_y + B_\alpha$, there exist $a_s, b_s, c_s \in \mathbb{R}$ for $s \in [n_{\text{sq}}]$ with $n_{\text{sq}} := \lceil \frac{B_y + B_\alpha}{\sqrt{\epsilon_{\text{sq}}/N}} \rceil$, and $d_{\text{sq}} \in \mathbb{R}$, such that

$$\sup_{|x| \le B_y + B_\alpha} \left| x^2 - \phi_{\text{sq}}(x) \right| \le \frac{\epsilon_{\text{sq}}}{N}, \tag{22}$$

where $\phi_{\text{sq}}(x) := d_{\text{sq}} + \sum_{s=1}^{n_{\text{sq}}} c_s \text{ReLU}\left(a_s x + b_s\right)$. We set $\boldsymbol{W}_{\text{in}} \in \mathbb{R}^{2n_{\text{sq}} \times D}$ and $\boldsymbol{W}_{\text{out}} \in \mathbb{R}^{D \times 2n_{\text{sq}}}$ such that

$$\boldsymbol{W}_{\text{in}} \boldsymbol{z}_i = \begin{bmatrix} a_1 (y_i + \alpha_i) + b_1 \\ \vdots \\ a_{n_{\text{sq}}} (y_i + \alpha_i) + b_{n_{\text{sq}}} \\ a_1 (y_i - \alpha_i) + b_1 \\ \vdots \\ a_{n_{\text{sq}}} (y_i - \alpha_i) + b_{n_{\text{sq}}} \end{bmatrix}, \quad [\boldsymbol{W}_{\text{out}}]_{j,:} = \begin{cases} \frac{\eta}{4}[c_1, \ldots, c_{n_{\text{sq}}}, -c_1, \ldots, -c_{n_{\text{sq}}}], & \text{if } j = d+6, \\ \boldsymbol{0}, & \text{otherwise.} \end{cases}$$

It turns out that for $i \in \mathbb{Z}_{N+2}$,

$$\begin{aligned} \beta_i \equiv [\boldsymbol{W}_{\text{out}} \text{ReLU}(\boldsymbol{W}_{\text{in}} \boldsymbol{Z})]_{d+6,i} &= \frac{\eta}{4} \sum_{s=1}^{n_{\text{sq}}} c_s \text{ReLU}\left(a_s (y_i + \alpha_i) + b_s\right) - \frac{\eta}{4} \sum_{s=1}^{n_{\text{sq}}} c_s \text{ReLU}\left(a_s (y_i - \alpha_i) + b_s\right) \\ &= \frac{\eta}{4} \left(\phi_{\text{sq}}(y_i + \alpha_i) - \phi_{\text{sq}}(y_i - \alpha_i)\right). \end{aligned}$$

Since $y_{N+1} = 0$ and $\alpha_{N+1} = 0$, we have $\beta_{N+1} = \frac{\eta}{4}\left(\phi_{\text{sq}}(0) - \phi_{\text{sq}}(0)\right) = 0$. So, after this MLP layer, we have output

$$\begin{bmatrix} \boldsymbol{x}_0 & \boldsymbol{x}_1 & \boldsymbol{x}_2 & \ldots & \boldsymbol{x}_N & \boldsymbol{x}_{N+1} \\ 0 & y_1 & y_2 & \ldots & y_N & 0 \\ 0 & w_1 & w_2 & \ldots & w_N & 0 \\ 0 & \|\boldsymbol{x}_1\|^2 & \|\boldsymbol{x}_2\|^2 & \ldots & \|\boldsymbol{x}_N\|^2 & \|\boldsymbol{x}_{N+1}\|^2 \\ * & k_1 & k_2 & \ldots & k_N & k_{N+1} \\ * & \alpha_1 & \alpha_2 & \ldots & \alpha_N & 0 \\ \boxed{*} & \boxed{\beta_1} & \boxed{\beta_2} & \ldots & \boxed{\beta_N} & \boxed{0} \\ \boldsymbol{0}_2 & \boldsymbol{0}_2 & \boldsymbol{0}_2 & \ldots & \boldsymbol{0}_2 & \boldsymbol{0}_2 \\ s_0 & s_1 & s_2 & \ldots & s_N & s_{N+1} \\ t_0 & t_1 & t_2 & \ldots & t_N & t_{N+1} \\ 1 & 1 & 1 & \ldots & 1 & 1 \end{bmatrix}.$$

Moreover, when $i \in [N]$, let

$$\tau_{+,i} := \phi_{\text{sq}}\left(y_i + \alpha_i\right) - \left(y_i + \alpha_i\right)^2, \quad \tau_{-,i} := \phi_{\text{sq}}\left(y_i - \alpha_i\right) - \left(y_i - \alpha_i\right)^2.$$

Then we can rewrite β_i as

$$\beta_i = \frac{\eta}{4}\left((y_i + \alpha_i)^2 + \tau_{+,i} - (y_i - \alpha_i)^2 - \tau_{-,i}\right) = \eta y_i \alpha_i + \frac{\eta}{4}\left(\tau_{+,i} - \tau_{-,i}\right). \tag{23}$$

Noting that $|y_i \pm \alpha_i| \le |y_i| + |\alpha_i| \le B_y + B_\alpha$, and by (22), we have

$$|\tau_{+,i}| \le \frac{\epsilon_{\text{sq}}}{N}, \quad |\tau_{-,i}| \le \frac{\epsilon_{\text{sq}}}{N}. \tag{24}$$

C.3 Iteration Phase

One transformer block exactly computes $-\boldsymbol{D}^{-1}\boldsymbol{K}\boldsymbol{w}$ and zeros out the corresponding feature for the test token. We first construct a single-head attention layer. Set σ to be Softmax, and $H = 1$. Set $\boldsymbol{W}_Q$, $\boldsymbol{W}_K$ and $\boldsymbol{W}_V$ such that

$$\boldsymbol{W}_Q \boldsymbol{z}_i = \frac{1}{v}[\boldsymbol{x}_i; \|\boldsymbol{x}_i\|^2; -1/2; \boldsymbol{0}], \quad \boldsymbol{W}_K \boldsymbol{z}_j = \frac{1}{v}[\boldsymbol{x}_j; -1/2; \|\boldsymbol{x}_j\|^2; \boldsymbol{0}], \quad \boldsymbol{W}_V \boldsymbol{z}_j = -w_j \boldsymbol{e}_{d+7}$$

Note that $\langle \boldsymbol{W}_Q \boldsymbol{z}_i, \boldsymbol{W}_K \boldsymbol{z}_j \rangle = -\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2v^2}$ for $i, j \in \mathbb{Z}_{N+2}$. We mask out the dummy token and the test token, i.e., setting mask matrix $\boldsymbol{M} \in \mathbb{R}^{(N+2)\times(N+2)}$ to be

$$\boldsymbol{M}_{ij} = \begin{cases} -\infty, & \text{if } j = 0 \text{ or } j = N+1, \\ 0, & \text{otherwise.} \end{cases}$$

It follows that for all $i \in \mathbb{Z}_{N+2}$,

$$\sum_{j=0}^{N+1} \frac{\exp\left(\langle \boldsymbol{W}_Q \boldsymbol{z}_i, \boldsymbol{W}_K \boldsymbol{z}_j \rangle + \boldsymbol{M}_{ij}\right)}{\sum_{j'=0}^{N+1} \exp\left(\langle \boldsymbol{W}_Q \boldsymbol{z}_i, \boldsymbol{W}_K \boldsymbol{z}_{j'} \rangle + \boldsymbol{M}_{ij'}\right)} \boldsymbol{W}_V \boldsymbol{z}_j = -\sum_{j=1}^{N} \frac{\exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2v^2}\right) w_j}{\sum_{j'=1}^{N} \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_{j'}\|^2}{2v^2}\right)} \boldsymbol{e}_{d+7} = p_i \boldsymbol{e}_{d+7}$$

where we denote

$$p_i := -\sum_{j=1}^{N} \frac{\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) w_j}{\sum_{j'=1}^{N} \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_{j'})}. \tag{25}$$

Again, the value of p_0 is not important, hence we will hide it as $*$. So, after this attention layer, we have output

$$\begin{bmatrix} \boldsymbol{x}_0 & \boldsymbol{x}_1 & \boldsymbol{x}_2 & \dots & \boldsymbol{x}_N & \boldsymbol{x}_{N+1} \\ 0 & y_1 & y_2 & \dots & y_N & 0 \\ 0 & w_1 & w_2 & \dots & w_N & 0 \\ 0 & \|\boldsymbol{x}_1\|^2 & \|\boldsymbol{x}_2\|^2 & \dots & \|\boldsymbol{x}_N\|^2 & \|\boldsymbol{x}_{N+1}\|^2 \\ * & k_1 & k_2 & \dots & k_N & k_{N+1} \\ * & \alpha_1 & \alpha_2 & \dots & \alpha_N & 0 \\ * & \beta_1 & \beta_2 & \dots & \beta_N & 0 \\ \boxed{*} & \boxed{p_1} & \boxed{p_2} & \dots & \boxed{p_N} & \boxed{p_{N+1}} \\ 0 & 0 & 0 & \dots & 0 & 0 \\ s_0 & s_1 & s_2 & \dots & s_N & s_{N+1} \\ t_0 & t_1 & t_2 & \dots & t_N & t_{N+1} \\ 1 & 1 & 1 & \dots & 1 & 1 \end{bmatrix}.$$

We next construct an MLP layer to explicitly zero out p_{N+1}. Even if p_{N+1} is currently zero due to the initialization of the weights $w_i = 0$, it may become non-zero in later iterations. As we will see in the construction of the next block, the network executes the inexact preconditioned Richardson iteration (5), which updates the weights w_i for $i \in [N]$. Therefore, we would still need this layer to enforce $p_{N+1} = 0$ across all iterations.

We fix B_w as defined in (16) and verify inductively that the constructed iterates satisfy $\left\|\boldsymbol{w}^{(\ell)}\right\|_\infty \le B_w$. The base case holds since $\boldsymbol{w}^{(0)} = \boldsymbol{0}$. We assume by induction that $\left\|\boldsymbol{w}^{(\ell)}\right\|_\infty \le B_w$, i.e., $|w_j|$ for $j \in [N]$ in the current step is bounded above by B_w. This guarantees that $|p_i|$ is also bounded by B_w, namely

$$|p_i| = \left| -\sum_{j=1}^{N} \frac{\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) w_j}{\sum_{j'=1}^{N} \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_{j'})} \right| \le \max_{j \in [N]} |w_j| \le B_w, \quad \forall i \in [N+1].$$

Then we set $\boldsymbol{W}_{\text{in}} \in \mathbb{R}^{2\times D}$ and $\boldsymbol{W}_{\text{out}} \in \mathbb{R}^{D\times 2}$ such that

$$\boldsymbol{W}_{\text{in}} \boldsymbol{z}_i = \begin{bmatrix} -p_i - B_w(1 - t_i) \\ p_i - B_w(1 - t_i) \end{bmatrix}, \quad [\boldsymbol{W}_{\text{out}}]_{j,:} = \begin{cases} [1, -1], & \text{if } j = d+7, \\ \boldsymbol{0}, & \text{otherwise.} \end{cases}$$

Recalling the definition of t_i, when $i \in [N]$, $t_i = 0$, and hence

$$\boldsymbol{z}_i + \boldsymbol{W}_{\text{out}} \text{ReLU}(\boldsymbol{W}_{\text{in}} \boldsymbol{z}_i) = \boldsymbol{z}_i + \text{ReLU}(-p_i - B_w) \boldsymbol{e}_{d+7} - \text{ReLU}(p_i - B_w) \boldsymbol{e}_{d+7} = \boldsymbol{z}_i,$$

moreover, when $i = N+1$, $t_{N+1} = 1$ gives

$$\boldsymbol{z}_{N+1} + \boldsymbol{W}_{\text{out}} \text{ReLU}(\boldsymbol{W}_{\text{in}} \boldsymbol{z}_{N+1}) = \boldsymbol{z}_{N+1} + \text{ReLU}(-p_{N+1}) \boldsymbol{e}_{d+7} - \text{ReLU}(p_{N+1}) \boldsymbol{e}_{d+7} = \boldsymbol{z}_{N+1} - p_{N+1} \boldsymbol{e}_{d+7}.$$

So, after this MLP layer, we have output

$$\begin{bmatrix} \boldsymbol{x}_0 & \boldsymbol{x}_1 & \boldsymbol{x}_2 & \dots & \boldsymbol{x}_N & \boldsymbol{x}_{N+1} \\ 0 & y_1 & y_2 & \dots & y_N & 0 \\ 0 & w_1 & w_2 & \dots & w_N & 0 \\ 0 & \|\boldsymbol{x}_1\|^2 & \|\boldsymbol{x}_2\|^2 & \dots & \|\boldsymbol{x}_N\|^2 & \|\boldsymbol{x}_{N+1}\|^2 \\ * & k_1 & k_2 & \dots & k_N & k_{N+1} \\ * & \alpha_1 & \alpha_2 & \dots & \alpha_N & 0 \\ * & \beta_1 & \beta_2 & \dots & \beta_N & 0 \\ * & p_1 & p_2 & \dots & p_N & \boxed{0} \\ 0 & 0 & 0 & \dots & 0 & 0 \\ s_0 & s_1 & s_2 & \dots & s_N & s_{N+1} \\ t_0 & t_1 & t_2 & \dots & t_N & t_{N+1} \\ 1 & 1 & 1 & \dots & 1 & 1 \end{bmatrix}.$$

An MLP-only transformer block implements one-step inexact precondition Richardson update. Let $\widetilde{\epsilon}_{\mathrm{sq}} \in (0,1)$. According to Corollary 7 with $\delta = B_w + B_\alpha$, there exist $\widetilde{a}_s, \widetilde{b}_s, \widetilde{c}_s \in \mathbb{R}$ for $s \in [\widetilde{n}_{\mathrm{sq}}]$ with $\widetilde{n}_{\mathrm{sq}} := \lceil \frac{B_w+B_\alpha}{\sqrt{\widetilde{\epsilon}_{\mathrm{sq}}/N^2}} \rceil$, and $\widetilde{d}_{\mathrm{sq}} \in \mathbb{R}$, such that

$$\sup_{|x| \le B_w + B_\alpha} \left| x^2 - \widetilde{\phi}_{\mathrm{sq}}(x) \right| \le \frac{\widetilde{\epsilon}_{\mathrm{sq}}}{N^2}, \tag{26}$$

where $\widetilde{\phi}_{\mathrm{sq}}(x) := \widetilde{d}_{\mathrm{sq}} + \sum_{s=1}^{\widetilde{n}_{\mathrm{sq}}} \widetilde{c}_s \mathrm{ReLU}\left(\widetilde{a}_s x + \widetilde{b}_s\right)$. Set $\boldsymbol{W}_{\mathrm{in}} \in \mathbb{R}^{(2\widetilde{n}_{\mathrm{sq}}+4)\times D}$ such that

$$\boldsymbol{W}_{\mathrm{in}} \boldsymbol{z}_i = \begin{bmatrix} \widetilde{a}_1 (w_i + \alpha_i) + \widetilde{b}_1 \\ \vdots \\ \widetilde{a}_{\widetilde{n}_{\mathrm{sq}}} (w_i + \alpha_i) + \widetilde{b}_{\widetilde{n}_{\mathrm{sq}}} \\ \widetilde{a}_1 (w_i - \alpha_i) + \widetilde{b}_1 \\ \vdots \\ \widetilde{a}_{\widetilde{n}_{\mathrm{sq}}} (w_i - \alpha_i) + \widetilde{b}_{\widetilde{n}_{\mathrm{sq}}} \\ \beta_i \\ -\beta_i \\ p_i \\ -p_i \end{bmatrix}$$

and set $\boldsymbol{W}_{\mathrm{out}} \in \mathbb{R}^{D \times (2\widetilde{n}_{\mathrm{sq}}+4)}$ such that

$$[\boldsymbol{W}_{\mathrm{out}}]_{j,:} = \begin{cases} \left[-\frac{\eta\lambda\widetilde{c}_1}{4}, \dots, -\frac{\eta\lambda\widetilde{c}_{\widetilde{n}_{\mathrm{sq}}}}{4}, \frac{\eta\lambda\widetilde{c}_1}{4}, \dots, \frac{\eta\lambda\widetilde{c}_{\widetilde{n}_{\mathrm{sq}}}}{4}, 1, -1, \eta, -\eta\right], & \text{if } j = d+2, \\ [0, 0, \dots, 0, -1, 1], & \text{if } j = d+7, \\ \boldsymbol{0}, & \text{otherwise.} \end{cases}$$

It turns out that for $i \in \mathbb{Z}_{N+2}$,

$$\begin{aligned} & [\boldsymbol{W}_{\mathrm{out}} \mathrm{ReLU}(\boldsymbol{W}_{\mathrm{in}} \boldsymbol{Z})]_{d+2,i} \\ = & -\frac{\eta\lambda}{4} \left(\sum_{s=1}^{\widetilde{n}_{\mathrm{sq}}} \widetilde{c}_s \mathrm{ReLU}\left(\widetilde{a}_s (w_i + \alpha_i) + \widetilde{b}_s\right) - \sum_{s=1}^{\widetilde{n}_{\mathrm{sq}}} \widetilde{c}_s \mathrm{ReLU}\left(\widetilde{a}_s (w_i - \alpha_i) + \widetilde{b}_s\right) \right) \\ & + (\mathrm{ReLU}(\beta_i) - \mathrm{ReLU}(-\beta_i)) + \eta (\mathrm{ReLU}(p_i) - \mathrm{ReLU}(-p_i)) \\ = & -\frac{\eta\lambda}{4} \left(\widetilde{\phi}_{\mathrm{sq}}(w_i + \alpha_i) - \widetilde{\phi}_{\mathrm{sq}}(w_i - \alpha_i) \right) + \beta_i + \eta p_i \end{aligned} \tag{27}$$

and

$$[\boldsymbol{W}_{\mathrm{out}} \mathrm{ReLU}(\boldsymbol{W}_{\mathrm{in}} \boldsymbol{Z})]_{d+7,i} = -\mathrm{ReLU}(p_i) + \mathrm{ReLU}(-p_i) = -p_i.$$

We first look at the update at the $(d+2)$-th row.

- When $i = N+1$, note that $\beta_{N+1} = 0$, $w_{N+1} = 0$, $\alpha_{N+1} = 0$ and $p_{N+1} = 0$. Hence (27) with $i = N+1$ gives

$$\Delta w_{N+1} \equiv [\boldsymbol{W}_{\text{out}}\text{ReLU}(\boldsymbol{W}_{\text{in}}\boldsymbol{Z})]_{d+2,N+1} = -\frac{\eta\lambda}{4}\left(\widetilde{\phi}_{\text{sq}}(0) - \widetilde{\phi}_{\text{sq}}(0)\right) = 0.$$

- Now we consider $i \in [N]$. Let

$$\widetilde{\tau}_{+,i} := \widetilde{\phi}_{\text{sq}}(w_i + \alpha_i) - (w_i + \alpha_i)^2, \quad \widetilde{\tau}_{-,i} := \widetilde{\phi}_{\text{sq}}(w_i - \alpha_i) - (w_i - \alpha_i)^2.$$

Note that $|w_i \pm \alpha_i| \le |w_i| + |\alpha_i| \le B_w + B_\alpha$, which with (26) implies

$$|\widetilde{\tau}_{+,i}| \le \frac{\widetilde{\epsilon}_{\text{sq}}}{N^2}, \quad |\widetilde{\tau}_{-,i}| \le \frac{\widetilde{\epsilon}_{\text{sq}}}{N^2}. \tag{28}$$

Then with the above notations and using forms (19), (23), (25) for r_i, β_i and p_i, we proceed from (27) to obtain that for $i \in [N]$,

$$\begin{aligned}
\Delta w_i &\equiv [\boldsymbol{W}_{\text{out}}\text{ReLU}(\boldsymbol{W}_{\text{in}}\boldsymbol{Z})]_{d+2,i} \\
&= -\frac{\eta\lambda}{4}\left((w_i+\alpha_i)^2 + \widetilde{\tau}_{+,i} - (w_i-\alpha_i)^2 - \widetilde{\tau}_{-,i}\right) + \beta_i + \eta p_i \\
&= \eta(y_i - \lambda w_i)\alpha_i + \eta p_i + \frac{\eta}{4}(\tau_{+,i} - \tau_{-,i}) - \frac{\eta\lambda}{4}(\widetilde{\tau}_{+,i} - \widetilde{\tau}_{-,i}) \\
&= \eta(y_i - \lambda w_i)\boldsymbol{D}_{ii}^{-1} - \eta\boldsymbol{D}_{ii}^{-1}\sum_{j=1}^{N}\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)w_j + \frac{\eta}{4}(\tau_{+,i} - \tau_{-,i}) - \frac{\eta\lambda}{4}(\widetilde{\tau}_{+,i} - \widetilde{\tau}_{-,i}) + \eta(y_i - \lambda w_i)r_i
\end{aligned}$$

Therefore, with the residual connection, we have updated at the $(d+2)$-th row

$$w_i \leftarrow w_i + \Delta w_i.$$

Moreover, the update at the $(d+7)$-th row is

$$p_i \leftarrow p_i - p_i = 0.$$

After this MLP layer, we have output

$$\begin{bmatrix}
\boldsymbol{x}_0 & \boldsymbol{x}_1 & \boldsymbol{x}_2 & \dots & \boldsymbol{x}_N & \boldsymbol{x}_{N+1} \\
0 & y_1 & y_2 & \dots & y_N & 0 \\
\boxed{*} & \boxed{w_1 + \Delta w_1} & \boxed{w_2 + \Delta w_2} & \dots & \boxed{w_N + \Delta w_N} & \boxed{0} \\
0 & \|\boldsymbol{x}_1\|^2 & \|\boldsymbol{x}_2\|^2 & \dots & \|\boldsymbol{x}_N\|^2 & \|\boldsymbol{x}_{N+1}\|^2 \\
* & k_1 & k_2 & \dots & k_N & k_{N+1} \\
* & \alpha_1 & \alpha_2 & \dots & \alpha_N & 0 \\
* & \beta_1 & \beta_2 & \dots & \beta_N & 0 \\
\boxed{0} & \boxed{0} & \boxed{0} & \dots & \boxed{0} & \boxed{0} \\
0 & 0 & 0 & \dots & 0 & 0 \\
s_0 & s_1 & s_2 & \dots & s_N & s_{N+1} \\
t_0 & t_1 & t_2 & \dots & t_N & t_{N+1} \\
1 & 1 & 1 & \dots & 1 & 1
\end{bmatrix}$$

We notice that the constructed transformer implements the following update

$$w_i \leftarrow w_i + \eta\left((y_i - \lambda w_i)\boldsymbol{D}_{ii}^{-1} - \boldsymbol{D}_{ii}^{-1}\sum_{j=1}^{N}\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)w_j\right) + \eta\left((y_i - \lambda w_i)r_i + \frac{\tau_{+,i} - \tau_{-,i} - \lambda\widetilde{\tau}_{+,i} + \lambda\widetilde{\tau}_{-,i}}{4}\right).$$

In vector form, it is the inexact preconditioned Richardson iteration (5) with

$$\begin{aligned}
&\boldsymbol{r} := [r_i : i \in [N]], \quad \boldsymbol{\tau}_+ := [\tau_{+,i} : i \in [N]], \quad \boldsymbol{\tau}_- := [\tau_{-,i} : i \in [N]], \\
&\widetilde{\boldsymbol{\tau}}_+^{(\ell)} := [\widetilde{\tau}_{+,i}^{(\ell)} : i \in [N]], \quad \widetilde{\boldsymbol{\tau}}_-^{(\ell)} := [\widetilde{\tau}_{-,i}^{(\ell)} : i \in [N]].
\end{aligned}$$

Here, the error vectors $\boldsymbol{r}$, $\boldsymbol{\tau}_+$, and $\boldsymbol{\tau}_-$ do not depend on $\boldsymbol{w}^{(\ell)}$ and therefore remain unchanged throughout the iterations. On the other hand, $\widetilde{\boldsymbol{\tau}}_+^{(\ell)}$ and $\widetilde{\boldsymbol{\tau}}_-^{(\ell)}$ depend on $\boldsymbol{w}^{(\ell)}$, which is why they carry the index ℓ.

We verify the required bounds by induction, starting from $\boldsymbol{w}^{(0)} = \mathbf{0}$. The bounds on $\boldsymbol{r}$, $\boldsymbol{\tau}_+$, and $\boldsymbol{\tau}_-$ follow directly from (20) and (24). Moreover, the induction hypothesis $\left\|\boldsymbol{w}^{(\ell)}\right\|_\infty \le B_w$ implies (28) at iteration ℓ. Thus, all the conditions in (6) hold through iteration ℓ. The recursive estimate in the proof of Theorem 4 then gives $\left\|\boldsymbol{w}^{(\ell+1)}\right\|_\infty \le B_w$. This completes the induction.

We remark that the two blocks constructed in this phase implement a single step of the inexact preconditioned Richardson iteration (5). By sequentially repeating this construction, the transformer executes multiple iterations of the algorithm.

C.4 Read-out Phase

One transformer block computes the test token's normalized prediction and inverse scaling factor. We first construct a single-head attention layer. Set $\boldsymbol{W}_Q$, $\boldsymbol{W}_K$ and $\boldsymbol{W}_V$ such that

$$\boldsymbol{W}_Q \boldsymbol{z}_i = \frac{1}{v}[\boldsymbol{x}_i; \|\boldsymbol{x}_i\|^2; -1/2; \mathbf{0}], \quad \boldsymbol{W}_K \boldsymbol{z}_j = \frac{1}{v}[\boldsymbol{x}_j; -1/2; \|\boldsymbol{x}_j\|^2; \mathbf{0}], \quad \boldsymbol{W}_V \boldsymbol{z}_j = w_j \boldsymbol{e}_{d+7}.$$

Note that $\langle \boldsymbol{W}_Q \boldsymbol{z}_i, \boldsymbol{W}_K \boldsymbol{z}_j \rangle = -\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2v^2}$. We mask out the dummy token, i.e., setting mask matrix $\boldsymbol{M} \in \mathbb{R}^{(N+2)\times(N+2)}$ to be

$$\boldsymbol{M}_{ij} = \begin{cases} -\infty, & \text{if } j = 0, \\ 0, & \text{otherwise.} \end{cases}$$

It follows that for all $i \in [N+1]$,

$$\sum_{j=0}^{N+1} \frac{\exp\left(\langle \boldsymbol{W}_Q \boldsymbol{z}_i, \boldsymbol{W}_K \boldsymbol{z}_j \rangle + \boldsymbol{M}_{ij}\right)}{\sum_{j'=0}^{N+1} \exp\left(\langle \boldsymbol{W}_Q \boldsymbol{z}_i, \boldsymbol{W}_K \boldsymbol{z}_{j'} \rangle + \boldsymbol{M}_{ij'}\right)} \boldsymbol{W}_V \boldsymbol{z}_j = \sum_{j=1}^{N+1} \frac{\exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2v^2}\right) w_j}{\sum_{j'=1}^{N+1} \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_{j'}\|^2}{2v^2}\right)} \boldsymbol{e}_{d+7}$$

$$= \sum_{j=1}^{N+1} \frac{\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) w_j}{\sum_{j'=1}^{N+1} \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_{j'})} \boldsymbol{e}_{d+7} = \underbrace{\sum_{j=1}^{N} \frac{\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) w_j}{\sum_{j'=1}^{N+1} \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_{j'})}}_{\text{denoted by } \widehat{p}_i} \boldsymbol{e}_{d+7}$$

where the last equality uses $w_{N+1} = 0$. In particular, when $i = N+1$, the value is

$$\widehat{p}_{N+1} = \sum_{j=1}^{N} \frac{\mathcal{K}(\boldsymbol{x}_{N+1}, \boldsymbol{x}_j) w_j}{\sum_{j'=1}^{N+1} \mathcal{K}(\boldsymbol{x}_{N+1}, \boldsymbol{x}_{j'})} = \frac{1}{1 + \sum_{j'=1}^{N} \mathcal{K}(\boldsymbol{x}_{N+1}, \boldsymbol{x}_{j'})} \sum_{j=1}^{N} \mathcal{K}(\boldsymbol{x}_{N+1}, \boldsymbol{x}_j) w_j$$

$$= k_{N+1} \sum_{j=1}^{N} \mathcal{K}(\boldsymbol{x}_{N+1}, \boldsymbol{x}_j) w_j$$

where the last equation follows from definition (17) of k_{N+1}. Note that the $(d+7)$-th row of the input is a zero vector, and we only care about the feature $\widehat{p}_{N+1}$ for the test token (the features for the training tokens are hidden as $*$). Therefore, after this attention layer, the output is

$$\begin{bmatrix} \boldsymbol{x}_0 & \boldsymbol{x}_1 & \boldsymbol{x}_2 & \dots & \boldsymbol{x}_N & \boldsymbol{x}_{N+1} \\ 0 & y_1 & y_2 & \dots & y_N & 0 \\ * & w_1 & w_2 & \dots & w_N & 0 \\ 0 & \|\boldsymbol{x}_1\|^2 & \|\boldsymbol{x}_2\|^2 & \dots & \|\boldsymbol{x}_N\|^2 & \|\boldsymbol{x}_{N+1}\|^2 \\ * & k_1 & k_2 & \dots & k_N & k_{N+1} \\ * & \alpha_1 & \alpha_2 & \dots & \alpha_N & 0 \\ * & \beta_1 & \beta_2 & \dots & \beta_N & 0 \\ \boxed{*} & \boxed{*} & \boxed{*} & \dots & \boxed{*} & \boxed{\widehat{p}_{N+1}} \\ 0 & 0 & 0 & \dots & 0 & 0 \\ s_0 & s_1 & s_2 & \dots & s_N & s_{N+1} \\ t_0 & t_1 & t_2 & \dots & t_N & t_{N+1} \\ 1 & 1 & 1 & \dots & 1 & 1 \end{bmatrix}.$$

The MLP layer of this block approximates $1/k_{N+1}$. Let $\epsilon_{\text{inv}} \in (0,1)$. According to Corollary 9 with $\delta = 1/(N+1)$, there exist $a_s, b_s, c_s \in \mathbb{R}$ for $s \in [n_{\text{inv}}]$ with $n_{\text{inv}} := \lceil 3\sqrt{(N+1)/\epsilon_{\text{inv}}} \rceil$ and $d_{\text{inv}} \in \mathbb{R}$, such that

$$\sup_{x \in [\frac{1}{N+1}, 1]} \left| \frac{1}{x} - \phi_{\text{inv}}(x) \right| < \epsilon_{\text{inv}} \tag{29}$$

where $\phi_{\text{inv}}(x) := d_{\text{inv}} + \sum_{s=1}^{n_{\text{inv}}} c_s \text{ReLU}\left(a_s x + b_s\right)$. Set $\boldsymbol{W}_{\text{in}} \in \mathbb{R}^{(n_{\text{inv}}+1) \times D}$ and $\boldsymbol{W}_{\text{out}} \in \mathbb{R}^{D \times (n_{\text{inv}}+1)}$ such that

$$\boldsymbol{W}_{\text{in}} \boldsymbol{z}_i = \begin{bmatrix} a_1 k_i + b_1 \\ a_2 k_i + b_2 \\ \vdots \\ a_{n_{\text{inv}}} k_i + b_{n_{\text{inv}}} \\ 1 \end{bmatrix}, \quad [\boldsymbol{W}_{\text{out}}]_{j,:} = \begin{cases} [c_1, c_2, \ldots, c_{n_{\text{inv}}}, d_{\text{inv}}], & \text{if } j = d+8, \\ \boldsymbol{0}, & \text{otherwise.} \end{cases}$$

Let $\widehat{k}_i := \phi_{\text{inv}}(k_i)$. It turns out that for $i \in \mathbb{Z}_{N+2}$, $[\boldsymbol{W}_{\text{out}} \text{ReLU}(\boldsymbol{W}_{\text{in}} \boldsymbol{Z})]_{d+8,i} = \widehat{k}_i$. In particular, we only need the case $i = N+1$, and the output of this MLP layer is

$$\begin{bmatrix} \boldsymbol{x}_0 & \boldsymbol{x}_1 & \boldsymbol{x}_2 & \ldots & \boldsymbol{x}_N & \boldsymbol{x}_{N+1} \\ 0 & y_1 & y_2 & \ldots & y_N & 0 \\ * & w_1 & w_2 & \ldots & w_N & 0 \\ 0 & \|\boldsymbol{x}_1\|^2 & \|\boldsymbol{x}_2\|^2 & \ldots & \|\boldsymbol{x}_N\|^2 & \|\boldsymbol{x}_{N+1}\|^2 \\ * & k_1 & k_2 & \ldots & k_N & k_{N+1} \\ * & \alpha_1 & \alpha_2 & \ldots & \alpha_N & 0 \\ * & \beta_1 & \beta_2 & \ldots & \beta_N & 0 \\ * & * & * & \ldots & * & \widehat{p}_{N+1} \\ \boxed{*} & \boxed{*} & \boxed{*} & \cdots & \boxed{*} & \boxed{\widehat{k}_{N+1}} \\ s_0 & s_1 & s_2 & \ldots & s_N & s_{N+1} \\ t_0 & t_1 & t_2 & \ldots & t_N & t_{N+1} \\ 1 & 1 & 1 & \ldots & 1 & 1 \end{bmatrix}.$$

Moreover, we remark that

$$\frac{1}{1+N} \leq k_i = \frac{1}{1 + \sum_{j'=1}^{N} \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)} \leq 1, \quad \forall i \in [N+1].$$

The above estimate with (29) implies

$$\left| \frac{1}{k_i} - \widehat{k}_i \right| < \epsilon_{\text{inv}}. \tag{30}$$

One MLP-only transformer block approximates the product $\widehat{k}_{N+1}\widehat{p}_{N+1}$. Let $\widehat{\epsilon}_{\text{sq}} \in (0,1)$. According to Corollary 7 with $\delta = B_w + 3$, there exist $\widehat{a}_s, \widehat{b}_s, \widehat{c}_s \in \mathbb{R}$ for $s \in [\widehat{n}_{\text{sq}}]$ with $\widehat{n}_{\text{sq}} := \lceil \frac{B_w+3}{\sqrt{\widehat{\epsilon}_{\text{sq}}/N}} \rceil$ and $\widehat{d}_{\text{sq}} \in \mathbb{R}$, such that

$$\sup_{|x| \leq B_w+3} \left| x^2 - \widehat{\phi}_{\text{sq}}(x) \right| \leq \frac{\widehat{\epsilon}_{\text{sq}}}{N}, \tag{31}$$

where $\widehat{\phi}_{\mathrm{sq}}(x) := \widehat{d}_{\mathrm{sq}} + \sum_{s=1}^{\widehat{n}_{\mathrm{sq}}} \widehat{c}_s \mathrm{ReLU}\left(\widehat{a}_s x + \widehat{b}_s\right)$. Set $\boldsymbol{W}_{\mathrm{in}} \in \mathbb{R}^{2\widehat{n}_{\mathrm{sq}} \times D}$ and $\boldsymbol{W}_{\mathrm{out}} \in \mathbb{R}^{D \times 2\widehat{n}_{\mathrm{sq}}}$ such that

$$\boldsymbol{W}_{\mathrm{in}} \boldsymbol{z}_i = \begin{bmatrix} \widehat{a}_1 \left(\widehat{k}_i/N + \widehat{p}_i\right) + \widehat{b}_1 \\ \widehat{a}_2 \left(\widehat{k}_i/N + \widehat{p}_i\right) + \widehat{b}_2 \\ \vdots \\ \widehat{a}_{\widehat{n}_{\mathrm{sq}}} \left(\widehat{k}_i/N + \widehat{p}_i\right) + \widehat{b}_{\widehat{n}_{\mathrm{sq}}} \\ \widehat{a}_1 \left(\widehat{k}_i/N - \widehat{p}_i\right) + \widehat{b}_1 \\ \widehat{a}_2 \left(\widehat{k}_i/N - \widehat{p}_i\right) + \widehat{b}_2 \\ \vdots \\ \widehat{a}_{\widehat{n}_{\mathrm{sq}}} \left(\widehat{k}_i/N - \widehat{p}_i\right) + \widehat{b}_{\widehat{n}_{\mathrm{sq}}} \end{bmatrix},$$

and

$$[\boldsymbol{W}_{\mathrm{out}}]_{j,:} = \begin{cases} \frac{N}{4}[\widehat{c}_1, \widehat{c}_2, \ldots, \widehat{c}_{\widehat{n}_{\mathrm{sq}}}, -\widehat{c}_1, -\widehat{c}_2, \ldots, -\widehat{c}_{\widehat{n}_{\mathrm{sq}}}], & \text{if } j = d+1, \\ \mathbf{0}, & \text{otherwise.} \end{cases}$$

Therefore, for $i \in \mathbb{Z}_{N+2}$,

$$o_i := [\boldsymbol{W}_{\mathrm{out}} \mathrm{ReLU}(\boldsymbol{W}_{\mathrm{in}} \boldsymbol{Z})]_{d+1,i} = \frac{N}{4}\left(\widehat{\phi}_{\mathrm{sq}}(\widehat{k}_i/N + \widehat{p}_i) - \widehat{\phi}_{\mathrm{sq}}(\widehat{k}_i/N - \widehat{p}_i)\right). \tag{32}$$

Thus, after this layer, we have output

$$\begin{bmatrix} \boldsymbol{x}_0 & \boldsymbol{x}_1 & \boldsymbol{x}_2 & \ldots & \boldsymbol{x}_N & \boldsymbol{x}_{N+1} \\ \boxed{*} & \boxed{y_1 + *} & \boxed{y_2 + *} & \ldots & \boxed{y_N + *} & \boxed{o_{N+1}} \\ * & w_1 & w_2 & \ldots & w_N & 0 \\ 0 & \|\boldsymbol{x}_1\|^2 & \|\boldsymbol{x}_2\|^2 & \ldots & \|\boldsymbol{x}_N\|^2 & \|\boldsymbol{x}_{N+1}\|^2 \\ * & k_1 & k_2 & \ldots & k_N & k_{N+1} \\ * & \alpha_1 & \alpha_2 & \ldots & \alpha_N & 0 \\ * & \beta_1 & \beta_2 & \ldots & \beta_N & 0 \\ * & * & * & \ldots & * & \widehat{p}_{N+1} \\ * & * & * & \cdots & * & \widehat{k}_{N+1} \\ s_0 & s_1 & s_2 & \ldots & s_N & s_{N+1} \\ t_0 & t_1 & t_2 & \ldots & t_N & t_{N+1} \\ 1 & 1 & 1 & \ldots & 1 & 1 \end{bmatrix}.$$

Moreover, when $i \in [N+1]$, let

$$\widehat{\tau}_{+,i} := \widehat{\phi}_{\mathrm{sq}}(\widehat{k}_i/N + \widehat{p}_i) - (\widehat{k}_i/N + \widehat{p}_i)^2, \quad \widehat{\tau}_{-,i} := \widehat{\phi}_{\mathrm{sq}}(\widehat{k}_i/N - \widehat{p}_i) - (\widehat{k}_i/N - \widehat{p}_i)^2.$$

It follows that o_i can be rewritten as

$$o_i = \frac{N}{4}\left((\widehat{k}_i/N + \widehat{p}_i)^2 - (\widehat{k}_i/N - \widehat{p}_i)^2 + \widehat{\tau}_{+,i} - \widehat{\tau}_{-,i}\right) = \widehat{k}_i \widehat{p}_i + \frac{N}{4}\left(\widehat{\tau}_{+,i} - \widehat{\tau}_{-,i}\right). \tag{33}$$

Recall that $|w_j|, j \in [N]$ are bounded by B_w, hence $\widehat{p}_i$ is also bounded by B_w. That is,

$$|\widehat{p}_i| = \left|\sum_{j=1}^{N} \frac{\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) w_j}{\sum_{j'=1}^{N+1} \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_{j'})}\right| \leq \max_{j \in [N]} |w_j| \leq B_w, \quad \forall i \in [N+1].$$

Furthermore, it follows from (30) that

$$\left|\frac{\widehat{k}_i}{N}\right| < \frac{1}{N}\left(\frac{1}{k_i} + \epsilon_{\mathrm{inv}}\right) = \frac{1}{N}\left(1 + \sum_{j'=1}^{N} \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) + \epsilon_{\mathrm{inv}}\right) < \frac{2+N}{N} \leq 3.$$

Therefore, $\left|\frac{\widehat{k}_i}{N} \pm \widehat{p}_i\right| \leq \left|\frac{\widehat{k}_i}{N}\right| + |\widehat{p}_i| < 3 + B_w$. Combining this with (31) implies

$$|\widehat{\tau}_{\pm,i}| < \widehat{\epsilon}_{\mathrm{sq}}/N. \tag{34}$$

C.5 Analysis

Proposition 10. *Suppose that assumptions of Theorem 4 hold and $\boldsymbol{w}^* = [w_i^* : i \in [N]]$ is the solution of linear system (2). Let* TF *denote the complete transformer network constructed by composing the read-in blocks from Appendix C.2, ℓ repetitions of the iteration blocks from Appendix C.3, and the read-out blocks from Appendix C.4. Then*

$$\left|\mathrm{readout}(\mathrm{TF}(\boldsymbol{Z})) - \sum_{i=1}^{N} w_i^* \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_{N+1})\right|$$
$$\leq (1 - \eta\lambda_0 (1-c))^{\ell} \frac{\left(\frac{1}{\sqrt{\lambda_0}} + 1\right) B_y}{\lambda_0 \sqrt{\kappa_{\min}}} + \frac{\frac{B_y}{\sqrt{\lambda_0}} \epsilon_{\mathrm{flip}} + (1+\lambda_0)(\epsilon_{\mathrm{sq}} + \widetilde{\epsilon}_{\mathrm{sq}})}{\lambda_0 (1-c) \sqrt{\kappa_{\min}}} + \frac{\widehat{\epsilon}_{\mathrm{sq}}}{2} + \epsilon_{\mathrm{inv}} \widetilde{B}_w,$$

where

$$\widetilde{B}_w := \frac{\left(\frac{1}{\sqrt{\lambda_0}} + 1\right) B_y}{\lambda_0 \sqrt{\kappa_{\min}}} + \frac{\frac{B_y}{\sqrt{\lambda_0}} + 2(1+\lambda_0)}{\lambda_0 (1-c) \sqrt{\kappa_{\min}}} + \frac{\left(\frac{1}{\sqrt{\lambda_0}} + 1\right) B_y}{\lambda_0}.$$

Proof. It follows from (33) and (34) with $i = N+1$ that

$$\left|o_{N+1} - \widehat{k}_{N+1} \widehat{p}_{N+1}\right| = \frac{N}{4} \left|\widehat{\tau}_{+,i} - \widehat{\tau}_{-,i}\right| \leq \frac{\widehat{\epsilon}_{\mathrm{sq}}}{2}. \tag{35}$$

Note that by definition of $\widehat{p}_{N+1}$ and relation (30), we have

$$\left|\widehat{k}_{N+1}\widehat{p}_{N+1} - \sum_{j=1}^{N} \mathcal{K}(\boldsymbol{x}_{N+1}, \boldsymbol{x}_j) w_j^{(\ell)}\right| = \left|\widehat{k}_{N+1} k_{N+1} - 1\right| \left|\sum_{j=1}^{N} \mathcal{K}(\boldsymbol{x}_{N+1}, \boldsymbol{x}_j) w_j^{(\ell)}\right|$$
$$< \epsilon_{\mathrm{inv}} k_{N+1} \left|\sum_{j=1}^{N} \mathcal{K}(\boldsymbol{x}_{N+1}, \boldsymbol{x}_j) w_j^{(\ell)}\right| \leq \epsilon_{\mathrm{inv}} \left|\sum_{j=1}^{N} \frac{\mathcal{K}(\boldsymbol{x}_{N+1}, \boldsymbol{x}_j)}{1 + \sum_{j=1}^{N} \mathcal{K}(\boldsymbol{x}_{N+1}, \boldsymbol{x}_j)}\right| \left\|\boldsymbol{w}^{(\ell)}\right\|_{\infty} \leq \epsilon_{\mathrm{inv}} B_w \leq \epsilon_{\mathrm{inv}} \widetilde{B}_w, \tag{36}$$

where in the last inequality, we relax the bound B_w to a strictly larger value $\widetilde{B}_w$ that is independent of N. Hence, by triangle inequality and estimates (35), (36), we obtain

$$\left|o_{N+1} - \sum_{j=1}^{N} \mathcal{K}(\boldsymbol{x}_{N+1}, \boldsymbol{x}_j) w_j^{(\ell)}\right| \leq \frac{\widehat{\epsilon}_{\mathrm{sq}}}{2} + \epsilon_{\mathrm{inv}} \widetilde{B}_w.$$

Combining this with Proposition 5 and triangle inequality gives the estimate of $\left|o_{N+1} - \sum_{i=1}^{N} w_i^* \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_{N+1})\right|$. The desired result immediately follows by noting that

$$o_{N+1} = \mathrm{readout}(\mathrm{TF}).$$

□

Theorem 11 (Formal version of Theorem 1)**.** *Fix a constant $c \in (0,1)$ and a step size η, satisfying*

$$0 < \eta < \frac{1}{c + 1 + \exp(2B_x^2/v^2)}. \tag{37}$$

Let $\lambda_0 \in (0,1]$ and $\epsilon \in (0, c)$. There exists a single-head transformer network TF *consisting of $(2L+5)$ blocks with MLP width W such that the final prediction satisfies*

$$\left|\mathrm{readout}(\mathrm{TF}(\boldsymbol{Z})) - \sum_{i=1}^{N} w_i^* \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_{N+1})\right| \leq C_{\mathrm{sys}} \epsilon,$$

where the constant C_{sys} is defined by

$$C_{\mathrm{sys}} := \frac{\left(\frac{1}{\sqrt{\lambda_0}} + 1\right) B_y}{\lambda_0} \left(\frac{2}{\sqrt{\kappa_{\min}}} + 1\right) + 2\frac{\frac{B_y}{\sqrt{\lambda_0}} + 2(1+\lambda_0)}{\lambda_0 (1-c) \sqrt{\kappa_{\min}}} + \frac{1}{2} = \mathcal{O}\left(\frac{1}{\lambda_0^{3/2} \sqrt{\kappa_{\min}}}\right), \tag{38}$$

the iteration count L is

$$L := \lceil \frac{\log(1/\epsilon)}{\log(1/(1-\eta\lambda_0(1-c)))} \rceil = \mathcal{O}\left(\frac{\log(1/\epsilon)}{\lambda_0}\right), \tag{39}$$

the maximum MLP width is

$$W := \max\{n_{\text{flip}}, 2n_{\text{sq}}, 2\widetilde{n}_{\text{sq}}+4, n_{\text{inv}}+1, 2\widehat{n}_{\text{sq}}, 2\} = \mathcal{O}\left(\frac{\sqrt{N/\epsilon}}{\lambda_0^{3/2}\kappa_{\min}}\right), \tag{40}$$

in which

$$n_{\text{flip}} := \lceil \frac{2\left((1-\frac{1}{1+N\kappa_{\min}})^{-1/2}-1\right)}{\sqrt{\epsilon/N}} \rceil,\ n_{\text{sq}} := \lceil \frac{B_y+B_\alpha}{\sqrt{\epsilon/N}} \rceil,$$
$$\widetilde{n}_{\text{sq}} := \lceil \frac{B_w+B_\alpha}{\sqrt{\epsilon/N^2}} \rceil,\ n_{\text{inv}} := \lceil 3\sqrt{(N+1)/\epsilon} \rceil,\ \widehat{n}_{\text{sq}} := \lceil \frac{B_w+3}{\sqrt{\epsilon/N}} \rceil,$$

with B_w and B_α defined in (16) and (21), respectively.

Proof. Let TF denote the complete transformer network constructed by composing the read-in blocks from Appendix C.2, ℓ repetitions of the iteration blocks from Appendix C.3, and the read-out blocks from Appendix C.4. We set ϵ_{sq}, $\widetilde{\epsilon}_{\text{sq}}$, $\widehat{\epsilon}_{\text{sq}}$, ϵ_{inv} and ϵ_{flip} in Proposition 10 to be ϵ. By definition $\kappa_{\min} = \exp(-2B_x^2/v^2)$ and $\lambda_0 \in (0,1]$, the step-size condition (37) results to

$$0 < \eta < \frac{1}{c+1+\exp(2B_x^2/v^2)} = \frac{1}{c+1+1/\kappa_{\min}} \le \frac{1}{\lambda_0 c+1+\lambda_0/\kappa_{\min}}.$$

Together with $\epsilon < c$, this implies the step-size condition (8). Moreover, the induction in Appendix C.3 verifies the assumptions of Theorem 4. Thus, Proposition 10 applies, which gives

$$\left|\text{readout}(\text{TF}(\boldsymbol{Z})) - \sum_{i=1}^{N} w_i^* \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_{N+1})\right|$$
$$\le (1-\eta\lambda_0(1-c))^{\ell} \frac{\left(\frac{1}{\sqrt{\lambda_0}}+1\right)B_y}{\lambda_0\sqrt{\kappa_{\min}}} + \left(\frac{\frac{B_y}{\sqrt{\lambda_0}}+2(1+\lambda_0)}{\lambda_0(1-c)\sqrt{\kappa_{\min}}} + \frac{1}{2} + \widetilde{B}_w\right)\epsilon.$$

Moreover, we set $\ell = L$ for L defined in (39), and it follows that $(1-\eta\lambda_0(1-c))^{\ell} \le \epsilon$. Therefore,

$$\left|\text{readout}(\text{TF}(\boldsymbol{Z})) - \sum_{i=1}^{N} w_i^* \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_{N+1})\right| \le \left(\frac{\left(\frac{1}{\sqrt{\lambda_0}}+1\right)B_y}{\lambda_0\sqrt{\kappa_{\min}}} + \frac{\frac{B_y}{\sqrt{\lambda_0}}+2(1+\lambda_0)}{\lambda_0(1-c)\sqrt{\kappa_{\min}}} + \frac{1}{2} + \widetilde{B}_w\right)\epsilon.$$

The overall MLP width W is determined by taking the maximum of the widths required for the zero-out layers and each of the individual approximations (18), (22), (26), (29), and (31).

□

D CLASSICAL SOLVERS

In this section, we review the classical KRR solvers with their standard convergence rates. For each method, we use its standard theoretically tuned parameters and initialize $\boldsymbol{w}^{(0)} = \boldsymbol{0}$. By $\kappa(\boldsymbol{A})$ we denote the condition number of a matrix $\boldsymbol{A}$.

Plain Richardson. Plain Richardson numerically solves the kernel system (2) with the update rule $\boldsymbol{w}^{(\ell+1)} = \boldsymbol{w}^{(\ell)} + \eta\left(\boldsymbol{y} - (\boldsymbol{K}+\lambda\boldsymbol{I})\boldsymbol{w}^{(\ell)}\right)$. Set $\eta = 2/(\text{eig}_{\max}(\boldsymbol{S}) + \text{eig}_{\min}(\boldsymbol{S}))$, where $\boldsymbol{S} := \boldsymbol{K}+\lambda\boldsymbol{I}$, and it requires $\mathcal{O}(\kappa(\boldsymbol{S})\log(1/\varepsilon))$ steps to achieve ε error (Ryaben'kii & Tsynkov, 2006).

Preconditioned Richardson. Preconditioned Richardson also solves the kernel system (2) with the update rule given in (3). Set $\eta = 2/(\text{eig}_{\max}(\boldsymbol{S}) + \text{eig}_{\min}(\boldsymbol{S}))$, where $\boldsymbol{S} := \boldsymbol{D}^{-1/2}(\boldsymbol{K}+\lambda\boldsymbol{I})\boldsymbol{D}^{-1/2}$, and it requires $\mathcal{O}(\kappa(\boldsymbol{S})\log(1/\varepsilon))$ steps to achieve ε error (Ryaben'kii & Tsynkov, 2006).

Conjugate Gradient. Conjugate Gradient is applied to the kernel system (2). Let $\boldsymbol{S} := \boldsymbol{K} + \lambda \boldsymbol{I}$. With $\boldsymbol{r}^{(0)} = \boldsymbol{y}$ and $\boldsymbol{p}^{(0)} = \boldsymbol{r}^{(0)}$, the update rule is $\boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} + \eta_k \boldsymbol{p}^{(k)}$, where $\eta_k = \frac{\|\boldsymbol{r}^{(k)}\|^2}{(\boldsymbol{p}^{(k)})^\top \boldsymbol{S}\boldsymbol{p}^{(k)}}$, $\boldsymbol{p}^{(k+1)} = \boldsymbol{r}^{(k+1)} + \frac{\|\boldsymbol{r}^{(k+1)}\|^2}{\|\boldsymbol{r}^{(k)}\|^2}\boldsymbol{p}^{(k)}$, and $\boldsymbol{r}^{(k+1)} = \boldsymbol{r}^{(k)} - \eta_k \boldsymbol{S}\boldsymbol{p}^{(k)}$. It requires $\mathcal{O}(\sqrt{\kappa(\boldsymbol{S})}\,\log(1/\varepsilon))$ steps to achieve ε error (Shewchuk, 1994).

Gradient Descent. Standard Gradient Descent is applied to the RKHS loss $L(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{K}\boldsymbol{w} - \boldsymbol{y}\|^2 + \frac{1}{2}\lambda\, \boldsymbol{w}^\top \boldsymbol{K}\boldsymbol{w}$, with update rule $\boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} - \eta\, \boldsymbol{K}((\boldsymbol{K} + \lambda \boldsymbol{I})\boldsymbol{w}^{(k)} - \boldsymbol{y})$. Set $\eta = 2/(\mathrm{eig}_{\max}(\boldsymbol{S}) + \mathrm{eig}_{\min}(\boldsymbol{S}))$, where $\boldsymbol{S} := \boldsymbol{K}(\boldsymbol{K} + \lambda \boldsymbol{I})$. It requires $\mathcal{O}(\kappa(\boldsymbol{S})\log(1/\varepsilon))$ steps to achieve ε error (Bach, 2024).

Nesterov Gradient Descent. Nesterov Gradient Descent is applied to the same RKHS loss $L(\boldsymbol{w})$. Let $\boldsymbol{S} := \boldsymbol{K}(\boldsymbol{K} + \lambda \boldsymbol{I})$. It has update rule $\boldsymbol{w}^{(k+1)} = \boldsymbol{z}^{(k+1)} + \beta\,(\boldsymbol{z}^{(k+1)} - \boldsymbol{z}^{(k)})$, where $\boldsymbol{z}^{(k+1)} = \boldsymbol{w}^{(k)} - \eta\, \boldsymbol{K}((\boldsymbol{K} + \lambda \boldsymbol{I})\boldsymbol{w}^{(k)} - \boldsymbol{y})$. Initializing $\boldsymbol{z}^{(0)} = \boldsymbol{0}$, we set $\eta = 1/\mathrm{eig}_{\max}(\boldsymbol{S})$ and $\beta = (\sqrt{\kappa(\boldsymbol{S})} - 1)/(\sqrt{\kappa(\boldsymbol{S})} + 1)$. It requires approximately $\mathcal{O}(\sqrt{\kappa(\boldsymbol{S})}\log(1/\varepsilon))$ steps to achieve ε error (Bach, 2024).

E Training and implementation details

This appendix gives the architectural, optimization, curriculum, probing, and evaluation details used for all experiments in Section 5.

Transformer architecture. The detailed structure of the GPT-2-style transformer is listed in Table 1.

Optimization. All models are trained using hyperparameters listed in Table 2.

Context-length curriculum. Following the curriculum strategy used in (Bai et al., 2023; Fu et al., 2024), we train with a gradually increasing context length. This stabilizes optimization at the maximum context length $N = 40$. At training step s, the active context length is

$$n(s) = \min\{n_{\text{start}} + \lfloor s/\Delta s \rfloor \Delta n,\ N\},$$

with $n_{\text{start}} = 11$, $\Delta n = 2$, and $\Delta s = 2000$. The curriculum reaches the maximum context length $N = 40$ after approximately 3×10^4 steps, after which all remaining training is performed at full context length.

Probe training. After training the transformer, we freeze its weights and train a separate linear probe with weight $\boldsymbol{W}^{(\ell)}_{\text{probe}} \in \mathbb{R}^{d_{\text{model}} \times 1}$ and bias $b^{(\ell)}_{\text{probe}} \in \mathbb{R}$ for each layer $\ell \in [L]$. Each probe maps the hidden state at the final query x-token to a scalar prediction. Probes are trained for 500,000 steps on the same data distribution and context-length curriculum as the base model, using learning rate 10^{-4}.

Computational resources. For all experiments, the transformer is trained on a single NVIDIA V100 or RTX A4000 GPU. Layer-wise probes are trained in parallel across 4 RTX A4000 GPUs.

Table 1: Default transformer architecture used in the main experiments.

Hyperparameter	Value
Layers L	12
Heads H	8
Hidden width d_{model}	256
Feedforward width d_{ff}	1024 ($= 4d_{\text{model}}$)
Dropout	0.0
Layer normalization	Pre-LN
Activation	GELU
Positional encoding	Learned absolute embeddings
Attention mask	Causal

Table 2: Optimization hyperparameters for transformer training runs.

Hyperparameter	Value
Optimizer	AdamW
β_1, β_2	$0.9, 0.999$
Weight decay	10^{-4}
Initial learning rate	10^{-4}
Final learning rate	10^{-5}
Schedule	Cosine annealing
Training steps	500,000
Batch size	64
Gradient clipping	1.0
Loss	MSE on causal x-token predictions

F Additional Experimental Results

Throughout this appendix we report full results for all experiments developed in the main text.

F.1 Plain Richardson versus Preconditioned Richardson

We compare plain and preconditioned Richardson iteration across Gaussian, spherical, and uniform inputs, examining step-size thresholds (Figure 3), spectral condition numbers (Figure 4), and relative coefficient error over iterations under each method's optimal step size (Figure 5). We use 5 seeds with 256 sequences per seed. Curves show the mean of the within-seed medians, with shaded bands indicating standard deviation across seeds.

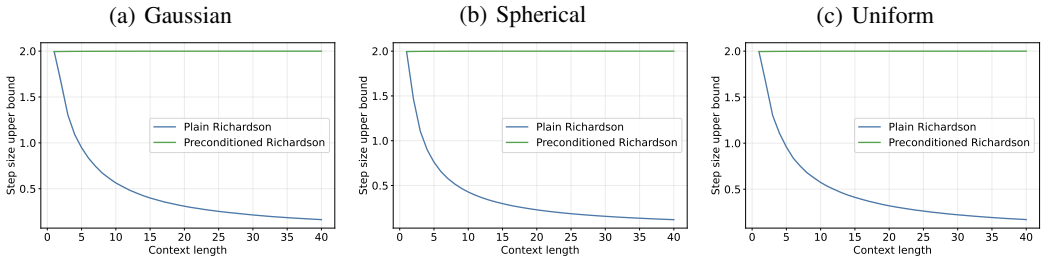


Figure 3: The step-size threshold $2/\mathrm{eig}_{\max}(\boldsymbol{S})$ versus context length. The threshold remains approximately constant for preconditioned Richardson but decreases with context length for plain Richardson.

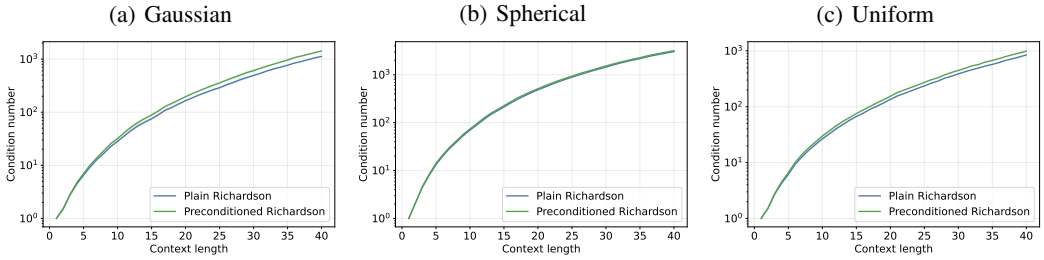


Figure 4: Spectral condition number $\kappa(\boldsymbol{S})$ versus context length. Plain and preconditioned Richardson have comparable condition numbers.

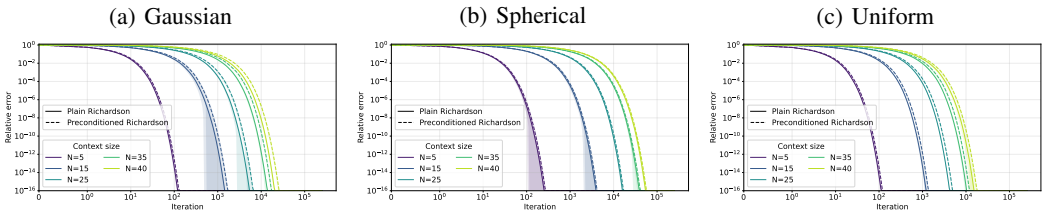


Figure 5: Convergence under each method's optimal step size $2/(\mathrm{eig}_{\max}(\boldsymbol{S}) + \mathrm{eig}_{\min}(\boldsymbol{S}))$. Plain and preconditioned Richardson require comparable numbers of iterations to reach the same relative coefficient error.

F.2 Additional results on progressive alignment

We supplement Section 5.1 with additional results on prediction-error and induced-weight alignment. Figure 6 extends the prediction-error comparison in Figure 1(a) to three input distributions, and Figure 7 extends the induced-weight alignment analysis in Figure 1(b) across context lengths and input distributions.

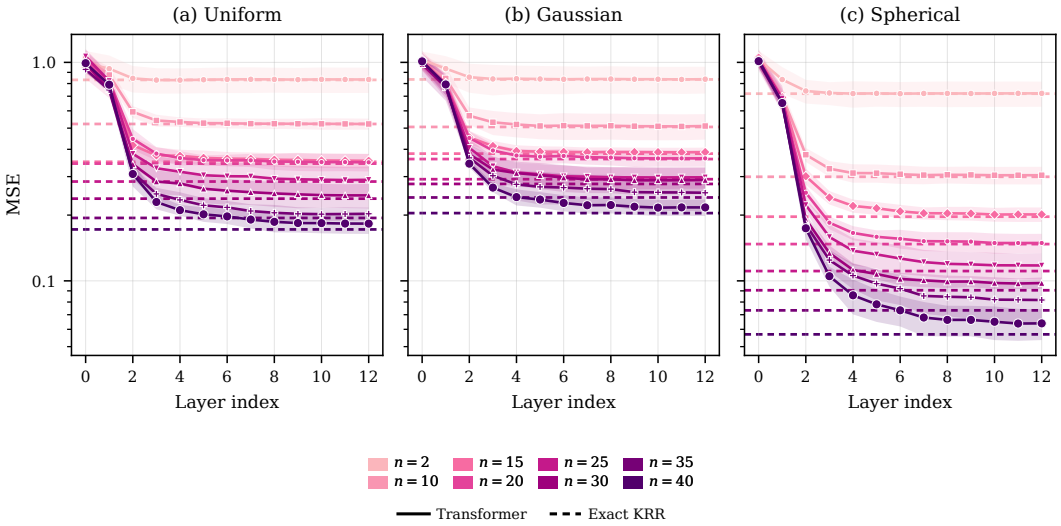
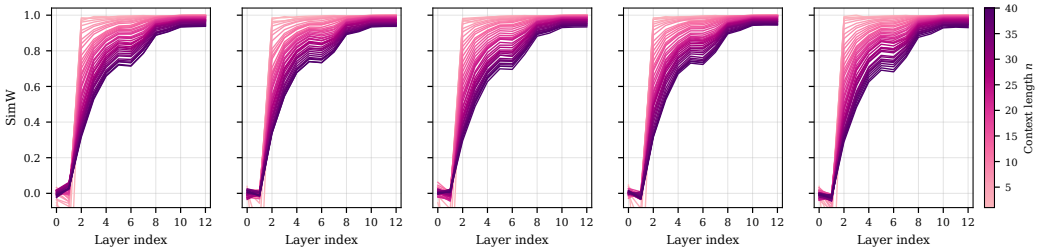
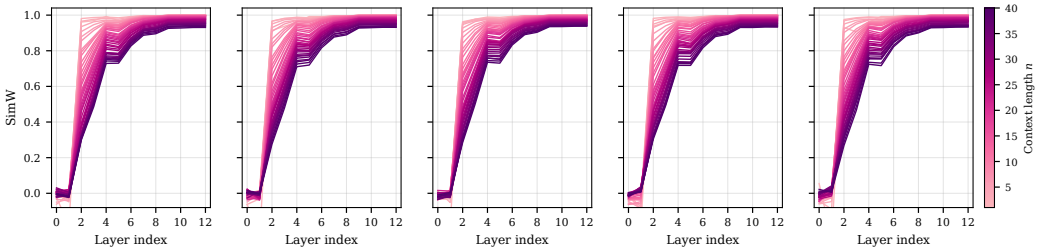


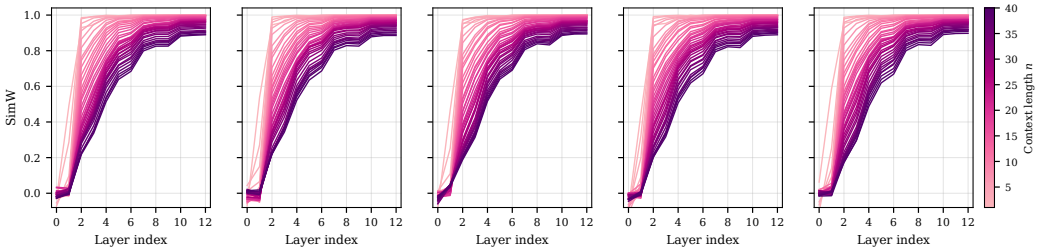
Figure 6: **Prediction-error alignment across input distributions.** Layer-wise prediction MSE across context lengths for Gaussian, spherical, and Uniform inputs. Solid curves show the transformer MSE, and dashed horizontal lines show the corresponding exact-KRR MSEs. Curves and shaded regions indicate the mean and one standard deviation over five evaluation seeds, respectively.



(a) Uniform



(b) Gaussian



(c) Spherical

Figure 7: **SimW across depth and context length.** Layer-wise SimW between the transformer's induced weights and the exact KRR weights under Uniform, Gaussian, and Spherical inputs. Within each subfigure, the five panels correspond to evaluation seeds 1–5 from left to right, and each curve represents a different context length n. Across all three input distributions and context lengths, alignment with the exact KRR weights progressively increases with transformer depth.

F.3 Ablation results

Robustness to linear probing. Figures 8 and 9 explore the role of linear probes, discussed in Section 5.2, through random-backbone controls and comparison with the transformer's native readout, respectively.

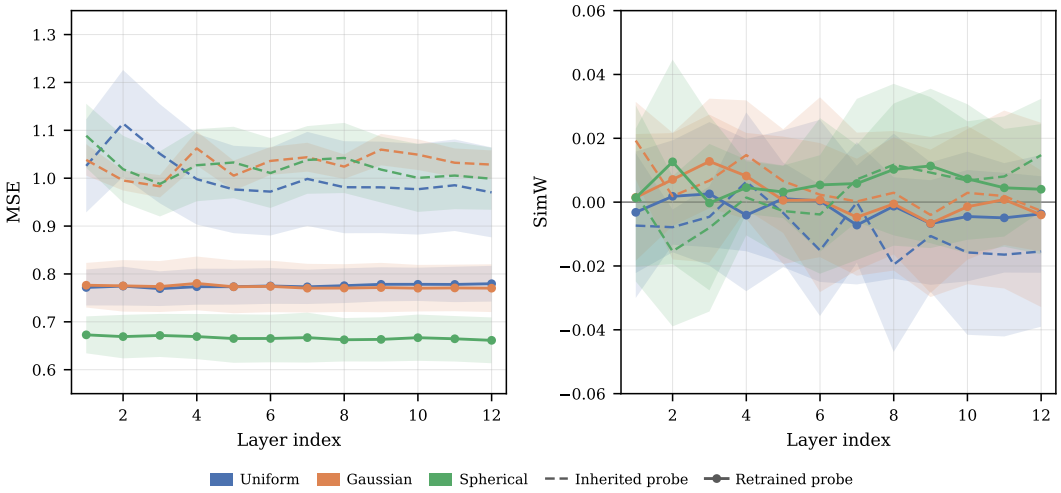


Figure 8: **Random-backbone ablations.** Layer-wise prediction MSE and SimW at $n = 40$ for Gaussian, spherical, and Uniform inputs. The random backbone is evaluated using either probes transferred from the trained backbone or probes retrained on the random backbone.

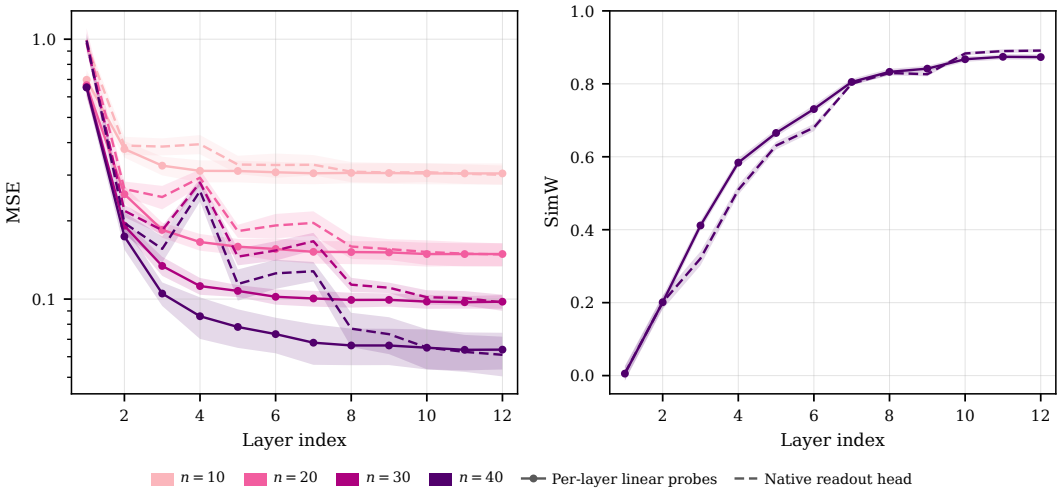


Figure 9: **Linear probing versus the native transformer readout on spherical inputs.** Layer-wise prediction MSE and SimW at $n = 40$ obtained using separately trained linear probes or the transformer's trained final LayerNorm and prediction head applied directly to each intermediate hidden state.

Mismatched estimator kernel. Figure 10 presents the spherical-input evaluation for the kernel-mismatch ablation discussed in Section 5.2.

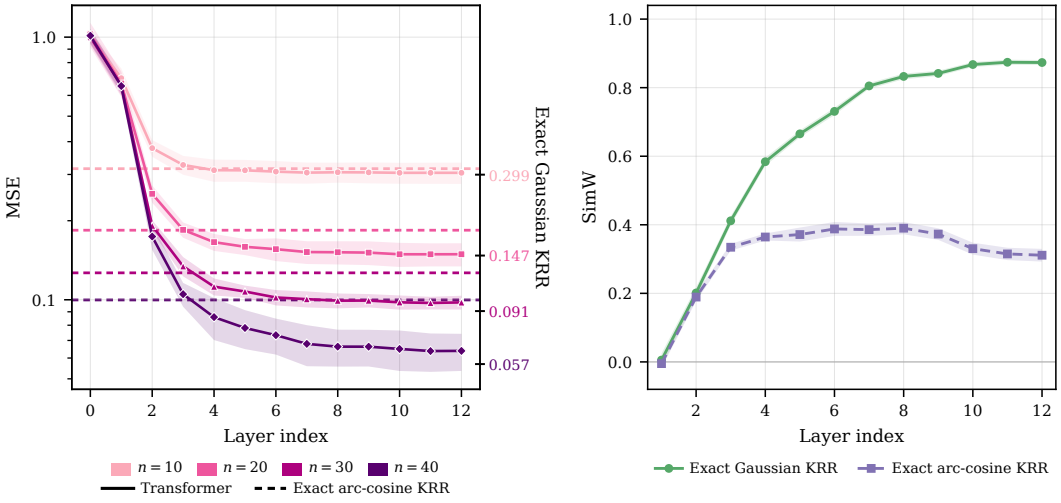


Figure 10: **Kernel-mismatch ablation on spherical inputs.** *Left:* Layer-wise transformer prediction MSE; dashed horizontal lines show the exact arc-cosine-KRR MSEs, while the exact Gaussian-KRR MSEs are marked on the right axis. *Right:* Layer-wise SimW between the transformer-induced weights and the exact KRR coefficient vectors obtained using the Gaussian and arc-cosine estimator kernels.

Noise-level ablation. Figure 11 reports the noise-level sweep across the three input distributions for the ablation discussed in Section 5.2.

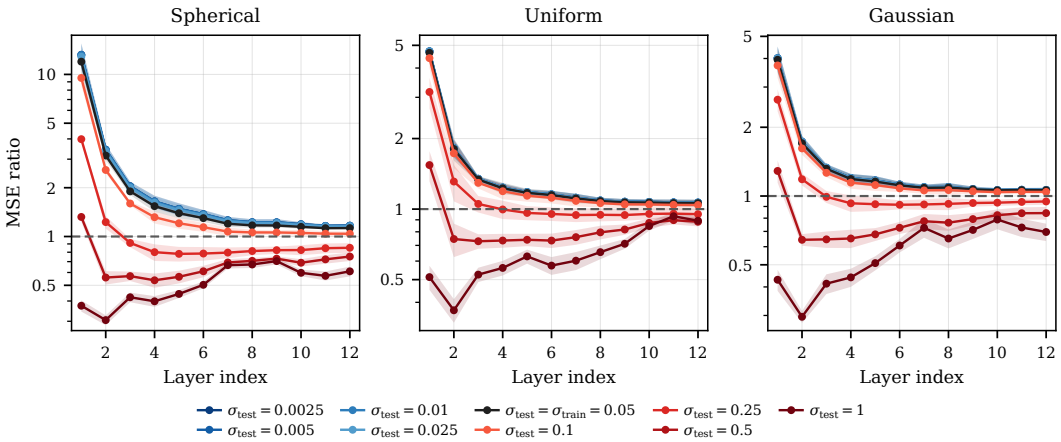


Figure 11: **Layer-wise $\mathrm{MSE}_{\mathrm{TF}}/\mathrm{MSE}_{\mathrm{KRR}}$ for different test noise levels and input distributions.** KRR uses $\lambda = \sigma_{\mathrm{train}}^2$. Curves and shaded regions show the mean and one standard deviation over five evaluation seeds, respectively.

F.4 Solver comparisons for Gaussian and Uniform inputs

Figures 12 and 13 extend the solver comparison in Figure 2 to Gaussian and Uniform inputs using the same evaluation protocol.

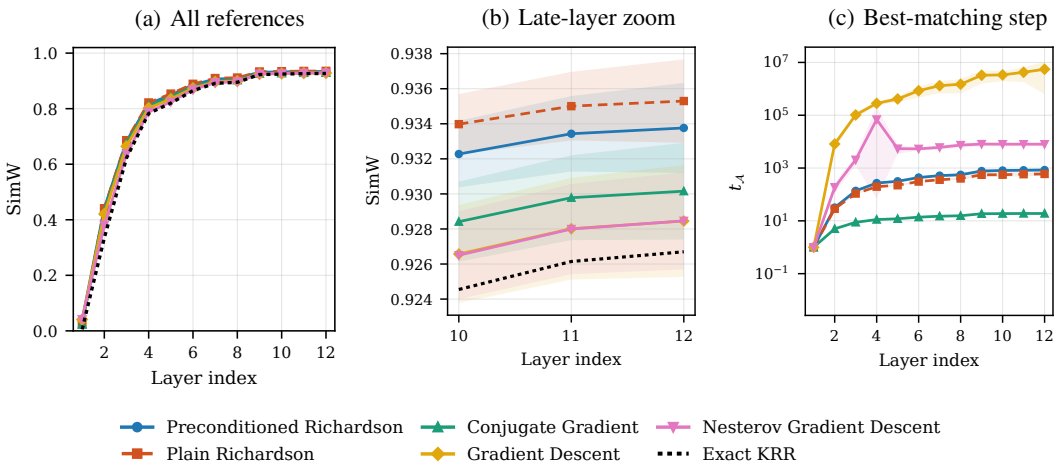


Figure 12: **Best-match SimW and corresponding solver iterations for Gaussian inputs** at $n = 40$. Panels follow the organization of Figure 2.

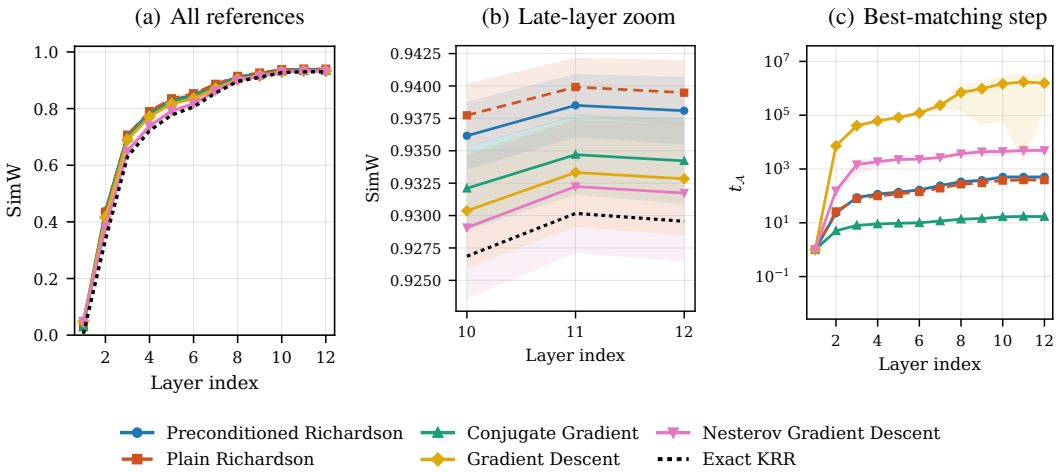


Figure 13: **Best-match SimW and corresponding solver iterations for Uniform inputs** at $n = 40$. Panels follow the organization of Figure 2.