

---

# Linear Transformers Implicitly Discover Unified Numerical Algorithms

---

Anonymous Author(s)

Affiliation

Address

email

## Abstract

1 A transformer is merely a stack of learned data-to-data maps—yet those maps  
2 can hide rich algorithms. We train a *linear, attention-only* transformer on millions  
3 of *masked-block* completion tasks: each prompt is a masked low-rank matrix  
4 whose missing block may be (i) a scalar prediction target or (ii) an unseen kernel  
5 slice for Nyström extrapolation. The model sees only input-output pairs and a  
6 mean-squared loss; it is given no normal equations, no handcrafted iterations, and  
7 no hint that the tasks are related. Surprisingly, after training, algebraic unrolling  
8 reveals the *same* parameter-free update rule across *all* three resource regimes (full  
9 visibility, bandwidth-limited heads, rank-limited attention). We prove that this  
10 rule achieves second-order convergence on full-batch problems, cuts distributed  
11 iteration complexity, and remains accurate with compute-limited attention. Thus, a  
12 transformer trained solely to patch missing blocks *implicitly discovers* a unified,  
13 resource-adaptive iterative solver spanning prediction, estimation, and Nyström  
14 extrapolation—highlighting a powerful capability of in-context learning.

## 15 1 Introduction

16 Models trained on next-token prediction achieve strong performance across various NLP tasks,  
17 including question answering, summarization, and translation (Radford et al., 2019). This multitask  
18 capability suggests an intriguing possibility: within structured mathematical contexts transformers  
19 might implicitly learn generalizable numerical algorithms solely through next-token prediction.

20 Next-token prediction can naturally be viewed as a form of matrix completion—inferring missing en-  
21 tries by exploiting dependencies in observed data. Prior work extensively explores matrix completion  
22 under computational constraints, such as distributed data access Chen et al. (2020), limited communi-  
23 cation bandwidth Ma and Chen (2020), and low-rank recovery from partial observations Candès and  
24 Recht (2009); Candès and Tao (2010); Davenport and Romberg (2016). Yet a key question remains:  
25 how do transformers implicitly handle these constraints to perform multitask learning?

26 *Can a neural network invent a numerical algorithm simply by learning to fill in missing data?*

27 Transformers excel at *in-context learning* (ICL), adapting to new tasks from a short prompt of exam-  
28 ples Brown et al. (2020); Xie et al. (2021). Recent studies suggest that transformer computations  
29 resemble gradient-based methods Akyürek et al. (2022); Von Oswald et al. (2023); Ahn et al. (2023),  
30 typically in single-task scenarios. However, this analogy is limited: *gradient descent operates explic-  
31 itly in parameter space, whereas transformers perform data-to-data transformations without direct  
32 access or updates to parameters*. Thus, it remains unclear whether transformers implicitly develop  
33 distinct, unified numerical algorithms suited to diverse tasks and resource constraints.

34 **Masked-block Completion and Architectural Masks:** To probe this question, we train a linear  
35 transformer on masked-block completion tasks that hide one block of a low-rank matrix mirroring the  
36 classical Nyström completion problem. The transformer’s task is to infer these missing blocks based

solely on observable data and a mean-squared error objective—without explicit guidance about the underlying numerical method or relationships between tasks. We impose three distinct architectural visibility constraints on transformers, each reflecting practical computational limitations common in large-scale optimization: *centralized* (full visibility), *distributed* (restricted communication), and *computation-limited* (restricted complexity via low-dimensional attention). Each regime employs the same underlying transformer architecture, differing only minimally in their attention patterns.

**Emergence of a Unified Algorithm:** Remarkably, despite training independently under these distinct computational constraints, we find that transformers implicitly uncover the same concise, two-line iterative update rule. This unified algorithm emerges consistently across tasks and constraints, exhibiting strong theoretical and empirical properties: it achieves second-order convergence on full-batch problems, matches classical methods in centralized settings, significantly reduces communication complexity in distributed settings and remains accurate under computation-limited conditions.

**Contributions:** • *Unified masked-block completion benchmark:* A single training framework integrating multiple prediction or inference tasks into a unified interface.

• *Resource-aware transformer architectures:* Transformer-based models naturally adapt to centralized, distributed, and computation-limited environments through simple architectural constraints.

• *Implicit numerical algorithm discovery:* Transformers implicitly discover a unified, efficient numerical solver that achieves second-order convergence on full-batch problems and consistently matches or outperforms classical methods across completion, extrapolation, and estimation tasks under varying resource constraints. Our theoretical results uniformly apply across all these tasks, highlighting the broad applicability of the discovered algorithm.

These findings position transformers trained on block completion as powerful tools for uncovering numerical algorithms, offering a promising route toward developing adaptive, data-driven solvers.

## 2 Related Work

We focus on literature most pertinent to viewing transformers as fixed data-to-data transforms whose forward pass exposes emergent algorithms.

*Implicit algorithm learning.* Transformers have been shown to recover 1<sup>st</sup>- and 2<sup>nd</sup>-order methods for least squares, dual GD for optimal transport, and TD updates for RL (Von Oswald et al., 2023; Ahn et al., 2023; Daneshmand, 2024; Wang et al., 2025; Fu et al., 2024; Giannou et al., 2023). Weight-level circuit studies reverse-engineer copy-and-addition “induction heads” (Olah et al., 2022; Nanda, 2023), but are still confined to token-manipulation algorithms. RNNs and MLPs display related behavior (Siegelmann and Sontag, 1992; Tong and Pehlevan, 2024), yet attention yields depth-aligned, easily inspected updates (Garg et al., 2023). Most prior work is therefore limited to first-order rules and a single, centralized computational regime.

*Task vs. regime.* Earlier in-context studies (Akyürek et al., 2022; Bai et al., 2023; Min et al., 2021) vary the regression task—sampling while the hardware regime stays fixed. We take the opposite view: we fix one low-rank matrix-completion task and show that the same transformer discovers a rule that adapts automatically as compute, memory, or communication budgets are tightened.

*Data-space vs. parameter-space.* These analyses interpret the forward pass as hidden *parameter* updates. In reality, a transformer is a *data-to-data map*: weights are frozen at test time. Weight-level ‘circuit’ studies (Olah et al., 2022; Nanda, 2023) view trained transformers as data-space algorithms, but focus on copying or addition; we uncover a second-order projector for numerical linear algebra. We show that pre-training can imprint a *second-order, data-space* Newton–Schulz method that converges quadratically and runs unchanged across centralized, distributed, and sketched settings.

*In-context regression limits.* Theoretical lenses that view in-context learning as linear regression (Akyürek et al., 2022; Bai et al., 2023) recover gradient-descent dynamics with  $\sqrt{\kappa}$  dependence on the condition number. Vladymyrov et al. (2024) show that, with suitable parametrization, preconditioning yields second-order dynamics and  $\log \kappa$  convergence. We show that transformer training naturally converges to such parametrizations in practice, with this behavior robust to bandwidth and memory constraints. Theoretically, we prove that favorable guarantees hold across all studied settings.

*Links to classical numerical algorithms.* Newton–Schulz iterations achieve quadratic rates (Higham, 1997) but require explicit matrix products and break down under noise. Nyström and randomized low-rank approximations accelerate kernel methods (Williams and Seeger, 2001; Halko et al., 2011; Gittens and Mahoney, 2016) yet revert to Krylov solvers for accuracy. We show that a single transformer automatically instantiates the same rule and adapts across resource regimes—without hand-crafted communication or sketch design.

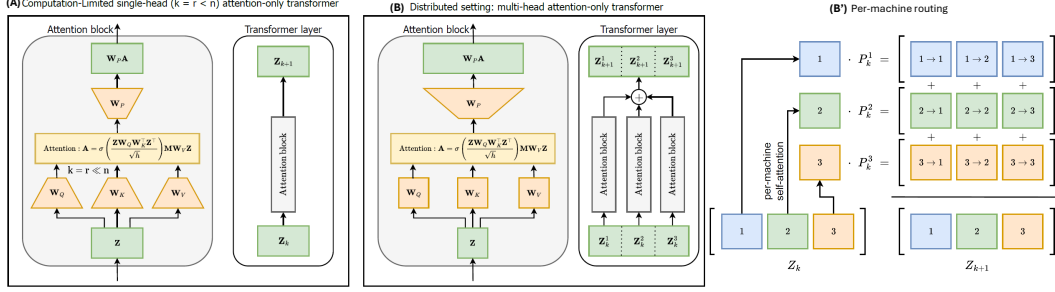


Figure 1: **Architectural regimes studied in this work.** (A) *Computation-limited*: a single-head attention-only transformer whose query, key and value projections are restricted to a low-rank embedding of dimension  $k = r \ll n$ , creating an explicit bottleneck that reduces the per-layer cost from  $\Theta(n^2 d)$  to  $\Theta(n r d)$ ; *Unconstrained*, this is the case where the embedding dimension is set to  $k = n + n'$ . (B) *Distributed*: a multi-head transformer in which each head operates on data stored on a separate machine; the heads run local attention and their outputs are aggregated. (B') *Per-machine routing*: detailed view of the distributed setting showing how each machine  $\mu$  forms its local projection  $P_k^\mu$  and contributes to the next-layer representation  $Z_{k+1}$ .

93 *Distributed and sketched solvers.* Block-CG, communication-avoiding Krylov (Demmel et al., 2013;  
 94 Hoeffler et al., 2019) and sketch-and-solve techniques (Clarkson and Woodruff, 2017; Woodruff,  
 95 2014) dominate practice but remain first-order. Our diversity index  $\alpha$  predicts when the learned rule  
 96 out-iterates these baselines ( $\alpha^{-1} \ll \sqrt{\kappa}$ ).

### 97 3 Method

98 We explain how data are generated and encoded, how architectural constraints enforce three resource  
 99 regimes, and how we extract an explicit numerical solver from trained weights.

100 **Block Completion Setup.** We generate a low-rank matrix task by first drawing a matrix of the form

$$X = \begin{bmatrix} A & C \\ B & D \end{bmatrix} \in \mathbb{R}^{(d+d') \times (n+n')}, \quad \text{rank}(X) = \text{rank}(A), \quad (1)$$

101 and then feeding the transformer a prompt matrix  $Z_0$  in which the lower-right grey block  $D$  is masked,  
 102 i.e., set to zero. Given the visible blocks  $A, B, C$ , the model must reconstruct  $D$  with low  $\ell_2$  error.  
 103 This single prompt template generalizes Nyström extrapolation and scalar regression (Appx. B). We  
 104 train on both exact samples and noisy variants obtained by adding Gaussian noise with variance  
 105  $\sigma^2 \in \{0, 10^{-2}\}$ , so the low-rank assumption is approximate but realistic. Note that the Nyström  
 106 approximation is the minimum-rank completion that matches observed entries, even if  $X$  is full rank.

107 **Data generation.** We construct rank- $s$  data matrices  $X \in \mathbb{R}^{(d+d') \times (n+n')}$  by sampling  $R_1 \in$   
 108  $\mathbb{R}^{(d+d') \times s}$  and  $R_2 \in \mathbb{R}^{(n+n') \times s}$ , each with rows drawn independently from  $\mathcal{N}(0, \Sigma)$ , and setting  
 109  $X = R_1 R_2^T / \sqrt{s}$ . To control the difficulty of the problem, we choose  $\Sigma$  to be diagonal with entries  
 110  $\Sigma_{ii} = \alpha^i$ , where  $\alpha < 1$ , inducing strong anisotropy. Unless stated otherwise, we use the following  
 111 parameters throughout:  $n = d = 18$ ,  $n' = d' = 2$ ,  $s = 10$  and  $\alpha = 0.7$ . This yields matrices  $X$  with  
 112 condition number on the order of  $10^3$ . Half the time, we add Gaussian noise of variance 0.01 to  $X$ .

113 **Linear-attention transformer.** We employ the linear-attention variant of transformers (Ahn et al.,  
 114 2023; Von Oswald et al., 2023; Wang et al., 2025). Each layer  $\ell$  applies multi-head attention with a  
 115 residual skip connection, to transform  $Z_0 = \begin{bmatrix} A & C \\ B & 0 \end{bmatrix}$  to  $Z_\ell := \begin{bmatrix} A_\ell & C_\ell \\ B_\ell & D_\ell \end{bmatrix}$  via the iterative structure

$$Z_{\ell+1} = Z_\ell + \sum_{h \in [1:H]} \text{Attn}_\ell^h(Z_\ell), \quad \text{Attn}_\ell^h(Z) = (Z W_{Q,\ell}^h (Z W_{K,\ell}^h)^T \odot M_\ell^h) Z W_{V,\ell}^h W_{P,\ell}^{h\top}, \quad (1)$$

116 where  $W_Q, W_K, W_V, W_P \in \mathbb{R}^{n \times k}$  are query, key, value, and projection matrices, and the fixed  
 117 mask  $M_\ell^h = \begin{bmatrix} \mathbf{1}_{(d+d')} \mathbf{1}_d^\top & 0_{(d+d') \times d'} \end{bmatrix}$  blocks the flow of information from the incomplete  $(C_\ell, D_\ell)$   
 118 column in  $Z_\ell$  towards the visible column. Models are trained to minimize mean-squared error on  $D$ ;  
 119 training details are in §B. A schematic of the architecture appears in Fig. 1(A).

120 **Computational Regimes via Architectural Constraints.** We study three distinct computational  
 121 regimes—unconstrained (or centralized), computation-limited (low-dimensional attention), and  
 122 distributed (restricted inter-machine communication)—each encoded into the architecture through  
 123 explicit attention and dimensionality constraints. Figure 1 summarizes these architectural regimes.

124 *Unconstrained.* No visibility or dimension constraints are imposed; each token attends to all others.  
 125 We set the embedding dimension  $k = n + n'$ .

126 *Computation-Limited.* To emulate memory- or latency-bounded hardware we enforce a *low-rank*  
 127 *attention* constraint: the query and key matrices have embedding size  $k = r \ll n$ , and the value and  
 128 projection to  $r + n'$  (we use  $r = 5 \approx n/4$ ; see Fig. 1A). This bottleneck compresses the input and  
 129 cuts the per-layer cost of recovered algorithms from  $\Theta(nd^2)$  to  $\Theta(ndr)$ . (See §B for details).

130 *Distributed Computation.* The prompt  $Z_0$  distributes  $X$  across  $M$  machines through a block structure

$$Z_0 = [\dots \ X^\mu \ X^{\mu+1} \ \dots] \in \mathbb{R}^{(d+d') \times M(n+n')}, \text{ where } X^\mu = \begin{bmatrix} A^\mu & C \\ B^\mu & D \end{bmatrix}, \mu \in [1 : M]$$

131 Each head is assigned to a machine  $\mu$ , and all but the  $\mu$ th block in its query, key matrices are zeroed,  
 132 enforcing local self-attention. Each  $\mu$  then ‘transmits’ information to others through the value,  
 133 projection matrices, and incoming transmissions are summed to generate  $Z_{\ell+1}$ . The full algebraic  
 134 form, matrix partitioning and communication setups are provided in §B.

135 **Algorithm extraction.** We generate explicit algorithms from a trained transformer by progressively  
 136 simplifying its weights until a concrete update rule emerges. The steps taken are (also see §B):

- 137 • *Weight quantization.* Cluster component values and sparsify matrices by dropping the values below  
 138  $\leq \tau$ . We then evaluate the performance with clustered weights to confirm no loss in performance.
- 139 • *Matrix Property Tests.* Check whether resulting matrices are random, sparse, or low-rank.
- 140 • *Scaling Laws.* Identify how transformer scales attention and value blocks layer-by-layer.

141 For the sake of the same simplicity, in the unconstrained and compute-limited settings, we restrict the  
 142 architecture to one head per layer, while in the distributed setting we use one head per machine. After  
 143 these simplifications every regime collapses to the *same* two-line, parameter-free update rule (Eq. 2  
 144 in § 4), providing a direct algorithmic interpretation of the transformer’s in-context computation.

## 145 4 Emergent Algorithm

146 **One method, three resource regimes.** Across all three archi-  
 147 tectural constraints, we find that the trained transformer  
 148 solves the matrix completion task (Fig. 2) while implic-  
 149 itly implementing the *same* two-line scaling transformation  
 150 when its weights are algebraically interpreted. Algorithm 1  
 151 visualises the result, which we call EAGLE (Emergent  
 152 Algorithm for Global Low-rank Estimation). The blue  
 153 UPDATE box is identical in all scenarios and across all  
 154 machines, while the outer grey look captures the informa-  
 155 tion fusion in the distributed setting (i.e., if  $M > 1$ ). The  
 156 matrix  $S$  in UPDATE is an orthogonal *sketching matrix*  
 157 (see below). Table 1 shows that this extracted algorithm  
 158 reproduces the exact layer-wise activations of the trans-  
 159 former to within  $6 \cdot 10^{-4}$  error, demonstrating its fidelity.

160 We now discuss how this algorithm is derived in the three settings, focusing on the noiseless case.  
 161 However, our findings remain valid under modest data noise ( $\sigma^2 = 0.01$ , see §C).

162 **Unconstrained setting.** Because token updates are not restricted, the weight patterns are the cleanest  
 163 to interpret here, and will re-appear in the distributed and compute-limited regimes.

layer	U $\times 10^4$	D $\times 10^4$	C-L $\times 10^4$
0	0.00	0.00	0.00
1	2.36	0.62	0.01
2	5.27	1.40	0.02
3	3.70	1.28	0.13
4	2.16	1.32	0.14

Table 1: Differences across iterations between transformer and extracted algorithm (squared Frobenius norm divided by size of  $Z$ ) in the unconstrained (U), distributed (D), and compute-limited (C-L) settings. Mean times  $10^4$  across 10 seeds is reported.

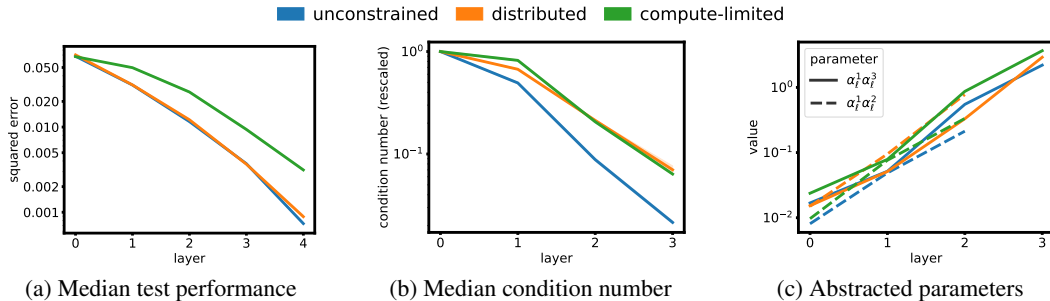


Figure 2: The trained transformer solves matrix completion with a unified algorithm over all three computational settings. The evolution of key quantities throughout the transformer layers illustrate the remarkable similarity between the latent algorithms. Mean across 10 training seeds is reported. See §C for plots with noisy data.

<pre> 1: <b>function</b> UPDATE(<math>A, B, C, D</math>) 2:   <math>\tilde{A} \leftarrow AS, \tilde{B} \leftarrow BS</math> 3:   <math>\rho \leftarrow \ \tilde{A}\ _2^{-2}</math> 4:   <math>A' \leftarrow A - \eta\rho\tilde{A}\tilde{A}^\top\tilde{A}S^\top</math> 5:   <math>B' \leftarrow B - \eta\rho\tilde{B}\tilde{A}^\top\tilde{A}S^\top</math> 6:   <math>C' \leftarrow C - \gamma\rho\tilde{A}\tilde{A}^\top C</math> 7:   <math>D' \leftarrow D - \gamma\rho\tilde{B}\tilde{A}^\top C</math> 8:   <b>return</b> (<math>A', B', C', D'</math>) 9: <b>end function</b> </pre>	<pre> 1: <b>Input:</b> <math>\{A_0^\mu, B_0^\mu\}_{\mu=1}^M</math>, query/output (<math>C_0, D_0</math>) 2: <b>for</b> <math>\ell = 0, \dots, L-1</math> <b>do</b> 3:   <math>C_{\text{sum}} \leftarrow 0, D_{\text{sum}} \leftarrow 0</math> 4:   <b>for</b> <math>\mu = 1, \dots, M</math> <b>in parallel do</b> 5:     <math>(A_{\ell+1}^\mu, B_{\ell+1}^\mu, C'^\mu, D'^\mu) \leftarrow \text{UPDATE}(A_\ell^\mu, B_\ell^\mu, C_\ell, D_\ell)</math> 6:   <b>end for</b> 7:   <math>C_{\ell+1} \leftarrow \frac{1}{M} \sum_\mu C'^\mu; \quad D_{\ell+1} \leftarrow \frac{1}{M} \sum_\mu D'^\mu</math> 8: <b>end for</b> 9: <b>Return</b> <math>D_L</math> </pre>
---	--

**(A) Update: Identical for all regimes**

**(B) Fusion: Lightweight Communications**

Algorithm 1: **Emergent Algorithm for Global Low-rank Estimation (EAGLE)**. *Left*: the numerical kernel UPDATE runs unchanged on every machine. *Right*: the outer loop calls UPDATE, and in the distributed regime ( $M > 1$ ), it averages the resulting query updates ( $C'$ ) and outputs ( $D'$ ).  $M = 1$  in the unconstrained and computation-limited settings. In the former,  $S = I_n$ , while in the latter,  $S \in \mathbb{R}^{(d+d') \times r}$  is composed of random orthogonal rows. The values  $\eta, \gamma$  are global constants (see below for their values).

164 • *Emergent weight structure*. During the extraction procedure we progressively pruned attention  
165 heads and quantised weights while tracking validation loss. Remarkably, *one head per layer* is  
166 already sufficient: pruning from eight to a single head has little effect on test MSE. After pruning, the  
167 weight products  $W_{Q,\ell}W_{K,\ell}^\top$  and  $W_{V,\ell}W_{P,\ell}^\top$  collapse to (almost) diagonal matrices.

$$W_{QK,\ell} := W_{Q,\ell}W_{K,\ell}^\top \approx \text{diag}(\alpha_\ell^1 I_n, 0_{n'}), \quad W_{VP,\ell} := W_{V,\ell}W_{P,\ell}^\top \approx \text{diag}(\alpha_\ell^2 I_n, \alpha_\ell^3 I_{n'}),$$

168 where only the three scalars  $\alpha_\ell^{1,2,3}$  vary from layer to layer. Figure 3 (left) visualises a typical pair;  
169 layer-wise statistics across 10 seeds are reported in §C. This near-block-diagonal structure lets us  
170 algebraically reduce the transformer’s forward step to the update shown in Fig. 1.

171 • *From weights to UPDATE*. The additive steps in UPDATE are due to the residual connection. Further,  
172 the diagonal forms of  $W_{QK,\ell}$  and  $W_{VP,\ell}$  imply that (see §C), the attention block acts as

$$\text{Attn}_\ell(Z_\ell) = \begin{bmatrix} \alpha_\ell^1 \alpha_\ell^2 A_\ell A_\ell^\top A_\ell & \alpha_\ell^1 \alpha_\ell^3 A_\ell A_\ell^\top C_\ell \\ \alpha_\ell^1 \alpha_\ell^2 B_\ell A_\ell^\top A_\ell & \alpha_\ell^1 \alpha_\ell^3 B_\ell A_\ell^\top C_\ell \end{bmatrix}. \quad (2)$$

173 The form of the iterative update in Alg. 1 with  $M = 1, S = I_n$  is then immediate.

174 • *Rationale for Weights*. Further, the values  $\alpha_\ell^{1,2,3}$  see two persistent structural effects: throughout,  
175  $\alpha_\ell^1 \alpha_\ell^2 \approx \eta \|A_\ell\|_2^{-2}$ , and  $\alpha_\ell^1 \alpha_\ell^3 \approx \gamma \|A_\ell\|_2^{-2}$ , where  $\|A_\ell\|_2$  is the spectral norm of  $A_\ell$  (i.e., largest  
176 singular value, tuned to the largest such value typically seen in a training batch), and  $\gamma, \eta$  are the  
177 constants  $\eta \approx 1$  and  $\gamma \approx 1.9$ . Thus, the scale of these weights is determined by  $\|A_\ell\|_2^{-2}$ , which is  
178 used in a fixed way by the method. Figure 5 shows that  $\alpha_\ell^1 \alpha_\ell^2 \|A_\ell\|_2^2$  is constant across layers.

179 • *Understanding the method*. Note that the transformation  $A \mapsto (I - \rho\eta AA^\top)A$  contracts large  
180 singular values of  $A$  more than small singular values. The net effect of this repeated action on  
181  $A_\ell$  is that for large  $\ell$ , all initially nonzero singular values of  $A_\ell$  converge to one another, i.e., the  
182 matrix  $A$  becomes well-conditioned (see Fig. 2). This aspect of the method is reminiscent of the  
183 Newton-Schulz method for matrix inversion (Schulz, 1933; Ben-Israel, 1965; Higham, 2008; Fu et al.,  
184 2024). The overall structure of the iterations can then be seen as a ‘continuous conditioning update’  
185 for  $A_\ell$ . The iterations for  $C_\ell, D_\ell$  are reminiscent of gradient descent, adapted to the varying  $A_\ell$ .

186 • *Relation to prior work*. Von Oswald et al. (2023) dubbed the same update “GD<sup>++</sup>” in the scalar  
187 case ( $d' = n' = 1$ ) and interpreted it as pre-conditioned gradient descent. Our analysis shows it is  
188 closer to a *Newton–Schulz conditioning loop* wrapped around direct prediction; we therefore refer to  
189 it as the EAGLE method in the rest of the paper. §C contrasts the two viewpoints in detail.

190 **Distributed setting**. Following the unconstrained setting, we train with a single head per machine.

191 • *Communication Structure* As detailed in §3, for each head, the block structure of the  $((n + n')) \times$   
192  $(M(n + n'))$  value-projection matrix  $W_{VP,\ell}^\mu$  governs what machine  $\mu$  sends to other machines. Fig. 3  
193 (middle) reveals two striking regularities in this matrix.

194 – *Within-machine blocks*. Every diagonal block equals  $\text{diag}(\alpha_\ell^1 I_n, \alpha_\ell^2 I_{n'})$ —exactly the same  
195 form as in the unconstrained model, with the *same* pair  $(\alpha_\ell^1, \alpha_\ell^2)$  across all machines.

196 – *Across-machine blocks*. Off-diagonal blocks are  $\text{diag}(0_n, \alpha_\ell^2 I_{n'})$ : i.e. only the incomplete  
197 columns  $(C_\ell, D_\ell)$  are transmitted, and with the very same factor  $\alpha_\ell^2$  as the within machine blocks.

198 • *Resulting algorithm*. Since  $W_{QK,\ell}^\mu$  and the local block of  $W_{VP,\ell}^\mu$  have the same structure as the un-  
199 constrained setting, each machine executes the same local iteration as in the UPDATE method. Further,

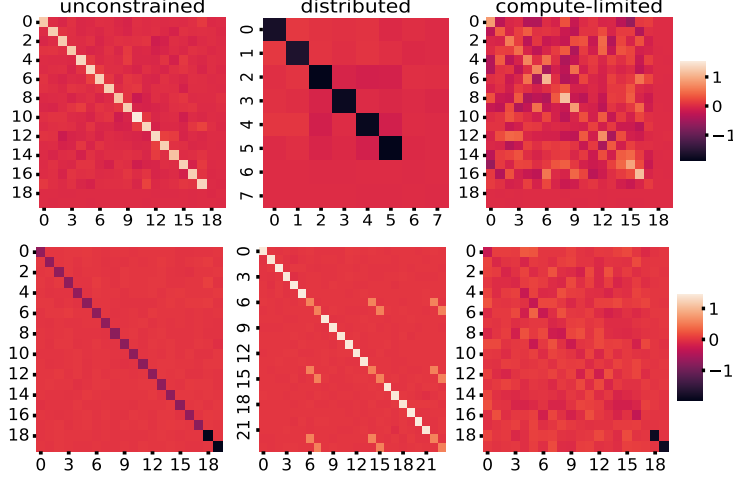


Figure 3: Block structure of  $W_{QK,\ell}$  (top) and  $W_{VP,\ell}$  (bottom) learned in the three regimes. Example shown for  $(n, d, d', n') = (18, 18, 2, 2)$ ; distributed run uses  $M = 3$  workers with per-worker  $n = 6$ , and all of the  $\{W_{VP}^\mu\}_{\mu=1,2,3}$  are collated together with each of the three block-wise rows corresponding to one head, while omitting all (null) non- $\mu$  blocks in  $W_{QK}^\mu$ . Off-diagonal blocks appear *only* in  $W_{VP,\ell}^\mu$ s, and are identical suggesting structure of messaging. Statistics over 10 seeds are in §C.

via the off-diagonal blocks in  $W_{VP,\ell}^\mu$ , each machine transmits only the  $O(n'(d + d'))$  entries of its update to  $C_\ell, D_\ell$ , captured in the  $C', D'$  outputs in Alg. 1. Since attention heads are simply added, every machine averages its received blocks, leading to  $C_{\ell+1} = M^{-1} \sum_\mu C'^\mu$ ,  $D_{\ell+1} = M^{-1} \sum_\mu D'^\mu$ . The shared columns are thus identical across all  $\mu$  at all  $\ell$ , recovering Alg. 1 with  $S = I_n$  and  $M > 1$ .

• *Communication cost.* Under the star topology in Alg. 1 each machine communicates  $O((d + d') n')$  floats per round; for scalar prediction ( $d' = n' = 1$ ) this is the advertised  $2d$ -float message. Notably, the transformer recovers this sparse pattern *without* any explicit communication constraint. Exploring how stronger topological or bandwidth limits shape the learned algorithm is an open problem.

**Computation-limited setting.** Here the query, key, value and projection matrices are explicitly rank-constrained:  $W_{Q,\ell}, W_{K,\ell} \in \mathbb{R}^{(d+d') \times r}$  and  $W_{V,\ell}, W_{P,\ell} \in \mathbb{R}^{(d+d') \times (r+n')}$  with  $r \ll n$ . Similar low-rank constraints commonly arise when attention head dimensions are smaller than the transformer’s overall embedding size.

• *Emergent weight structure.* Again, only one head per layer is needed. The  $(n + n') \times (n + n')$  matrices  $W_{QK,\ell}$  and  $W_{VP,\ell}$  share three universal properties (Fig. 3)

– *Block-diagonal form.* Both are block-diagonal; the cross blocks  $(n \times n')$  and  $(n' \times n)$  vanish as in the unconstrained regime. The  $(n' \times n')$  block of  $W_{VP,\ell}$  is a scaled identity, exactly as before.

– *Rank- $r$  top left block.* The leading  $n \times n$  block of each matrix has numerical rank  $r (< n)$  and the two blocks are similar up to a sign when rescaled to spectral norm 1 (relative difference: 0.28).

– *Random sketch spectrum.* The eigenvalues and entry distribution of the leading block match those of  $SS^\top$  where  $S$  is a random  $n \times r$  orthogonal matrix (Fig. 4).

• *Sketching interpretation.* The transformer therefore materializes an *orthogonal row sketch*  $S \in \mathbb{R}^{n \times r}$  within the ‘top left’ blocks of its  $W_{QK}$  and  $W_{VP}$  matrices. This sketch acts upon the columns within the  $(A_\ell, B_\ell)$  blocks of the state  $Z_\ell$ , and the output of the attention block is structured as

$$\text{Attn}_\ell(Z) = \begin{bmatrix} \alpha^1 (AS)(AS)^\top (AS)S^\top & \alpha^2 (AS)(AS)^\top C \\ \alpha^1 (BS)(AS)^\top (AS)S^\top & \alpha^2 (BS)(AS)^\top C \end{bmatrix}, \quad (3)$$

where  $\alpha^1, \alpha^2$  are constants depending on  $\|A\|^{-2}$ . In other words, the update first computes the sketches  $AS, BS \in \mathbb{R}^{d \times r}$  of the ‘complete’ columns, and then proceeds with the update as in the unconstrained case, lifting them back to  $n \times d$  by the terminal  $S^\top$ . Setting  $M = 1$  and  $S = S_\ell$  in Algorithm 1 recovers the exact layer dynamics.

• *What the sketch buys.* The contraction still needs  $\rho_\ell = \|\tilde{A}_\ell\|_2^{-2}/3$  but now  $\tilde{A}_\ell = AS$  is only  $r$  columns wide, reducing the per-layer flop count from  $O(n^2 d)$  to  $O(nrd)$ . Although this sketched update causes the iteration complexity of the method to increase, the overall hope is that the lower per-iteration cost may translate into a better total runtime. This aligns with the original motivation in Vaswani et al. (2017), where low-rank constraints on  $W_{Q,\ell}, W_{K,\ell}, W_{V,\ell}$ , and  $W_{P,\ell}$  were explicitly introduced for computational efficiency.



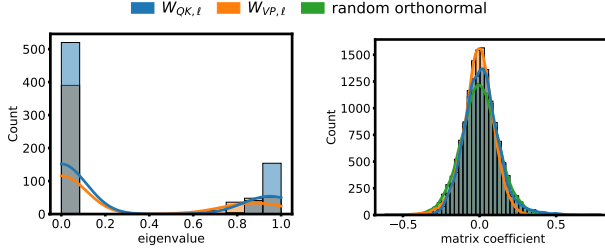


Figure 4: The computation-limited transformer implements pseudo-random sketching. The figure matches the candidate sketch matrices across all layers against common sketch characteristics (randomness, clustered eigenvalues). Left: Distribution of eigenvalues. Right: Distribution of coefficients.

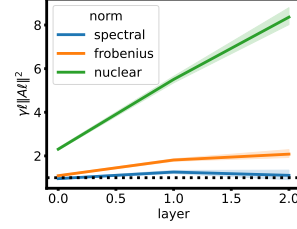


Figure 5: The transformer learns to normalize the batch-maximum spectral norm of the iterate  $A_\ell$  observed during training. The plot reports  $\alpha_\ell \beta_\ell \max_{b \in [B]} \|A_\ell^{(b)}\|^2$ , where the maximum is taken over each batch. Results are averaged over 10 training seeds.

## 233 5 Evaluation of EAGLE

234 Having identified the EAGLE update underlying the transformer weights, we move on to studying  
 235 how well the extracted algorithm performs in the three regimes consider, both theoretically and  
 236 empirically. All ablations are deferred to §E, while theoretical proofs appear in §F. Unless stated  
 237 otherwise, all numerical evaluations in this section use data sizes  $n = d = 240$  and  $n' = d' = 2$ ,  
 238 condition number  $\kappa(A) = 10^2$ ,  $\text{rank}(A) = 240$  and average performance across 50 runs is reported.

### 239 5.1 Unconstrained (centralized) setting

240 **Second-order convergence guarantee.** Let  $\kappa(M)$  denote the condition number of a matrix  $M$ .

241 **Theorem 1.** *For any  $X$ , let  $\hat{D}_*$  be the Nyström estimate for  $D$ . If  $\eta = 1/3, \gamma = 1$ , then under*  
 242 *EAGLE with  $S = I_n, M = 1$ , for any  $\varepsilon > 0$ , there exists*

$$L = O(\log \kappa + \log \log(\varepsilon^{-1} \sqrt{d'} \|W_*\|_F \|C\|_F))$$

243 *such that  $\forall \ell \geq L, \|D_\ell - \hat{D}_*\|_F \leq \varepsilon$ , where  $W_* = BA(AA^\top)^\dagger$  is the Nyström parameter (§B.1).*

244 **Mechanism.** We show Thm. 1 in §F by arguing that each iteration shrinks  $\kappa_\ell := \kappa(A_\ell)$  by at least a  
 245 constant factor, and that once  $\kappa_\ell \leq 2$ , then  $\kappa_\ell - 1$  decays supergeometrically. Error is controlled by  
 246 developing a telescoping series with terms decaying with  $(\kappa_\ell - 1)$ . The  $\log \log(\varepsilon^{-1})$  dependence in  
 247  $L$ , i.e., the quadratic rate, is related to the quadratic decay in  $\kappa_\ell - 1$ , which in turn arises since the  
 248 EAGLE iterations for  $A_\ell$  bear a strong relationship to the classical Newton-Schulz method (Higham,  
 249 2008, Ch. 5.7). We note that a simple  $\|D_\ell - D_{\ell-1}\|_F < \tau$  stopping rule suffices in practice.

250 **Positioning vs. classics.** Direct inversion via QR-decomposition or SVD costs  $O(\min n^2 d, d^2 n)$   
 251 once, independent of  $\kappa$ . Krylov methods such as the Conjugate Gradient method (CG, Hestenes and  
 252 Stiefel, 1952) and gradient descent (GD) need  $\kappa \log(\varepsilon^{-1})$  and  $\kappa^2 \log(\varepsilon^{-1})$  iterations and  $O(nd)$  time  
 253 per-iteration to compute matrix–vector products. By contrast, EAGLE achieves quadratic convergence  
 254 with iteration counts scaling only with  $\log(\kappa)$ , albeit with  $\min(n^2 d, d^2 n)$  cost per iteration.

255 **Empirical protocol.** We benchmark EAGLE against a QR-based solver (`torch.linalg.lstsq`),  
 256 the Conjugate Gradient method and gradient descent (GD) on synthetic  $A \in \mathbb{R}^{n \times n}$  (see Appx. D):

- 257 1. **Error vs. iteration (log–log).**  $d = n = 240, \kappa = 10^2$ ; confirms slope  $\approx 2$  predicted by Theorem 1.
- 258 2.  **$\kappa$ -sweep.** Time to reach  $\varepsilon = 10^{-20}$  for  $\kappa \in \{10^2, \dots, 10^5\}$ ; visualises the  $\log \kappa$  vs.  $\sqrt{\kappa}$  gap.
- 259 3. **Rank-deficient check.** Time to reach  $\varepsilon = 10^{-20}$  highlights robustness to rank-deficiency.

260 Figure 7 present (1)–(3); rank-deficient and extended size sweeps are in §E.

261 **Take-away.** The transformer-extracted update converges *quadratically* with only a  $\log \kappa$  penalty—  
 262  $\sim 100\times$  fewer iterations than CG at  $\kappa = 10^4$ —while retaining  $O(\min nd^3, dn^2)$ -per-iter cost. This  
 263 positions EAGLE in a previously unexplored region of the speed–accuracy trade-off: an iterative  
 264 solver that matches QR-based performance up to a log factor.

### 265 5.2 Distributed setting

266 **Diversity drives the rate.** Besides the local condition numbers  $\kappa^\mu = \kappa(A^\mu)$ , convergence depends  
 267 on how *distinct* the column spaces on different workers are. We capture this via:

268 **Definition 1. (Diversity index)** Normalise  $\|A^\mu\|_2 = 1$  and let  $P^\mu \in \mathbb{R}^{d \times d}$  be a projection onto the  
 269 column space of  $A^\mu$ . The diversity index of  $\{A^\mu\}_{\mu=1}^M$  is defined as

$$\alpha := \min_{\|v\|=1} M^{-1} \sum_{\mu \in [1:M]} \|P^\mu v\|_2^2 \in (0, 1].$$

270 A large overlap in subspaces captured by the distinct machines  $A^\mu$  causes a reduction in  $\alpha$ , and  
 271 orthogonal subspaces induce  $\alpha = 1$ .  $\alpha$  captures the convergence scale of distributed EAGLE via

272 **Theorem 2.** Let  $\kappa_{\max} = \max_\mu \kappa^\mu$ . In the noiseless case low rank case, i.e., when  $\exists W_*$  such that  
 273  $[B \ D] = W_* [A \ C]$ , EAGLE with  $\eta = 1/3, \gamma = 1$  ensures that for all  $\varepsilon > 0$ , there exists

$$L = O(\log(\kappa_{\max} + \alpha^{-1} \log(\sqrt{d'} \|C\|_F \|W_*\|_F / \varepsilon)) \text{ such that } \ell \geq L \implies \|D_L - D\|_F \leq \varepsilon.$$

274 **Mechanism.** After  $\log \kappa_{\max}$  iterations, every local  $A_\ell^\mu$  has near-unit singular spectrum. Subsequent  
 275 progress is controlled by the condition number of the average energy matrix  $\bar{E}_\ell = M^{-1} \sum_\mu A_\ell^\mu A_\ell^{\mu \top}$ ,  
 276 which tends to  $\alpha^{-1}$ . Second-order convergence re-emerges when  $\alpha = 1$ . Along with proving the  
 277 above result, we also show a noisy analogue of it with the same convergence scale  $L$  in §F.

278 **Baselines.** QR decompositions and the Conjugate Gradient  
 279 method are unavailable in the distributed regime limited to  
 280  $O(d)$  communications, leaving GD as the main competitor.  
 281 Both GD and EAGLE have the same communication costs,  
 282 transmitting  $O((d + d')n')$  floats per iteration, but the iteration  
 283 complexity of GD scales with  $\kappa(\bar{E}_0) > \alpha^{-1}$ . As a  
 284 result, EAGLE has a strong advantage in practical strongly  
 285 communication-limited scenarios.

286 **Empirical protocol.** We vary the two rate-determining param-  
 287 eters,  $M, \alpha$ , and report error against iteration in Figs. 6, 9  
 288 • **$M$ -sweep.**  $M = 1, 3, 5, 8$  workers on fixed total data ( $d =$   
 289  $n = 1000, \kappa_{\max} = 10^3, \alpha \approx 1$ ).

290 • **$\alpha$ -sweep.** Four synthetic data sets with fixed  $\kappa_{\max}$  and  $\alpha \in \{1, 0.9, 0.36, 0.004\}$ .

291 **Take-away.** EAGLE reaches  $\varepsilon = 10^{-2}$  in  $\leq 15$  iterations for every  $M \in \{1, 3, 5, 8\}$  when the  
 292 worker subspaces are nearly orthogonal ( $\alpha \approx 1$ ), and its iteration count grows exactly linearly with  
 293  $\alpha^{-1}$  as overlap increases (Fig. 9). This confirms our theoretical result. In contrast first-order GD  
 294 require 10–100 $\times$  more rounds. Noise experiments and a “best-of-both” hybrid (EAGLE until  $A$   
 295 stabilizes, then GD) are detailed in §E.

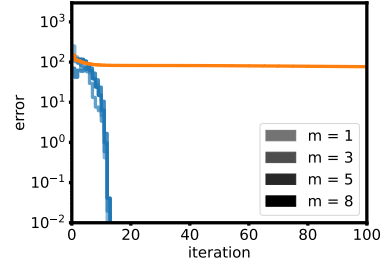


Figure 6: Iteration complexity of distributed EAGLE is independent of the number of workers  $M$ .

### 296 5.3 Computation-limited sketching

297 **Sketch-aware guarantee.** We now turn to iterations driven by the sketched columns  $\tilde{A}_\ell :=$   
 298  $S_\ell^\top A, \tilde{B}_\ell := S_\ell^\top B$  via the i.i.d. orthogonal sketches  $S_\ell \in \mathbb{R}^{n \times r}$  ( $r \ll n$ ).

299 **Theorem 3.** Let  $\hat{D}_*$  denote the Nyström approximation of  $D$ . Suppose we run EAGLE with  $M = 1$ ,  
 300 i.i.d. orthogonal sketches  $S \in \mathbb{R}^{n \times r}, \rho = \|A_\ell\|_2^{-2}, \eta = (n + 2)/3(r + 1)$  and  $\gamma = (n + 2)/(r + 1)$ .  
 301 Then with probability at least  $1 - \delta$ , for any  $\varepsilon > 0$ , there exists

$$L = O\left(\frac{n}{r} \log(\kappa(A)) + \log \frac{1}{\delta} + \log \log \frac{\|B\|_F \|C\|_F}{\varepsilon}\right) \text{ such that } \forall \ell \geq L, \|D_L - \hat{D}_*\|_F \leq \varepsilon.$$

302 **Mechanism.** The isotropicity of the sketch ensures that the singular-spectrum of  $A$  is conditioned  
 303 in roughly the same as the centralized setting (§5.1), up to a slowdown of a  $\approx r/n$  factor. As in  
 304 this previous setting, once  $\kappa_\ell \leq 2$ , the same quadratic Newton-Schulz phase takes over, leaving  
 305  $n/r \log(\kappa)$  as the main convergence scale with otherwise quadratic convergence.

306 **Baselines.** The standard competitor is Stochastic Gradient Descent (SGD): GD run on an  $r$ -column  
 307 sketch. It shares the  $n/r$  spectral slowdown but keeps its native  $\kappa^2$  dependence, so Theorem 3 predicts a  
 308 multiplicative  $\kappa^2 / \log \kappa$  advantage in total iterations for EAGLE on ill-conditioned data.

309 **Empirical protocol.** We vary the rank, the single rate-controlling parameter  $r$ :

- 310 1.  **$r$ -rank sweep.** We vary the rank  $r \in \{n/8, n/4, n/2, n\}$  on a  $242 \times 242$  matrix with  $\kappa = 10^2$ .
- 311 2. **Per-iteration cost.** The per-iteration wall-clock time decreases with sketch size: 21, 16, 7 and 5  
 312 milliseconds for  $s = 240, 120, 60$  and 30, respectively.

313 Noise robustness ( $\sigma^2 = 0.1$ ) and step-size sensitivity curves are in §E.



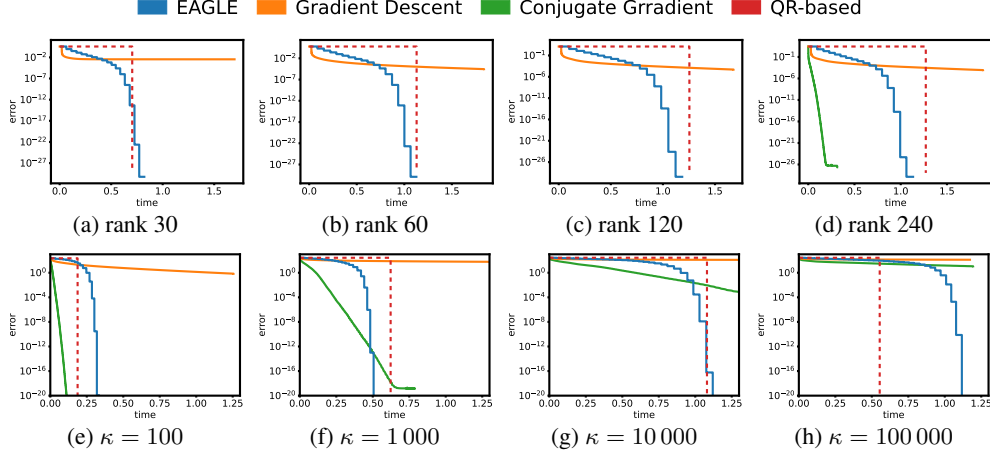


Figure 7: EAGLE shows second order convergence in the unconstrained regime. It extends to low-rank completion and scales logarithmically with the condition number  $\kappa$ .

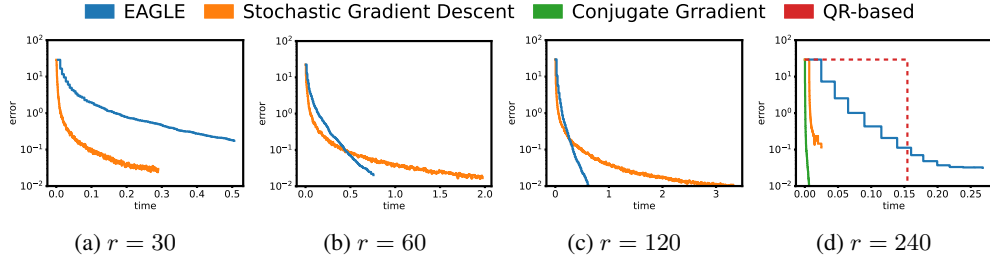


Figure 8: In the stochastic data access regime—a variant of the compute-limited setting—EAGLE outperforms SGD in wall-clock time for sketch sizes  $r \geq n/4$ .

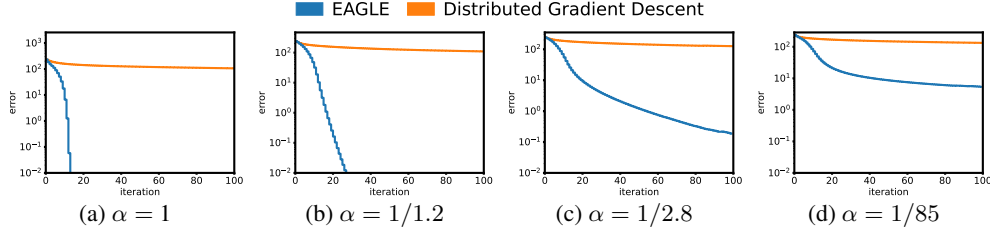


Figure 9: In the distributed setting, EAGLE shows significant improvements in iteration count. As the theory suggest, the convergence depends on the data alignment parameter  $\alpha$ .

**Take-away.** Iteration count grows linearly with  $n/r$ , matching Theorem 3, while time per iter falls up to  $7\times$ . Against RandSketch+CG the method reaches  $\varepsilon=10^{-6}$  in roughly one third the wall-clock for  $\kappa = 10^4$ , demonstrating that sketching *plus* the second-order update preserves the speed advantage of the unconstrained regime even under tight compute budgets.

## 6 Conclusions

We view transformers as fixed data-to-data transforms whose forward pass exposes emergent algorithms. We fix one low-rank matrix-completion task and explore various regimes. Our proposed rule, EAGLE is a transformer-induced method that emerges as a unifying algorithm in centralized, distributed and sketching regimes, achieving second-order convergence with only  $\log \kappa$  (where  $\kappa$  is the condition number),  $\alpha^{-1}$  (distributed data diversity) or  $n/r$  (sketch) slow-downs. Empirically, it beats Conjugate Gradient and Gradient Descent by 1–2 orders-of-magnitude in both iterations and communicated bytes and matches QR-based solvers up to log terms, offering a practical drop-in solver when memory or bandwidth is tight. *Limitations.* Numeric stability beyond  $\kappa > 10^8$  is untested. Second, we show one pre-training recipe that yields EAGLE; extensions to non-linear regimes that do so are open.

## References

- Ahn, K., Cheng, X., Daneshmand, H., and Sra, S. (2023). Transformers learn to implement pre-conditioned gradient descent for in-context learning. *Advances in Neural Information Processing Systems*, 36:45614–45650.
- Akyürek, E., Schuurmans, D., Andreas, J., Ma, T., and Zhou, D. (2022). What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*.
- Bai, Y., Chen, F., Wang, H., Xiong, C., and Mei, S. (2023). Transformers as statisticians: Provable in-context learning with in-context algorithm selection. *Advances in neural information processing systems*, 36:57125–57211.
- Ben-Israel, A. (1965). An iterative method for computing the generalized inverse of an arbitrary matrix. *Mathematics of Computation*, pages 452–455.
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33.
- Candès, E. J. and Recht, B. (2009). Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772.
- Candès, E. J. and Tao, T. (2010). The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053–2080.
- Chen, Y., Fan, J., Ma, C., and Wang, K. (2020). Communication-efficient distributed statistical inference. *Journal of the American Statistical Association*, 115(530):883–896.
- Clarkson, K. L. and Woodruff, D. P. (2017). Low-rank approximation and regression in input sparsity time. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC ’17*, pages 81–90.
- Daneshmand, H. (2024). Provable in-context optimal transport with transformers. In *arXiv*.
- Davenport, M. A. and Romberg, J. (2016). An overview of low-rank matrix recovery from incomplete observations. *IEEE Journal of Selected Topics in Signal Processing*, 10(4):608–622.
- Demmel, J., Grigori, L., Hoemmen, M., and Langou, J. (2013). Communication-optimal parallel and sequential krylov subspace methods. *SIAM Journal on Scientific Computing*, 34(1):A206–A239.
- Fu, D., Chen, T.-q., Jia, R., and Sharan, V. (2024). Transformers learn to achieve second-order convergence rates for in-context linear regression. *Advances in Neural Information Processing Systems*, 37:98675–98716.
- Garg, S., Tsipras, D., Liang, P., and Valiant, G. (2023). What can transformers learn in-context? a case study of simple function classes.
- Giannou, A., Rajput, S., Sohn, J.-y., Lee, K., Lee, J. D., and Papailiopoulos, D. (2023). Looped transformers as programmable computers. In *International Conference on Machine Learning*, pages 11398–11442. PMLR.
- Gittens, A. and Mahoney, M. W. (2016). Revisiting the nyström method for improved large-scale machine learning. *Journal of Machine Learning Research*, 17:1–65.
- Halko, N., Martinsson, P., and Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288.
- Hestenes, M. R. and Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49(6):409–436.

Higham, N. J. (1997). Stable iterations for the matrix sign function. *SIAM J. Matrix Analysis and Applications*, 18(1):274–291.

Higham, N. J. (2008). *Functions of matrices: theory and computation*. SIAM.

Hoefler, T., Huber, M., Luitjens, J., and Lumsdaine, A. (2019). Sparsity-aware communication schemes for distributed krylov solvers. In *Proceedings of SC '19*, pages 1–14.

Ma, C. and Chen, Y. (2020). Communication-efficient distributed statistical estimation with optimal convergence rate. In *Proceedings of Machine Learning Research (PMLR)*, volume 108, pages 3449–3459.

Meckes, E. S. (2019). *The random matrix theory of the classical compact groups*, volume 218. Cambridge University Press.

Min, S., Lewis, M., Zettlemoyer, L., and Hajishirzi, H. (2021). Metaicl: Learning to learn in-context. In *ACL Findings*.

Musco, C. and Musco, C. (2017). Recursive sampling for the nyström method. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Naim, O. and Asher, N. (2025). Two in context learning tasks with complex functions. *arXiv preprint arXiv:2502.03503*.

Nanda, N. (2023). Progress in mechanistic interpretability: Reverse-engineering induction heads in GPT-2.

Olah, C., Elhage, N., Nanda, N., Schiefer, N., Jones, A., Henighan, T., and DasSarma, N. (2022). Transformer circuits. *Distill*, 7(3). <https://distill.pub/2022/circuits/>.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Saad, Y. (2003). *Iterative Methods for Sparse Linear Systems*. SIAM, 2 edition.

Schulz, G. (1933). Iterative berechnung der reziproken matrix. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 13(1):57–59.

Siegelmann, H. T. and Sontag, E. D. (1992). On the computational power of neural nets. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 440–449.

Tong, W. L. and Pehlevan, C. (2024). Mlps learn in-context on regression and classification tasks. *arXiv preprint arXiv:2405.15618*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*.

Vladymyrov, M., Von Oswald, J., Sandler, M., and Ge, R. (2024). Linear transformers are versatile in-context learners. *Advances in Neural Information Processing Systems*, 37:48784–48809.

Von Oswald, J., Niklasson, E., Randazzo, E., Sacramento, J., Mordvintsev, A., Zhmoginov, A., and Vladymyrov, M. (2023). Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR.

Wang, J., Blaser, E., Daneshmand, H., and Zhang, S. (2025). Transformers can learn temporal difference methods for in-context reinforcement learning. In *The Thirteenth International Conference on Learning Representations*.

Williams, C. K. I. and Seeger, M. (2001). Using the nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, NeurIPS 13, pages 682–688.

Woodruff, D. P. (2014). Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 10(1–2):1–157.

Xie, S. M., Raghunathan, A., Liang, P., and Ma, T. (2021). An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*.

## 1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [Yes]

Justification: [NA]

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

## 2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: [NA]

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

## 3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: [NA]

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

#### 4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: [NA]

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

#### 5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: The simulations and transformer training is lightweight and easy to implement.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

**6. Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: [NA]

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

**7. Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: [NA]

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.



- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

## 8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: [NA]

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

## 9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: [NA]

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

## 10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: [NA]

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

## 11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: [NA]

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

## 12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [NA]

Justification: [NA]

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, [paperswithcode.com/datasets](https://paperswithcode.com/datasets) has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

## 13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: [NA]

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

#### 14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: [NA]

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

#### 15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: [NA]

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

#### 16. Declaration of LLM usage

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [NA]

Justification: [NA]

Guidelines:

- 733 • The answer NA means that the core method development in this research does not  
734 involve LLMs as any important, original, or non-standard components.
- 735 • Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>)  
736 for what should or should not be described.

738 **Neural networks as implicit algorithm learners.** Large models can *emerge* classical iterative  
 739 procedures when trained on synthetic tasks. For transformers this spans gradient descent for least  
 740 squares (Von Oswald et al., 2023; Ahn et al., 2023), dual gradient descent for optimal transport  
 741 (Daneshmand, 2024), and temporal-difference planning in RL (Wang et al., 2025). RNNs and MLPs  
 742 exhibit similar behavior (Siegelmann and Sontag, 1992; Tong and Pehlevan, 2024), though attention  
 743 yields depth-aligned, inspection-friendly updates (Garg et al., 2023). **Gap.** All of these works uncover  
 744 *first-order* rules tied to a single regime. **Our advance.** EAGLE is the first *second-order* update that  
 745 generalizes *unchanged* across centralized, distributed, and sketched settings.

746 **Data transformers vs. parameter updates.** Unlike classical solvers that iterate in *parameter space*,  
 747 a transformer’s computation is purely *data-to-data*: weights stay fixed at inference time, and all  
 748 state lives inside the forward activations. Prior analyses of in-context learning focus on mapping this  
 749 data flow to gradient descent on hidden parameters (Akyürek et al., 2022; Von Oswald et al., 2023).  
 750 EAGLE shows a stronger result—an *explicit, second-order data-space projector* that converges  
 751 quadratically without ever touching model parameters. This clarifies that algorithmic power can  
 752 emerge *without* an internal parameter update loop.

753 **In-context learning as regression—rates and limits.** Theoretical analyses cast in-context learning  
 754 as linear regression on prompt examples (Akyürek et al., 2022; Bai et al., 2023; Min et al., 2021).  
 755 Transformers so far reproduce gradient descent or SGD, yielding linear rates and the usual  $\sqrt{\kappa}$   
 756 condition-number dependence. **Our advance.** We extract an *emergent* Newton–Schulz-style update  
 757 with quadratic convergence and  $\log \kappa$  dependence that remains numerically stable under three  
 758 disparate resource constraints.

759 **Classical second-order solvers, Nyström, and low-rank approximation.** Newton–Schulz iterations  
 760 deliver quadratic convergence for matrix inversion (Higham, 1997) but require explicit matrix products  
 761 and have not been analyzed in noisy, data-dependent regimes. Nyström methods scale kernel learning  
 762 and matrix completion (Williams and Seeger, 2001; Halko et al., 2011; Gittens and Mahoney, 2016);  
 763 state-of-the-art variants use QR or Krylov subspace solvers (Saad, 2003; Musco and Musco, 2017).  
 764 **Gap.** These algorithms are hand-designed for fixed settings and degrade to first-order ( $\sqrt{\kappa}$ ) rates  
 765 when run under communication or memory limits. **Our advance.** The transformer learns the  
 766 same Newton–Schulz-type projector that attains Nyström accuracy in only  $\log \kappa$  steps and *adapts*  
 767 *automatically* to distributed or sketched execution.

768 **Distributed and sketched solvers.** Block-CG, communication-optimal Krylov methods (Demmel  
 769 et al., 2013; Hoeffer et al., 2019) and randomized sketch-and-solve techniques (Clarkson and Woodruff,  
 770 2017; Woodruff, 2014) are the de-facto choices at scale, yet remain first-order and  $\sqrt{\kappa}$ -limited. We  
 771 introduce a *diversity index*  $\alpha$  that predicts our distributed rate and achieves provably fewer rounds  
 772 when worker subspaces overlap ( $\alpha^{-1} \ll \sqrt{\kappa}$ ).

773 **Meta-learning across tasks.** Transformers pretrained on heterogeneous corpora behave as meta  
 774 learners (Min et al., 2021; Bai et al., 2023; Naim and Asher, 2025); prior work, however, stops at  
 775 behavioral observation. EAGLE offers a mechanistic account: pre-training can imprint a *single*  
 776 algorithm that flexes with compute, memory, and communication budgets.

777 **Scope.** This survey is necessarily selective; we focus on works most relevant to the *data-space*,  
 778 *second-order* perspective. A broader taxonomy of numerical solvers and in-context phenomena is  
 779 beyond our scope but complementary to the questions addressed here.

780 **Positioning.** To our knowledge this is the first demonstration that pre-training alone can induce *one*  
 781 second-order, matrix-completion rule that (i) bridges centralized, distributed, and sketch regimes,  
 782 (ii) admits tight non-asymptotic guarantees, and (iii) out-iterates classical Krylov baselines *and* cuts  
 783 communication under typical ML-scale conditioning.

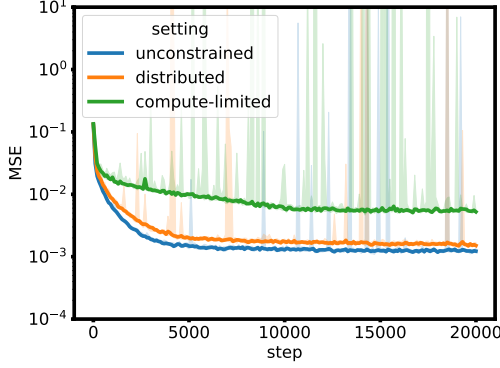


Figure 10: Training loss, median across 10 independent runs.

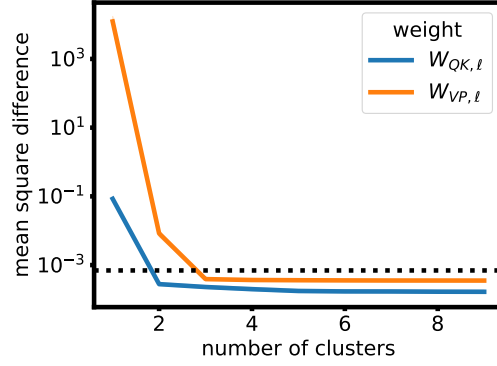


Figure 11: Prediction difference when sparsifying weights in centralized setting. For reference, the black dotted line marks the trained transformer’s mean squared error. Mean across 10 training runs reported.

## 784 B Methodological Details

### 785 B.1 Nyström Approximation

786 The Nyström approximation of a block matrix  $X = \begin{bmatrix} A & C \\ B & D \end{bmatrix}$  with missing block  $D$  is the matrix

$$\hat{X}_* = \begin{bmatrix} A & C \\ B & \hat{D}_* \end{bmatrix},$$

787 where

$$\hat{D}_* := BA(AA^\top)^\dagger C.$$

788 Here,  $Z^\dagger$  is the Moore-Penrose pseudoinverse of  $Z$ . Note that this approximation ensures that the  
 789 rank of the approximate  $\hat{X}_*$  is the same as the rank of  $A$ —in this way, the Nystrom approximation  
 790 reduces both the rank of  $X$ , as well as the Frobenius norm of the error. The latter property can  
 791 be seen from the fact that the approximation just amounts to a joint execution of ordinary least squares.  
 792 Indeed, if we treat the  $i$ th row of  $B$ ,  $B^i$  as a ‘response variable’ and the  $j$ th column of  $C$ ,  $c^j$  as a  
 793 query point, then

$$(\hat{D}_*)_{ij} = B^i A(AA^\top)^\dagger c^j$$

794 is precisely the estimate when one regresses the data  $(A, B^i)$  onto  $c^j$ . For this reason, we may also  
 795 interpret the method as computing the estimation parameters

$$W_* := BA(AA^\top)^\dagger \in \mathbb{R}^{d' \times d},$$

796 and then computing the estimate

$$\hat{D}_* = W_* C,$$

797 much as in ordinary linear regression. This value  $W_*$  will appear later in our theoretical analyses.

798 We note that the approximation above is defined whether the matrix  $X$  is noise-corrupted or not.  
 799 In the presence of noise, it inherits many of the statistical properties of linear regression when we  
 800 assume that the noise is restricted to the ‘response variables’  $B, D$ . As a special case, if the rank  
 801 of  $X$  is equal to that of  $\begin{bmatrix} A & C \end{bmatrix}$ , then in fact it holds that  $\begin{bmatrix} B & D \end{bmatrix} = W_* \begin{bmatrix} A & C \end{bmatrix}$ , i.e., the Nyström  
 802 approximation is exact. We will sometimes refer to this as the ‘noiseless’ case.

### 803 B.2 Training specifications

804 The transformer is trained using the Adam optimizer with a constant learning rate of 0.001 and  
 805 a batch size of 1024 for 20,000 iterations. To stabilize training, gradient clipping with a 2-norm  
 806 threshold of 0.1 is applied at each step. At every iteration, a new batch is independently sampled



as detailed in Section 3, ensuring that the model never sees the same data twice during training or inference. The block masking pattern remains fixed throughout. We observe occasional spikes in the training loss (Figure 10) and interpret those as occasional failures of the learned solver to converge, typically occurring when  $\|A\|_2$  attains untypically large values. We recall that, under our data sampling procedure,  $\|A\|_2$  is random and unbounded, allowing for such outlier cases.

### B.3 Runtime of attention map

Recall that the dominant computational cost in a transformer layer arises from the attention mechanism:

$$\text{Attn}(Z) = (ZW_Q(ZW_K)^\top \odot M)ZW_VW_P^\top,$$

where  $Z \in \mathbb{R}^{(d+d') \times (n+n')}$  is the input matrix and  $W_Q, W_K, W_V, W_P \in \mathbb{R}^{n \times k}$  are learned projection matrices. Assuming that the size of the submatrix  $D$  remains constant, we have  $d' = \Theta(1)$  and  $n' = \Theta(1)$ .

The computation of  $ZW_Q, ZW_K$ , and  $ZW_V$  requires  $\Theta(ndk)$  time. The inner product and masking operation  $(ZW_Q)(ZW_K)^\top \odot M$  incurs a cost of  $\Theta(d^2k)$ , and the subsequent multiplication with  $ZW_VW_P^\top$  requires an additional  $\Theta(d^2k + ndk)$ . Thus, the total time complexity of the attention computation is  $\Theta(d^2k + ndk)$ .

For the unconstrained case  $k = n$ , this yields a cost of  $\Theta(nd^2 + dn^2)$ . Therefore, selecting  $k < n$  can significantly reduce computational overhead.

Of course, in our recovered algorithms, we can ‘precompute’  $W_QW_K^\top$  and  $W_VW_P^\top$ , and these are further scaled versions of block-identity matrices, and so the computation of  $ZW_Q$  etc. can be avoided. This reduces the cost to  $O(d^2k)$ , which for the unconstrained setting of  $k = n$  works out to  $O(d^2n)$ .

In the sketched versions of the iterations, these matrices become nontrivial again, and their multiplication must be incorporated into the accounting of the costs of the recovered algorithm. Of course, in this regime,  $k = r \ll n$ , so the dominating cost is  $O(ndr)$ .

### B.4 Details on setup for distributed setting

We design the transformer to emulate a distributed algorithm by constraining the attention mechanism. Specifically, the query, key, and value matrices are restricted to data local to a single machine, and due to symmetry across machines, a single set of transformation is shared across all attention heads. At every layer, the input data is partitioned as:

$$Z_k = [\dots \quad X_k^\mu \quad X_k^{\mu+1} \quad \dots] \in \mathbb{R}^{(d+d') \times M(n+n')}, \text{ where } X_k^\mu = \begin{bmatrix} A_k^\mu & C_k \\ B_k^\mu & D_k \end{bmatrix}, \mu \in [1 : M]$$

with blocks  $A^\mu \in \mathbb{R}^{n \times d}$  and  $B^\mu \in \mathbb{R}^{n' \times d}$  distributed across machines, while the shared blocks  $C \in \mathbb{R}^{n \times d'}$  and  $D \in \mathbb{R}^{n' \times d'}$  are replicated identically in each  $X^\mu$ . This setup reflects common scenarios where the missing block  $D$  is significantly smaller than the full matrix  $A$ , often remaining constant in size (e.g.,  $d' = n' = 1$  in regression), making the distributed partitioning both efficient and natural.

The projection matrix after attention is unconstrained, enabling unrestricted communication between heads and implicitly modeling a fully connected communication topology. This architectural design encourages local, machine-specific computation in the attention mechanism, while allowing for global coordination in the projection step. Empirically, we find that the model leverages this flexibility efficiently: although capable of learning full data sharing, it consistently limits itself to communication scaling with  $O(|D|)$  bits.

We find that transposing the per-machine data  $X^\mu$  is necessary in the distributed setup. That is, we give the transformer inputs  $Z_k = [\dots \quad X_k^{\mu, \top} \quad X_k^{\mu+1, \top} \quad \dots]$  and train it on the mean-squared error to  $D^\top$ .

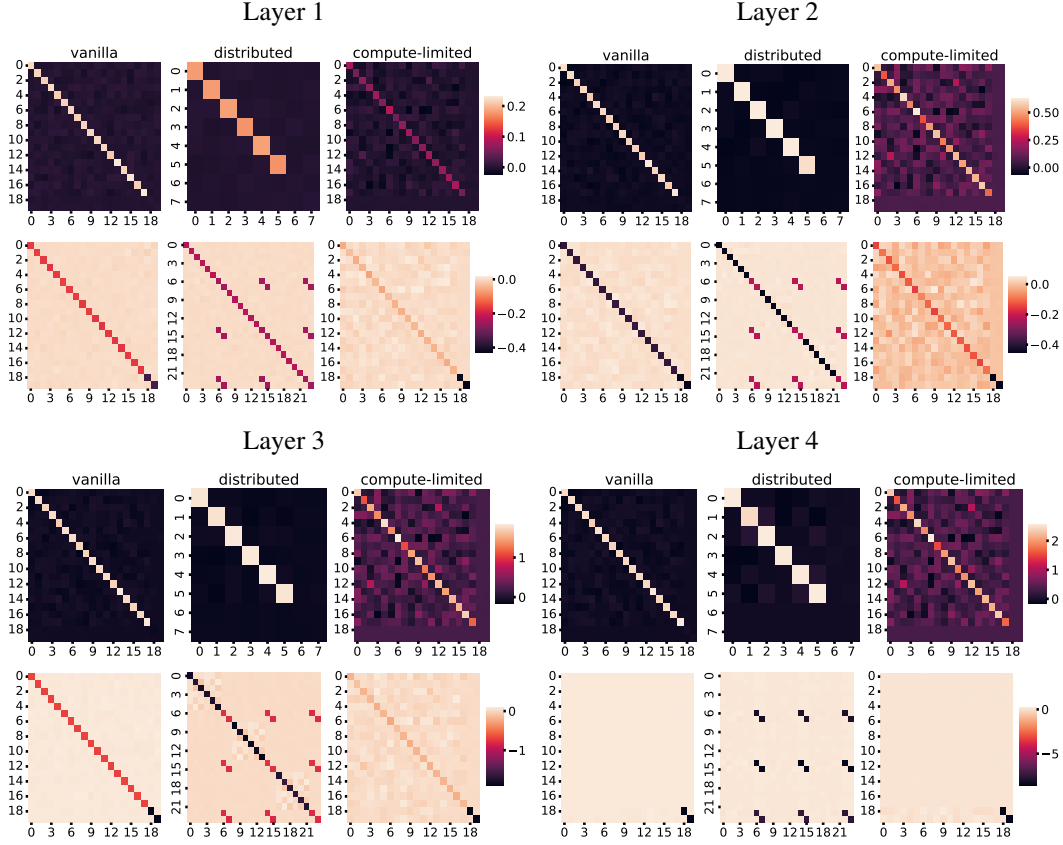


Figure 12: Transformer weights by layer, averaged over 10 training runs. The top panels display the weight products  $W_{QK, \ell}$ , and the bottom panels show  $W_{VP, \ell}$ . Prior to averaging, sign ambiguities were resolved by aligning the dominant coefficients: large entries in  $W_{QK, \ell}$  were set to be positive, and those in  $W_{VP, \ell}$  to be negative.

## 850 B.5 Details on algorithm extraction

851 We derive closed-form iterative updates from the trained transformer weights  $W_{QK, \ell}$  and  $W_{VP, \ell}$ . This  
 852 involves thresholding and clustering the weight matrices in both the unconstrained and distributed  
 853 settings:

- 854 • The sparsification threshold  $\tau$  is set dynamically as  $\tau = 1.5 \cdot \|W\|_1 / \text{\#elements}$ , accounting  
 855 for varying scales across weight matrices. The constant 1.5 is selected empirically and is  
 856 not further tuned.
- 857 • The number of quantization levels is determined by analyzing the mean prediction error  
 858 induced by weight sparsification (see Figure 11). To maintain performance comparable to  
 859 the original transformer, we cluster  $W_{QK, \ell}$  into 2 values and  $W_{VP, \ell}$  into 3.

860 In the compute-limited setting, a sketch matrix is extracted and its structural properties analyzed  
 861 (see Section 4). Parameter choices for the update steps are abstracted through empirically observed  
 862 scaling laws. Specifically, we identify  $\alpha_1 \alpha_2 \approx 1 / \|A\|_2^2$ , as supported by results in the main text.  
 863 Furthermore, we approximate  $\alpha_1 \alpha_3 \approx 1.92 \cdot \alpha_1 \alpha_3$ , where the resulting squared prediction error is  
 864  $3.5 \times 10^{-4}$  (averaged over 10 training runs), remaining below the baseline error of approximately  
 865  $1 \times 10^{-3}$ .

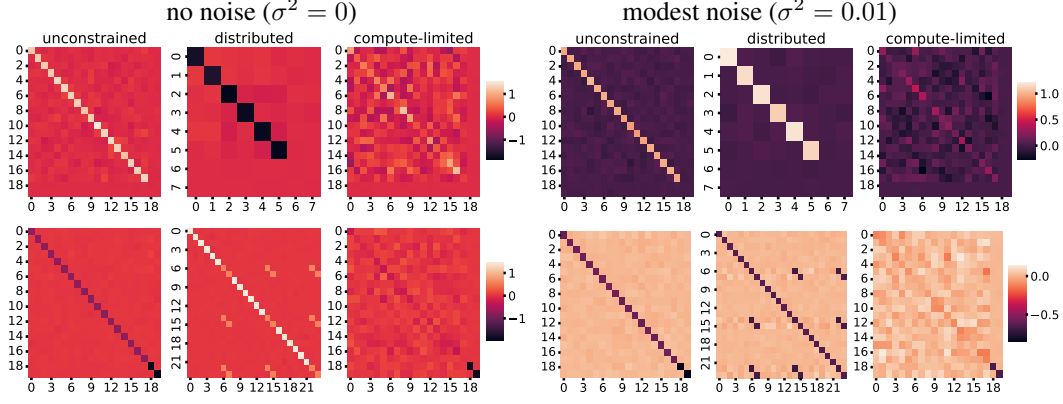


Figure 13: Comparison of learned model weights without noise vs. with modest data noise on a representative training run. The top row shows  $W_{QK,\ell}$ , and the bottom row shows  $W_{VP,\ell}$ .

## C Emergent Algorithm

### C.1 Transformer behaviour across training runs

We show that the emergent behavior identified in Section 4 is robust across training runs with varying sources of randomness, including initialization and training data. Figure 12 displays transformer weights averaged over 10 independent runs, reproducing the characteristic patterns reported in Section 4. Additionally, the sketching sub-matrix in the compute-limited setting is, on average, close to the identity, further supporting our interpretation as a random orthonormal sketch.

### C.2 Transformer behaviour under data noise

In this section, we demonstrate that the emergent behavior identified in Section 4 remains robust under moderate data noise with variance  $\sigma^2 = 0.01$ .

Figure 13 compares the learned weights of a representative transformer trained on noiseless data and on data corrupted with noise. Additionally, Figure 14 reports key diagnostic quantities across layers. In both analyses, the observed behavior closely aligns with that of the noiseless case, indicating stability of the learned algorithm under modest perturbations.

### C.3 From weights to updates

We derive the blockwise update rule implied by the abstracted weight parametrization

$$W_{QK,\ell} = \begin{bmatrix} \alpha_1 I & 0 \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad W_{VP,\ell} = \begin{bmatrix} \alpha_2 I & 0 \\ 0 & \alpha_3 I \end{bmatrix}$$

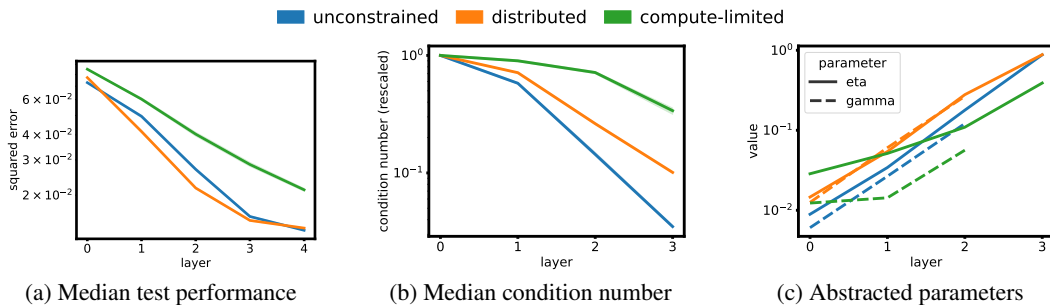


Figure 14: The trained transformer solves matrix completion (modest noise,  $\sigma^2 = 0.01$ ) with a unified algorithm over all three computational settings. The evolution of key quantities throughout the transformer layers illustrate the remarkable similarity between the latent algorithms. Mean across 10 training seeds is reported.

882 and show that it recovers the update structure presented in Section 4 for the unconstrained setting.  
 883 Recall that the layer representation takes the block form

$$Z_\ell = \begin{bmatrix} A_\ell & C_\ell \\ B_\ell & D_\ell \end{bmatrix}$$

884 and a single-head transformer layer performs the update

$$Z_{\ell+1} = Z_\ell + (Z_\ell W_{QK,\ell} Z_\ell^\top \odot M_\ell) Z_\ell W_{VP,\ell}. \quad (4)$$

885 Substituting the block structure into the attention term yields

$$Z_\ell W_{QK,\ell} Z_\ell^\top = \begin{bmatrix} \alpha_1 A_\ell A_\ell^\top & \cdot \\ \alpha_1 B_\ell A_\ell^\top & \cdot \end{bmatrix}$$

886 where the entries marked  $\cdot$  are not needed due to masking. Applying the attention mask,

$$Z_\ell W_{QK,\ell} Z_\ell^\top \odot M_\ell = \begin{bmatrix} \alpha_1 A_\ell A_\ell^\top & 0 \\ \alpha_1 B_\ell A_\ell^\top & 0 \end{bmatrix}$$

887 Further, we compute

$$Z W_{VP,\ell} = \begin{bmatrix} \alpha_2 A_\ell & \alpha_3 C_\ell \\ \alpha_2 B_\ell & \alpha_3 D_\ell \end{bmatrix}.$$

888 Multiplying these terms and adding to  $Z_\ell$ , we obtain the update:

$$Z_{\ell+1} = Z_\ell + \begin{bmatrix} \alpha_1 \alpha_2 A_\ell A_\ell^\top A_\ell & \alpha_1 \alpha_3 A_\ell A_\ell^\top C_\ell \\ \alpha_1 \alpha_2 B_\ell A_\ell^\top A_\ell & \alpha_1 \alpha_3 B_\ell A_\ell^\top C_\ell \end{bmatrix}.$$

889 This clean separation of updates across the blocks  $A_\ell$ ,  $B_\ell$ ,  $C_\ell$ , and  $D_\ell$  a posteriori motivates the  
 890 choice of block decomposition of  $Z_\ell$  used in our notation throughout the model layers.

#### 891 C.4 Interpreting the emergent algorithm

892 Prior work interprets EAGLE as gradient descent with preconditioning (Von Oswald et al., 2023; Ahn  
 893 et al., 2023; Vladymyrov et al., 2024). In particular, these works frame the method as comprising  
 894 two distinct phases: a preconditioning step involving only the updates to  $A$  and  $B$  (this is achieved  
 895 by setting  $\gamma = 0$ ), followed by a single gradient descent step using the data  $(A_\ell, B_\ell, C_0)$ . This  
 896 perspective treats the conditioning phase as auxiliary and external to the actual optimization step.

897 However, this interpretation does not accurately reflect the behavior of the trained transformer.  
 898 The model does not explicitly separate conditioning and update phases; instead, it interleaves  
 899 them continuously throughout the computation. Rather than implementing classical preconditioning  
 900 followed by gradient descent, the learned procedure functions as a continuous conditioning mechanism  
 901 that shapes the dynamics of the optimization process at every layer.

902 Moreover, this prior viewpoint underemphasizes the internal structure and complexity of the condi-  
 903 tioning dynamics themselves. In fact, the iteration bears a close resemblance to the Newton–Schultz  
 904 method for matrix inversion, suggesting a fundamentally different mechanism at play.

905 To make this connection explicit, consider the unconstrained version of the iteration. Assume  $\|A\|_2 =$   
 906 1, and set  $\eta = 1$ ,  $\gamma = 3$ , as in the statement of Theorem 1. Then, re-normalize  $\bar{A}_\ell = A_\ell / \|A_\ell\|_2$  at  
 907 every iteration. The update to the iterates  $\bar{A}_\ell$  in this setup can then be expressed in the compact form

$$\bar{A}_{\ell+1} = \frac{1}{2}(3I - \bar{A}_\ell \bar{A}_\ell^\top) \bar{A}_\ell,$$

908 which is exactly the Newton–Schultz iteration for approximate matrix inversion, as presented in  
 909 (Higham, 2008, Section 5.3). This reformulation reveals that the driving force behind the iteration is  
 910 more accurately characterized as implicit matrix inversion rather than traditional gradient descent  
 911 updates.

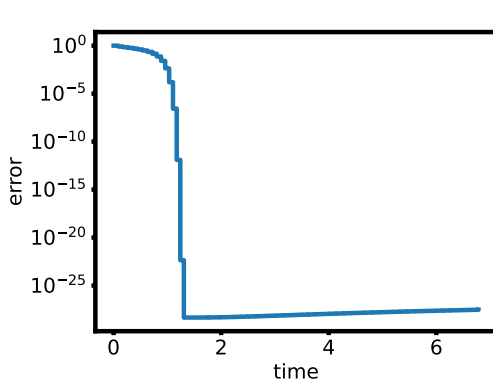


Figure 15: Convergence of EAGLE adapted to the estimation problem: unconstrained setting, with parameters  $n = d = 240$ ,  $n' = d' = 2$ ,  $\kappa(A) = 100$ . Mean over 50 runs is reported.

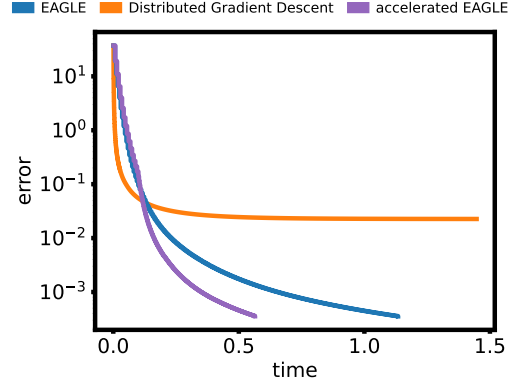


Figure 16: The accelerated EAGLE for distributed computing largely outperforms EAGLE in runtime. In this data generation, the data diversity  $\alpha \approx 0.1$  is relatively low. Mean over 50 runs is reported,  $n = d = 240$ ,  $n' = d' = 2$ ,  $\kappa(A) = 100$

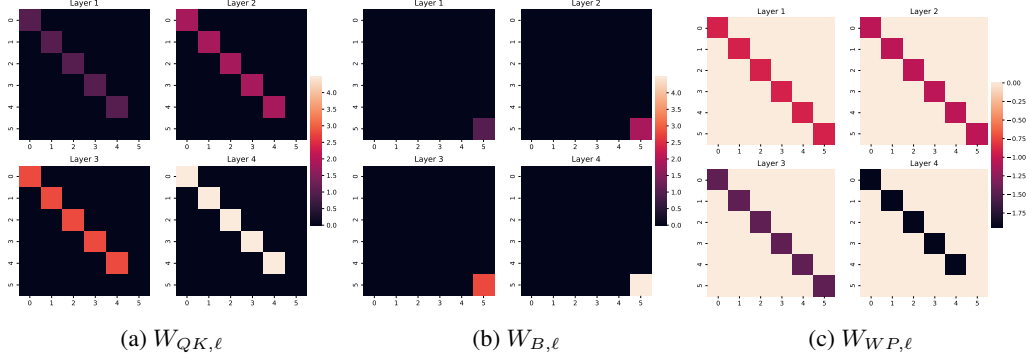


Figure 17: Learned transformer weights in the estimation setting. The transformer architecture is modified and a bias term is introduced:  $Z_{\ell+1} = Z_{\ell} + ((Z_{\ell}W_{QK,\ell} + W_{B,\ell})Z_{\ell}^{\top} \odot M_{\ell})Z_{\ell}W_{VP,\ell}$  where  $W_B \in \mathbb{R}^{n \times d}$  is a rank-2 bias term whose first  $n - 1$  rows are identical by design.

## 912 D Parameter Estimation

913 Beyond the standard prediction task in which the hidden block  $D$  is to be recovered, one may also be  
 914 interested in estimating the matrix  $W^* = BA(AA^{\top})^{\dagger}$ . In the least-squares variant of our Nyström  
 915 formulation—where  $D$  reduces to a scalar  $1 \times 1$  block—this corresponds to solving the classical  
 916 least-squares problem. While it is straightforward to recover  $W^*$  by setting  $C = I$  in EAGLE  
 917 (Algorithm 1), so that the prediction becomes  $BA(AA^{\top})^{\dagger}C = BA(AA^{\top})^{\dagger} = W^*$ , this approach is  
 918 inefficient in terms of memory and runtime. It is thus natural to ask whether the estimation task can  
 919 be achieved more efficiently, with reduced overhead.

920 To investigate this question, we consider an adapted transformer architecture trained on a matrix  
 921 completion problem posed as

$$Z = \begin{bmatrix} A^{\top} & B^{\top} \\ ? & 0 \end{bmatrix},$$

922 where the transformer is tasked with filling the missing block “?” with  $W^* = BA(AA^{\top})^{\dagger}$ . We  
 923 find that a minor architectural modification is necessary to achieve competitive performance on this  
 924 estimation task. Specifically, we introduce two biases to the query transformation. The first bias is  
 925 applied when updating the first  $n$  tokens, and the second is applied when updating the last  $n'$  tokens.

926 We train a transformer on the least-squares setting (i.e., with  $D \in \mathbb{R}^{1 \times 1}$ ) and extract the learned  
 927 weights, shown in Figure 17. This leads to a modified estimation version of EAGLE, defined by the  
 928 following iterative updates:

$$\begin{aligned} A_{\ell+1} &= A - \eta \rho_\ell A_\ell A_\ell^\top A_\ell, \\ B_{\ell+1} &= B - \eta \rho_\ell B_\ell A_\ell^\top A_\ell, \\ W_{\ell+1} &= W_\ell - \gamma \rho_\ell (W_\ell A_\ell - B_\ell) A_\ell^\top. \end{aligned}$$

929 These updates can be adapted to all computational settings considered in the main body. Importantly,  
 930 this iteration is theoretically equivalent to the original EAGLE algorithm in the sense that  $W_k C = D_k$ ,  
 931 where  $W_k$  evolves according to the update rule above and  $D_k$  evolves according to EAGLE. This  
 932 equivalence can be formally shown by expanding the recursive updates and applying a telescoping  
 933 argument, as in the proof in § F.

934 Figure 15 provides empirical evidence that the proposed estimation iteration (with the same parameter  
 935 setup  $\rho_\ell = 1/(3\|A_\ell\|_2^2)$ ,  $\eta = 1$ ,  $\gamma = 3$ ) converges as expected and recovers the same second order  
 936 convergence as EAGLE.

## 937 E Further Details on Evaluation of EAGLE

### 938 E.1 Details on the implementation of baselines

939 We provide additional implementation details for the baseline methods used in comparison with  
 940 EAGLE. All three implementations compute an approximation of  $X^* = BA^\dagger$  and return  $D \approx X^*C$ .

941 **Exact closed-form baseline** (`np.linalg.lstsq`) This native numpy impementation of the a least-  
 942 squares solver is base on a highly optimized LAPACK routine. It determines solution to the over/under-  
 943 determined linear system  $\min_X \|XA - B\|_F^2$ . The closed-form minimiser is  $X^* = BA(AA^\top)^\dagger$ . It  
 944 returns  $D = X^*C$ .

945 **Gradient Descent.** We minimize the reconstruction loss

$$f(X) = \frac{1}{2} \sum_{i=1}^m \|XA - B\|_F^2.$$

946 *Parameters.* The learning rate is set to  $\eta_\ell = 1/(\sigma_{\max}^2(A))$ , ensuring monotonic decrease of  $f$ .  
 947 Spectral norms are estimated via two power iterations, which incur negligible memory and runtime  
 948 overhead. To avoid instabilities caused by near-singular matrices, we add a fixed ridge penalty  
 949  $\lambda = 10^{-3}$  when  $A$  is low-rank, modifying the gradient as  $G \leftarrow G + \lambda X$ . When  $A$  is full rank, we  
 950 use  $\lambda = 0$ .

951 *Iteration and Variants.* The method extends naturally to distributed or stochastic data access variants.  
 952 Each worker computes either exact or column-sketched gradients ( $\tilde{A}_\ell^i$  is the data on worker  $i$  with  
 953 subsampled columns) which are aggregated on a central node, followed by a synchronous gradient  
 954 step ( $\eta_k = 1/\max_i(\sigma_{\max}^2(\tilde{A}_k^i))$ ). This yields the following iterative updates:

$$\begin{aligned} G_k &= \sum_{i=1}^m (X_k \tilde{A}_k^i - \tilde{B}_k^i) \tilde{A}_k^{i,\top} + \lambda X_k && \text{(gradient + ridge)} \\ X_{k+1} &= X_k - \frac{\eta_k}{m} G_k && \text{(synchronous update)} \end{aligned}$$

955 Training stops when the iterate  $X_\ell$  reaches a predefined tolerance or after a fixed maximum number  
 956 of iterations.

957 **Conjugate Gradient Method.** We implement the Conjugate Gradient (CG) method to solve the  
 958 normal equations arising in Nyström estimation. The objective is to find  $X$  minimizing the empirical  
 959 loss  $\frac{1}{2} \sum_{i=1}^m \|XA - B\|_F^2$ . We define the Gram matrix  $G = AA^\top$  and the right-hand side  $B = BA^\top$ ,  
 960 and solve the system  $XG = B$  using CG.



961 *Iterations.* The standard CG updates are as follows:

$$\begin{aligned} R_k &= B - X_k G, & \text{if } k = 0 : P_0 = R_0, rr_0 = \langle R_0, R_0 \rangle \\ \alpha_k &= rr_k / \langle P_k, P_k G \rangle, & X_{k+1} = X_k + \alpha_k P_k \\ R_{k+1} &= R_k - \alpha_k P_k G, & rr_{k+1} = \langle R_{k+1}, R_{k+1} \rangle \\ \beta_k &= rr_{k+1} / rr_k, & P_{k+1} = R_{k+1} + \beta_k P_k \end{aligned}$$

962 The algorithm halts when  $\|X_k C - X_{k-1} C\|_F$  falls below a pre-specified tolerance or after a fixed  
963 number of iterations.

964 *Limitations.* The CG method is not directly suited for distributed or stochastic settings. In distributed  
965 environments, computing  $G$  and  $B$  requires sharing all local data across machines, incurring com-  
966 munication costs that exceed the desired  $O(d)$  complexity. In stochastic settings, CG requires a  
967 consistent system matrix at every iteration to guarantee convergence, which is not available when data  
968 is sampled independently at each step. Empirically, CG under stochastic updates exhibits unstable  
969 and non-convergent behavior.

## 970 E.2 Accelerated EAGLE for distributed setting

971 In the distributed setting, convergence is impeded by low data diversity, characterized by  $\alpha \ll 1$ .  
972 Recall that the continuous conditioning performed by EAGLE operates exclusively on per-machine  
973 data. As a result, the condition number of the global matrix  $A$  is lower bounded by  $\alpha^{-1}$ , even if the  
974 data on each individual machine is perfectly conditioned.

975 Once this bottleneck is reached, further updates to  $A^\mu$ —the most computationally intensive com-  
976 ponent of EAGLE—cease to be effective. This motivates the question of whether runtime can be  
977 reduced by terminating updates to  $A^\mu$  and  $B^\mu$  once per-machine conditioning has converged.

978 To this end, we implement an accelerated version of EAGLE for the distributed setting. At each  
979 iteration, we evaluate the criterion  $\min_\mu \|I - A_\ell^{\mu, \top} A_\ell^\mu\|_F^2 < 10^{-10}$ , and halt updates to  $A^\mu$  and  $B^\mu$   
980 for all  $\mu$  once this threshold is satisfied. This condition ensures that per-machine data is sufficiently  
981 well-conditioned.

982 Figure 16 reports the runtime performance of this accelerated variant. As expected, the modified  
983 version achieves faster execution compared to standard EAGLE. We note that the number of iterations  
984 required remains unchanged; the acceleration only affects wall-clock efficiency. The theoretical  
985 guaranteed of the accelerated EAGLE remain unchanged as we ensure that  $\kappa(A_\ell^\mu) = 1$  (to machine  
986 precision) before freezing the updates to  $A^\mu$  and  $B^\mu$ .

## 987 E.3 Ablations for EAGLE

988 **Data distribution.** We perform ablations on various real-world data settings. As in the main body,  
989 we place ourselves in the setting  $n = d = 240$ ,  $n' = d' = 2$  and  $r = 240$ .

990 *SVD construction.* Sample  $A_{ij}, B_{ij}$  orthogonal and set  $Z = A \Sigma B^\top / \sqrt{r}$  with  $\Sigma$  diagonal with  
991 entries log-uniform in  $[1e - 2, 1]$ . This is the reference distribution in the simulation part.

992 *Gaussian.* Sample  $A_{ij}, B_{ij} \sim \mathcal{N}(0, 1)$  and set  $Z = AB^\top / \sqrt{r}$ . Gaussian data is a widely made  
993 modeling assumption, and this is the distribution the transformer was trained on.

994 *Student- $t_\nu$  factors.* Sample  $A_{ij}, B_{ij} \sim t_\nu / \sqrt{\nu / (\nu - 2)}$  for  $\nu = 4$  and set  $Z = AB^\top / \sqrt{r}$ . Heavy-  
995 tailed entries create sporadic outliers, checking that the solver is stable beyond sub-Gaussian data.

996 *Correlated Gaussian factors.* Draw each column of  $A$  and  $B$  from  $\mathcal{N}(0, \Sigma)$  with  $\Sigma_{ij} = \rho^{|i-j|}$   
997 ( $\rho = 0.8$ ). Strong row/column correlations are typical in spatio-temporal panels and challenge  
998 methods assuming independent observations.

999 *Sparse Rademacher factors.* With sparsity  $p = 0.1$ , take  $A_{ij} = s_{ij} \xi_{ij} / \sqrt{p}$  where  $\xi_{ij} \sim \text{Bernoulli}(p)$ ,  
1000  $s_{ij} \in \{\pm 1\}$  (analogous for  $B$ ), then  $Z = AB^\top / \sqrt{r}$ . Entry-wise sparsity yields highly *coherent*  
1001 singular vectors—a worst-case regime for many theoretical guarantees.

1002 *Block / clustered factors.* Assign each of the  $d$  rows to one of  $k$  clusters ( $k = 5$ ),  $A \in \{0, 1\}^{d \times k}$  is  
1003 one-hot, and  $B \in \mathbb{R}^{n \times k}$  holds cluster centroids; set  $Z = AB^\top / \sqrt{k}$ . Produces piece-wise-constant  
1004 structure seen in recommender systems, violating global incoherence yet preserving low rank.

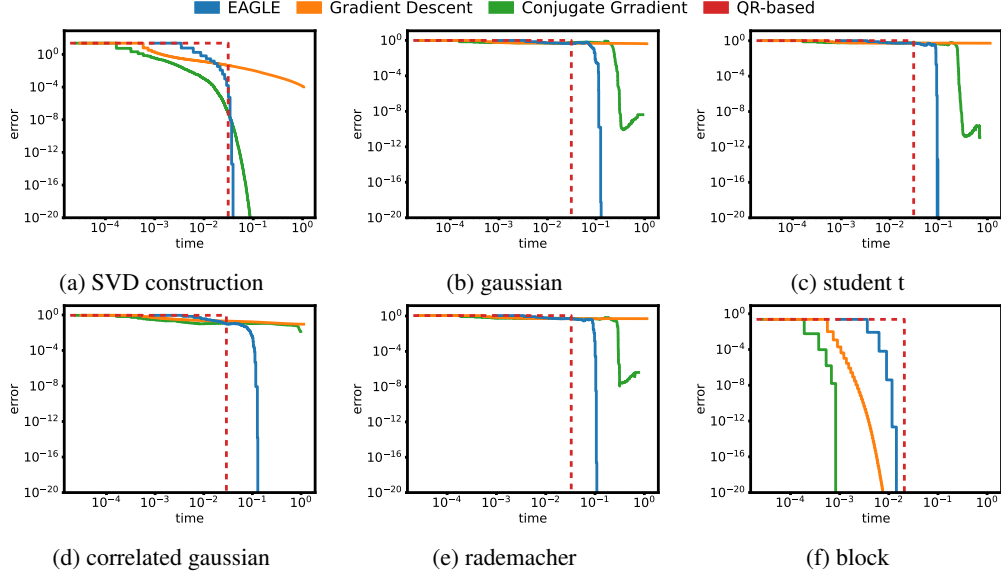


Figure 18: Performance with different data distribution,  $n = d = 240$ ,  $n' = d' = 2$ , rank 240.

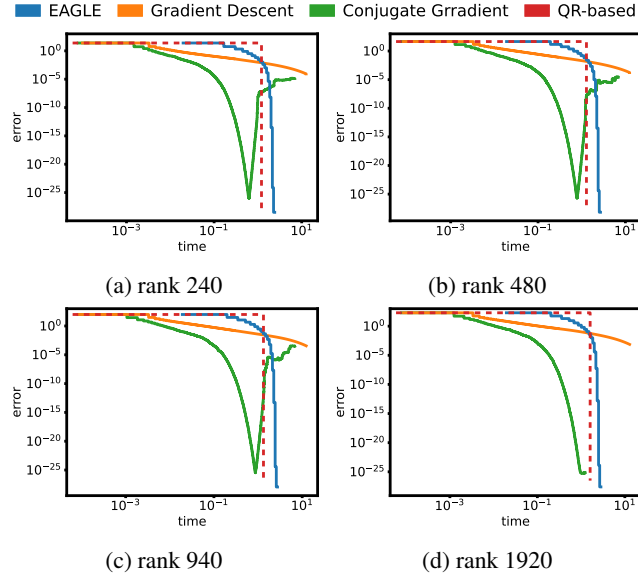


Figure 19: Performance on larger problems,  $n = d = 1920$ ,  $n' = d' = 2$ .

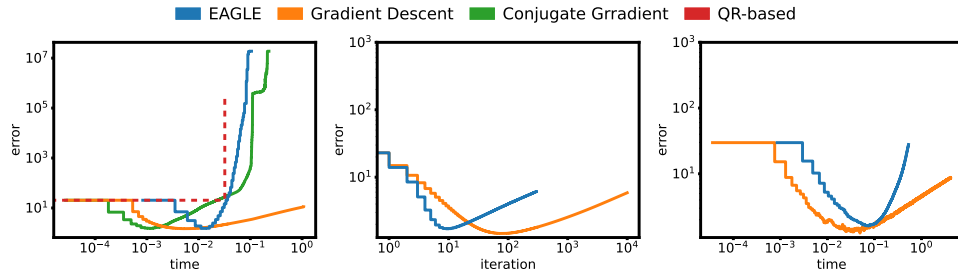


Figure 20: Performance with i.i.d. Gaussian noise (standard deviation  $\sigma = 0.01$ ) in the unconstrained (left), distributed (center) and stochastic data access (right) settings.

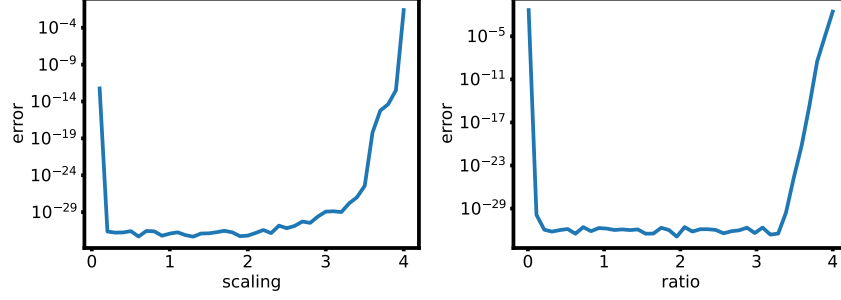


Figure 21: Best performance of EAGLE in 200 iterations as the step sizes are varied. Left:  $\gamma$  and  $\eta$  are both scaled by a common factor. Right:  $\gamma$  is scaled by  $\sqrt{\text{ratio}}$  while  $\eta$  is scaled by  $1/\sqrt{\text{ratio}}$  such that  $\gamma/\eta$  is scaled by ratio.

**Larger systems.** We increase the size of the data matrix  $X$  to  $n = d = 1920$  and  $n' = d' = 2$ ; results are shown in Figure 19. These experiments further reveal that the Conjugate Gradient method does not converge reliably when applied to low-rank matrices. As a result, its performance is omitted from the main results in the low-rank setting.

**Data noise.** To assess robustness to noise, we add moderate i.i.d. Gaussian noise to the matrix  $X$  and evaluate EAGLE in all three computational settings; results are presented in Figure 20. We observe that EAGLE exhibits the same form of implicit regularization during early iterations that is well-documented for gradient descent. Across all settings, EAGLE with early stopping achieves performance comparable to gradient descent, recovering regularized solutions that are more stable than the Nyström solution. Additionally, we note that the addition of Gaussian noise to  $X$  inadvertently improves the conditioning of the matrix  $A$ , effectively reducing  $\kappa(A)$ .

**Sensitivity to step sizes.** To evaluate the sensitivity of EAGLE to the choice of step sizes, we conduct a series of controlled experiments in the unconstrained setting, using a small-scale configuration with  $n = d = 15$ ,  $n' = d' = 2$ , and rank 15.

First, we scale both step sizes  $\gamma$  and  $\eta$  by a common factor, as shown in Figure 21 (left). This setup models scenarios in which the estimate of the scaling parameter  $\rho_\ell = 1/\|A_\ell\|_2^2$  is inaccurate. The results indicate that scaling  $\gamma$  and  $\eta$  jointly by a factor in the range  $[0.2, 2]$  has minimal effect on performance.

Second, we vary the ratio  $\gamma/\eta$  to test the algorithm’s robustness to discrepancies between these two parameters, as illustrated in Figure 21 (right). We find that EAGLE remains stable over a wide range of ratios; specifically, varying  $\gamma/\eta$  within  $[0.2, 2]$  has little impact on overall performance.

## F Theoretical Analyses

### F.1 Analysis of the Unconstrained or Centralized Iteration

We begin by analysing the behaviour of Algorithm 1 in the ‘centralized’ setting of  $S = I_n$ ,  $M = 1$ . For simplicity, let us set  $\eta_\ell = \eta\|A_\ell\|_2^{-2}$ , and  $\gamma_\ell = \gamma\|A_\ell\|_2^{-2}$ —we will incorporate the specific structure of  $\eta, \gamma$  later. Then we can write the iterations as

$$\begin{aligned} A_{\ell+1} &= A_\ell(I - \eta_\ell A_\ell^\top A_\ell), \\ B_{\ell+1} &= B_\ell(I - \eta_\ell A_\ell^\top A_\ell), \\ C_{\ell+1} &= (I - \gamma_\ell A_\ell A_\ell^\top)C_\ell, \\ D_{\ell+1} &= D_\ell + \gamma_\ell B_\ell A_\ell^\top C_\ell. \end{aligned} \tag{5}$$

Of course, we set  $A_0, B_0, C_0$  to be the observed blocks of the input matrix  $X$ , and  $D_0 = 0$ .

**Useful Definitions.** We will analyze these iterations through the follow two objects

1033 **Definition 2.** For  $\ell \geq 0$ , define the ‘signal energy’  $\mathcal{E}_\ell$ , and the ‘signal correlation’  $\mathcal{V}_\ell$  as

$$\mathcal{E}_\ell := A_\ell A_\ell^\top \in \mathbb{R}^{d \times d} \text{ and } \mathcal{V}_\ell := B_\ell A_\ell^\top \in \mathbb{R}^{d' \times d}.$$

1034 We further note the critical relationship that

$$B_0 = W_* A_0 \implies \mathcal{V}_0 = W_* \mathcal{E}_0),$$

1035 where  $W_*$  is the Nyström parameter for the data matrix  $X$  (see §B.1. An immediate consequence of  
1036 the iterations above is the following dynamics.

1037 **Lemma 4.** *Under the iterations of (5), we have the following dynamics*

$$\begin{aligned} \mathcal{E}_{\ell+1} &= \mathcal{E}_\ell (I - \eta_\ell \mathcal{E}_\ell)^2 \\ \mathcal{V}_{\ell+1} &= \mathcal{V}_\ell (I - \eta_\ell \mathcal{E}_\ell)^2 \\ C_{\ell+1} &= (I - \gamma_\ell \mathcal{E}_\ell) C_\ell \\ D_{\ell+1} &= D_\ell + \gamma_\ell \mathcal{V}_\ell C_\ell, \end{aligned} \tag{6}$$

1038 where  $I$  is the  $d \times d$  identity matrix.

1039 *Proof.* The  $D_\ell, C_\ell$  iterations follow immediately by definition. For the former iterations, first observe  
1040 that

$$\begin{aligned} \mathcal{V}_{\ell+1} &= B_{\ell+1} A_{\ell+1}^\top = B_\ell (I - \eta_\ell A_\ell^\top A_\ell) (I - \eta_\ell A_\ell^\top A_\ell) A_\ell^\top \\ &= B_\ell (I - \eta_\ell A_\ell A_\ell^\top) (A_\ell^\top - \eta_\ell A_\ell^\top A_\ell A_\ell^\top) \\ &= B_\ell (I - \eta_\ell A_\ell A_\ell^\top) A_\ell^\top (I - \eta_\ell A_\ell A_\ell^\top) \\ &= B_\ell A_\ell^\top (I - \eta_\ell A_\ell A_\ell^\top) (I - \eta_\ell A_\ell A_\ell^\top) \\ &= \mathcal{V}_\ell (I - \eta_\ell \mathcal{E}_\ell)^2, \end{aligned}$$

1041 where the first few inequalities arise by multiplying  $A_\ell^\top$  from the right, and then factoring it out by  
1042 the left, and the final is by definition. Similarly,

$$\begin{aligned} \mathcal{E}_{\ell+1} &= A_{\ell+1} A_{\ell+1}^\top = A_\ell (I - \eta_\ell A_\ell^\top A_\ell) (I - \eta_\ell A_\ell^\top A_\ell) A_\ell^\top \\ &= A_\ell A_\ell^\top (I - \eta_\ell A_\ell A_\ell^\top)^2 \\ &= \mathcal{E}_\ell (I - \eta_\ell \mathcal{E}_\ell)^2, \end{aligned}$$

1043 where we reuse this transfer of  $A_\ell^\top$  from the right to the left, and then use the definition of  $\mathcal{E}_\ell$ .  $\square$

1044 Note from the above expression that for any  $\ell' > \ell$ ,  $\mathcal{E}_{\ell'}$  is a polynomial in  $\mathcal{E}_\ell$ . As a result,  $\mathcal{E}_\ell$  and  $\mathcal{E}_{\ell'}$   
1045 commute for all pairs  $(\ell, \ell')$ .

1046 **Telescoping the Error.** For the sake of conciseness in the further expressions, we further introduce  
1047 the following notation.

1048 **Definition 3.** Define  $M_\ell := \prod_{l < \ell} (I - \eta_l \mathcal{E}_l)^2$  and  $N_\ell^\top := \prod_{l < \ell} (I - \gamma_l \mathcal{E}_l)$ , where we use the  
1049 convention that

$$\prod_{l < \ell} U_l = U_1 U_2 \cdots U_{\ell-1},$$

1050 and  $M_0 = N_0 = I$ .

1051 By Lemma 4, we have  $\mathcal{E}_\ell = \mathcal{E}_0 M_\ell$  and  $\mathcal{V}_\ell = \mathcal{V}_0 M_\ell$ . Further,  $C_\ell = N_\ell C_0$ .<sup>1</sup>

1052 This leads to the following estimate of the value of  $D_\ell$ .

1053 **Lemma 5.** *Let  $W_*$  be the Nyström regression parameter for the data  $(A, B)$ . For any  $\ell \geq 0$ , we have*

$$D_\ell = W_* (I - N_\ell) C_0.$$

<sup>1</sup>Note that the order of multiplication is flipped here, which follows notationally since we defined the transpose  $N_\ell^\top$  above. Of course, all the  $\mathcal{E}_l$  commute, so this is not very important in this case, but this subtle distinction will matter more in subsequent analysis.

1054 *Proof.* Since  $D_0 = 0$ , we have

$$\begin{aligned} D_\ell &= \sum_{l < \ell} \gamma_l \mathcal{V}_l C_l = \mathcal{V}_0 \left( \sum_{l < \ell} \gamma_l M_l N_l \right) C_0 \\ &= W_* \mathcal{E}_0 \left( \sum_{l < \ell} \gamma_l M_l N_l \right) C_0 = W_* \left( \sum_{l < \ell} \gamma_l \mathcal{E}_0 M_l N_l \right) C_0 \\ &= W_* \left( \sum_{l < \ell} \gamma_l \mathcal{E}_l N_l \right) C_0, \end{aligned}$$

1055 where we used the definition of  $M_\ell$ . But now, using the definition of  $N_\ell$ , notice that for any  $l$ ,

$$N_{l+1}^\top = N_l^\top (I - \gamma_l \mathcal{E}_l) \iff N_{l+1} = N_l - \gamma_l \mathcal{E}_l N_l \iff \gamma_l \mathcal{E}_l N_l = N_l - N_{l+1}.$$

1056 Consequently, we can write

$$D_\ell = W_* \left( \sum_{l < \ell} (N_l - N_{l+1}) \right) C_0 = W_* (I - N_\ell) C_0. \quad \square$$

1057 Recall that  $\hat{D}_* = W_* C_0$ , meaning that the error  $D_\ell - \hat{D}_*$  is  $W_* N_\ell C_0$ . Thus, this lemma captures a  
 1058 basic fact: as the size of the matrix  $N_\ell$  decays, the output  $D_\ell$  gets closer to the target output  $\hat{D}_*$ . Thus,  
 1059 our main focus is to characterize the decay of this matrix. We shall show this by arguing that the  
 1060 matrices  $\mathcal{E}_\ell$  quickly become well-conditioned through the course of the iterations, and consequently  
 1061  $N_\ell$  decays quickly.

1062 Before proceeding, note that in general,  $\mathcal{E}_0$  may have zero eigenvalues, which are left unchanged by  
 1063 the main dynamics (see below), meaning that  $I - N_\ell$  may even asymptotically have a large eigenvalue  
 1064 corresponding to vectors in the kernel of  $\mathcal{E}_0$ . However, we observe that  $W_* = \mathcal{V}_0 \mathcal{E}_0^\dagger$ , and thus any  
 1065 energy in  $C_0$  that lies in the kernel of  $\mathcal{E}_0$  is irrelevant to the prediction  $\hat{D}_*$ . As such, it is equivalent  
 1066 for us to analyse the two-norm of the matrix  $\tilde{N}_\ell := (I - P_0) N_\ell$ , where  $P_0$  projects onto the kernel  
 1067 of  $\mathcal{E}_0$ . Instead of this notational complication, we will henceforth just assume that  $\mathcal{E}_0$  is full rank (i.e.,  
 1068 all of its eigenvalues are nonzero), and mention where changes need to be made to handle the general  
 1069 case.

1070 **Conditioning of  $\mathcal{E}_\ell$ .** Let us then begin with some notation: for a positive (semi-)definite symmetric  
 1071 matrix  $\mathcal{M}$ , we let  $\lambda^i(\mathcal{M})$  denote its  $i$ th largest eigenvalue,  $\bar{\lambda}(\mathcal{M})$  denote its largest eigenvalue, and  
 1072  $\underline{\lambda}(\mathcal{M})$  denote its smallest (nonzero) eigenvalue. Notice that  $\bar{\lambda}(\cdot) = \lambda^1(\cdot)$ , and  $\underline{\lambda}(\mathcal{M}) = \lambda^r(\mathcal{M})$ ,  
 1073 where  $r$  is the rank of  $\mathcal{M}$  (we will work with full rank  $\mathcal{M}$ , but be cognizant of rank sensitivity). Let  
 1074  $v^i(\mathcal{M})$  denote the corresponding eigenvector. We further introduce the notation

$$\lambda_\ell^i = \lambda^i(\mathcal{E}_\ell), \quad \bar{\lambda}_\ell = \bar{\lambda}(\mathcal{E}_\ell), \quad \text{and} \quad \underline{\lambda}_\ell = \underline{\lambda}(\mathcal{E}_\ell).$$

1075 Our first observation is that the iterations leave the eigenstructure of  $\mathcal{E}_\ell$  undisturbed.

1076 **Lemma 6.** *Suppose that  $v$  is an eigenvector of  $\mathcal{E}_0$ . Then it is also an eigenvector of  $\mathcal{E}_\ell$  for any  $\ell > 0$ .  
 1077 Further, if  $v \in \ker(\mathcal{E}_0)$  then  $v \in \ker(\mathcal{E}_\ell)$ .*

1078 *Proof.* We have  $\mathcal{E}_0^n v = \lambda^n v$ . But note that  $\mathcal{E}_\ell$  is some polynomial in  $\mathcal{E}_0$ , i.e., for some finite number of  
 1079 coefficients  $\alpha_n$ ,  $\mathcal{E}_\ell v = \sum_{n \geq 0} \alpha_n \mathcal{E}_0^n v = (\sum_{n \geq 0} \alpha_n \lambda^n) v$ , and the claim follows. Crucially, observe  
 1080 that since  $\mathcal{E}_\ell = M_\ell \mathcal{E}_0$ , we have  $\alpha_0 = 0$ , and so if  $\mathcal{E}_0 v = 0$ , then  $\mathcal{E}_\ell v = 0$  as well.  $\square$

1081 Since the eigenstructure of  $\mathcal{E}_0$  is preserved, we can begin to study the behaviour of its eigenvalues in  
 1082 a simplified way. The critical relationship for our analysis is the following iterative structure on the  
 1083 largest and smallest (nonzero) eigenvalues of  $\mathcal{E}_\ell$ .

1084 **Lemma 7.** *Suppose that  $\forall \ell, \eta_\ell \leq \bar{\lambda}_\ell^{-1}/3$ . Then for all  $\ell$ ,*

$$\bar{\lambda}_{\ell+1} = (1 - \eta_\ell \bar{\lambda}_\ell)^2 \bar{\lambda}_\ell \quad \text{and} \quad \underline{\lambda}_{\ell+1} = (1 - \eta_\ell \underline{\lambda}_\ell)^2 \underline{\lambda}_\ell.$$

1085 *Proof.* Suppose  $u^1, u^2$  are two eigenvectors of  $\mathcal{E}_0$ , with positive eigenvalues  $\mu_0, \nu_0$ . Since  $u^1, u^2$   
 1086 are also eigenvectors of  $\mathcal{E}_\ell$ , denote the eigenvalues for the latter as  $\mu_\ell, \nu_\ell$ . Note that these remain  
 1087 nonnegative. Indeed, by multiplying the dynamics for  $\mathcal{E}_{\ell+1}$  by, say,  $u^1$ , we have

$$\mu_{\ell+1}u^1 = \mathcal{E}_{\ell+1}u^1 = \mathcal{E}_\ell(I - \eta_\ell \mathcal{E}_\ell)^2 u^1 = (1 - \eta_\ell \mu_\ell)^2 \mu_\ell,$$

1088 and similarly for  $\nu_\ell$ . We will first inductively show that if  $\mu_0 > \nu_0 \iff \mu_\ell > \nu_\ell$  for all  $\ell$ . In other  
 1089 words, not only are the eigenvectors of  $\mathcal{E}_0$  stable under the iterations, but also the ordering of the  
 1090 eigenvalues.

1091 We thus note that the claim follows directly from this result. Indeed, let  $v^1$  be the eigenvector  
 1092 corresponding to  $\bar{\lambda}_0$ . Then we see that it remains eigenvector corresponding to  $\bar{\lambda}_\ell$  for all  $\ell$ . But then  
 1093 using our observation above with  $u^1 = v^1$ , we get the result for  $\bar{\lambda}_{\ell+1}$ . A similar argument works  
 1094 for  $\underline{\lambda}_{\ell+1}$ . Note that nothing per se demands that these eigenvalues have unit multiplicity, and the  
 1095 argument is completely insensitive to this.

1096 To see the ordering claim on  $\mu_\ell, \nu_\ell$ , then, we observe that

$$\begin{aligned} \mu_{\ell+1} - \nu_{\ell+1} &= (1 - \eta_\ell \mu_\ell)^2 \mu_\ell - (1 - \eta_\ell \nu_\ell)^2 \nu_\ell \\ &= (\mu_\ell - \nu_\ell) - 2\eta_\ell(\mu_\ell^2 - \nu_\ell^2) + \eta_\ell^2(\mu_\ell^3 - \nu_\ell^3) \\ &= (\mu_\ell - \nu_\ell) (1 - 2\eta_\ell(\mu_\ell + \nu_\ell) + \eta_\ell^2(\mu_\ell^2 + \nu_\ell^2 + \mu_\ell \nu_\ell)). \end{aligned}$$

1097 Now, if the term multiplying  $(\mu_\ell - \nu_\ell)$  is nonnegative, then the claim follows. To see when this  
 1098 occurs, let us set  $\eta_\ell = \eta \bar{\lambda}_\ell^{-1}$ , and pull  $\bar{\lambda}^{-1}$  within the brackets. We are left with an expression of the  
 1099 form

$$(1 - 2\eta(x + y) + \eta^2(x^2 + y^2 + xy)),$$

1100 where  $(x, y) \in (0, 1)$ . For  $\eta \leq 2/3$ , the minimum over this range occurs when  $x = y = 1$ , and takes  
 1101 the value

$$1 - 4\eta + 3\eta^2.$$

1102 It is a triviality to show that this function is nonnegative for  $\eta \leq 1/3$ , and so we are done.  $\square$

1103 With this in hand, we will argue that  $\lambda_\ell \rightarrow \bar{\lambda}_\ell$  at a quadratic convergence rate after an initial burn-in  
 1104 period. Of course, this is equivalent to saying that  $\kappa_\ell := \bar{\lambda}_\ell / \lambda_\ell \rightarrow 1$ . Notice that this  $\kappa_\ell$  is precisely  
 1105 the condition number of  $\mathcal{E}_\ell$  (which in turn is the square of the condition number of  $A_\ell$  mentioned in  
 1106 our discussion in §5).

1107 **Lemma 8.** Define  $\kappa_\ell = \bar{\lambda}_\ell / \lambda_\ell$ , and  $\theta_\ell = \kappa_\ell - 1$ . If  $\eta_\ell \leq \bar{\lambda}^{-1}/3$ , then  $\theta_{\ell+1} \leq \theta_\ell \exp(-5\eta_\ell \bar{\lambda}_\ell/3)$ .  
 1108 Further, if  $\eta_\ell \bar{\lambda}_\ell = 1/3$ , then  $\theta_{\ell+1} \leq 3\theta_\ell^2/4$ .

1109 *Proof.* For the sake of succinctness, let us write  $\eta_\ell = \eta, \bar{\lambda}_\ell = \bar{\lambda}, \lambda_\ell = \underline{\lambda}, \theta_\ell = \theta$ , and  $\theta_{\ell+1} = \theta_+$ .  
 1110 Then, using the previous lemma, we can write

$$\kappa_{\ell+1} = \theta_+ + 1 = (\theta + 1) \left( \frac{1 - \eta \bar{\lambda}}{1 - \eta \underline{\lambda}} \right)^2.$$

1111 But notice that

$$\frac{1 - \eta \bar{\lambda}}{1 - \eta \underline{\lambda}} = 1 + \eta \frac{\bar{\lambda} - \underline{\lambda}}{1 - \eta \underline{\lambda}} = 1 - \eta \frac{\bar{\lambda}(1 - 1/(\theta + 1))}{1 - \eta \bar{\lambda}/(\theta + 1)} = 1 - \frac{\theta \eta \bar{\lambda}}{\theta + 1 - \eta \bar{\lambda}}.$$

1112 Thus,

$$\begin{aligned} \theta_+ + 1 &= (\theta + 1) \left( 1 - 2 \frac{\theta \eta \bar{\lambda}}{(\theta + 1 - \eta \bar{\lambda})} + \frac{\theta^2 \eta^2 \bar{\lambda}^2}{(\theta + 1 - \eta \bar{\lambda})^2} \right) \\ &= \theta + 1 - \theta \frac{\eta \bar{\lambda}}{1 - \eta \underline{\lambda}} \left( 2 - \frac{\theta \eta \bar{\lambda}}{(\theta + 1 - \eta \bar{\lambda})} \right). \end{aligned}$$

1113 But the bracketed term is nonincreasing in  $\eta \bar{\lambda}$  (in the negative term, the numerator increases and the  
 1114 denominator decreases with this value), and so

$$2 - \frac{\theta \eta \bar{\lambda}}{\theta + 1 - \eta \bar{\lambda}} \geq 2 - \frac{\theta}{3\theta + 3 - 1} = \frac{5\theta + 4}{3\theta + 2} \geq \frac{5}{3}.$$

Thus, we have

$$\theta_{\ell+1} \leq \theta_\ell - \theta_\ell \cdot \frac{5/3\eta_\ell \bar{\lambda}_\ell}{1 - \eta_\ell \underline{\lambda}_\ell} \implies \theta_{\ell+1} \leq \theta_\ell \exp\left(-\frac{5\eta_\ell \bar{\lambda}_\ell/3}{1 - \eta_\ell \underline{\lambda}_\ell}\right) \leq \theta_\ell \exp(-\eta_\ell \bar{\lambda}_\ell).$$

On the other hand (going back to the notation that suppresses  $\ell$ ), if we set  $\eta\bar{\lambda} = L$ , then since

$$\frac{1 - \eta\bar{\lambda}}{1 - \eta\underline{\lambda}} = \frac{1 - L}{(\theta + 1 - L)/(\theta + 1)},$$

we find by doing the long multiplication that

$$\begin{aligned} (\theta_+ + 1) &= \frac{(\theta + 1)^3(1 - L)^2}{(\theta + 1 - L)^2} \\ \iff \theta_+ &= \frac{\theta(1 - 4L + 3L^2) + \theta^2(2 - 6L + 3L^2) + \theta^3(1 - L)^2}{(\theta + 1 - L)^2}. \end{aligned}$$

Setting  $L = \eta_\ell \bar{\lambda}_\ell = 1/3$ , the linear term in  $\theta$  vanishes, and we end up with

$$\theta_+ = \frac{\theta^2/3 + 4\theta^3/9}{(\theta + 2/3)^2} = \theta^2 \cdot \frac{4\theta + 3}{(3\theta + 2)^2} \leq \frac{3\theta^2}{4}.$$

□

The statement of this Lemma has shown two decay modes of  $\theta_\ell = (\kappa_\ell - 1)$ . Firstly, as long as  $\eta_\ell \leq \bar{\lambda}_\ell^{-1}/3$ , this quantity decays at least at a linear rate. Further, if  $\eta_\ell$  is set to exactly  $\bar{\lambda}_\ell^{-1}/3$ , then once  $\theta_\ell$  dips below 1, which occurs after about  $3 \log(\theta_0)/5$  iterations, we can exploit the quadratic bound in  $\theta$  to recover a stronger convergence rate. Together, this yields the following convergence behaviour for  $\kappa_\ell$ .

**Lemma 9.** *Let  $\kappa_\ell := \bar{\lambda}_\ell/\underline{\lambda}_\ell$ . If for all  $\ell$ ,  $\eta_\ell \bar{\lambda} = 1/3$ , then  $\kappa_\ell - 1 \leq \varepsilon$  for all  $\ell \geq L = 2 + \lceil (3/5) \log(\kappa_0) \rceil + \lceil \log_2(\log(4/3\varepsilon)) \rceil$ .*

*Proof.* First using the geometric decay in Lemma 8, we know that  $\theta_\ell \leq (\kappa_0 - 1) \exp(-5(\ell - 1)/3)$ . Set  $\ell_0 = 2 + 3/5 \log(\kappa_0 - 1)$ . Then for  $\ell \geq \ell_0$ , we have  $\theta_{\ell_0} \leq 1/e$ . Further, for iterations beyond this  $\ell_0$ , the supergeometric decay  $\theta_{\ell+1} \leq 3\theta_\ell^2/4$  in Lemma 8 implies that

$$3\theta_{\ell+1}/4 \leq (3\theta_\ell/4)^2 \iff T_{\ell+1} \leq 2T_\ell,$$

where  $T_\ell := \log(3\theta_\ell/4)$ . Thus,

$$T_\ell \leq 2^{\ell-\ell_0} T_{\ell_0} \leq -2^{\ell-\ell_0} \implies \theta_\ell \leq 4/3 \exp(-2^{\ell-\ell_0}).$$

Setting this  $< \varepsilon$  gives the claim. □

We again note that this  $\kappa_\ell$  is the square of the condition number of  $A_\ell$  - however, this distinction is easily accommodated due to the logarithmic dependence on it - the only change is that the  $3/5$  above increases to  $6/5$ .

**Back to  $N_\ell$ .** The conditioning of  $\mathcal{E}_\ell$  yields a direct control on the behaviour of  $N_\ell$ , as encapsulated below. We note that if  $\mathcal{E}_0$  is not full rank, the statement holds for  $\tilde{N}_\ell = (I - P_0)N_\ell$ , where  $P_0$  projects onto the kernel of  $\mathcal{E}_0$ .

**Lemma 10.** *For all  $\ell$ ,  $\|N_\ell\|_2 \leq \exp(-\sum_{l < \ell} \gamma_l \underline{\lambda}_l)$ . Further, if  $\forall \ell, \eta_\ell = \bar{\lambda}^{-1}/3$  and  $\gamma_\ell = \bar{\lambda}^{-1}$ , then  $\|N_\ell\|_2 \leq \varepsilon$  for all  $\ell \geq L + 1$ , where*

$$L = 2 + \lceil 3/5 \log(\kappa_0) \rceil + \lceil \log_2(\log 4/3\varepsilon) \rceil.$$

*Proof.* Since  $N_{l+1} = (I - \gamma_l \mathcal{E}_l)N_l$ , we immediately have

$$\|N_\ell\|_2 \leq \prod_{l < \ell} \|I - \gamma_l \mathcal{E}_l\|_2.$$

But note that  $\|I - \gamma_l \mathcal{E}_l\|_2 = (1 - \gamma_l \underline{\lambda}_l)$ . Thus, we immediately have  $\|N_\ell\| \leq \prod (1 - \gamma_l \underline{\lambda}_l) \leq \exp(\sum_{l < \ell} \gamma_l \underline{\lambda}_l)$ . Further, recall from Lemma 9 that if  $\eta_\ell = \bar{\lambda}^{-1}/3$  for all  $\ell$ , then  $\underline{\lambda}_\ell \geq \bar{\lambda}_\ell/(1 + \varepsilon)$  for all  $\ell \geq L$ . But then, if  $\eta_\ell = \bar{\lambda}^{-1}$ , we have

$$1 - \eta_\ell \underline{\lambda} \leq 1 - \underline{\lambda}/\bar{\lambda} \leq \varepsilon/(1 + \varepsilon) \leq \varepsilon,$$

yielding the claimed bound on  $\|N_\ell\|_2$ . □

1143 **Finishing the Error Control.** With all the pieces in place, we conclude the main argument.

1144 *Proof of Theorem 1.* By Lemma 5, we have  $D_\ell - \hat{D}_* = W_* N_\ell C_0$ , and consequently,

$$\|D_\ell - \hat{D}_*\|_F \leq \|W_* N_\ell\|_F \|C_0\|_F \leq \sqrt{\text{rank}(W_* N_\ell)} \|W_* N_\ell\|_2 \|C_0\|_F.$$

1145 But, since  $W_* N_\ell \in \mathbb{R}^{d' \times d}$ , the rank about is at most  $d'$ . Finally,  $\|W_* N_\ell\|_2 \leq \|N_\ell\|_2 \|W_*\|_2$ .  
 1146 Thus, the claim follows as soon as  $\|N_\ell\|_2 \leq \varepsilon/(\sqrt{d'} \|W_*\|_2 \|C_0\|_F)$ , for which we may invoke  
 1147 Lemma 10.  $\square$

1148 Again, recall that for the purposes of error, if  $\mathcal{E}_0$  were not full rank, then we could instead replace  $N_\ell$   
 1149 by  $\tilde{N}_\ell$  in all statements above, and thus the claim also extends to this situation.

1150 **Comment on Rates  $\eta, \gamma$ .** Going back to the notation of Algorithm 1, we reparametrise  $\gamma_\ell = \rho_\ell =$   
 1151  $\bar{\lambda}_\ell^{-1} = \|A_\ell\|_2^{-2}$ , and  $\eta_\ell = \rho_\ell/3$ . It is worth discussing briefly how we may go about estimating this  
 1152  $\rho_\ell$ .

1153 A simple observation in this setting is that if we assume that we know  $\bar{\lambda}_0 = \|\mathcal{E}_0\|_0 = \|A\|_2^2$  to begin  
 1154 with, then it is a simple matter to compute the subsequent values of  $\bar{\lambda}_\ell$ , since if we set  $\eta_\ell, \gamma_\ell$  as per  
 1155 the above, this is directly computed iteratively via

$$\bar{\lambda}_{\ell+1} = \bar{\lambda}_\ell(1 - 1/3)^2 = 4\bar{\lambda}_\ell/9.$$

1156 In fact, under this assumption, we may avoid further numerical stability issues by first recaling  
 1157  $A$  to have 2-norm 1, and subsequently simply rescaling the matrices up  $A_\ell, B_\ell$  by  $3/2$  after each  
 1158 update to maintain the invariant  $\bar{\lambda}_\ell = \bar{\lambda}_0$ —the behaviour of  $N_\ell$  remains the same, and convergence  
 1159 rates are only driven by the conditioning of  $A_\ell$ . So, equivalently, the question we must concern  
 1160 ourselves with is finding  $\|A\|_2$ . Of course, this is quite cheap, since we can compute this simply  
 1161 by power iteration, which involves only matrix-vector products rather than matrix-matrix products.  
 1162 Nevertheless, note that power iteration is only linearly convergent (i.e., the number of iterations  
 1163 needed to find a  $\varepsilon$ -approximation of the top eigenvalue is  $\Theta(\log(1/\varepsilon))$ ), so in full generality, this  
 1164 procedure would destroy the second order convergence.

1165 In our simulations of §5, we indeed implemented these iterations by carrying out power iteration.  
 1166 Note that practically, then, the method essentially retains its second order behaviour despite this  
 1167 approximation. One aspect of this lies in the fact that even if we underestimate the top eigenvalue  
 1168 by a small amount, and so undertune  $\eta, \gamma$  by a slight amount, the second order bound of Lemma 9  
 1169 decays gracefully, and so retains practical resilience. Of course, characterising exactly how loose this  
 1170 can be requires more precise analysis, which we leave for future work.

## 1171 F.2 Distributed Analysis

1172 Beginning again with Algorithm 1, with  $S = I_n, M > 1$ , we need to analyse the iterations

$$\begin{aligned} \forall \mu, A_{\ell+1}^\mu &= A_\ell^\mu (I - \eta_\ell^\mu A_\ell^{\mu, \top} A_\ell^\mu), \\ \forall \mu, B_{\ell+1}^\mu &= B_\ell^\mu (I - \eta_\ell^\mu A_\ell^{\mu, \top} A_\ell^\mu), \\ C_{\ell+1} &= C_\ell - \frac{1}{m} \sum_\mu \gamma_\ell A_\ell^\mu A_\ell^{\mu, \top} C_\ell, \\ D_{\ell+1} &= D_\ell + \frac{1}{m} \sum_\mu \gamma_\ell B_\ell^\mu A_\ell^{\mu, \top} C_\ell. \end{aligned} \tag{7}$$

1173 We note that here we have set  $\gamma_\ell$  to be constant across all machines, and nominally it is not pegged  
 1174 to  $\|A_\ell^\mu\|_2^{-2}$ , which could vary across machines. However, we will analyze this method under the  
 1175 normalization assumption that

$$\forall \mu, \|A_0^\mu\|_2 = 1.$$

1176 In this setting, we will show that the choices  $\eta_\ell^\mu = \|A_\ell^\mu\|_2^{-2}/3$  are also the same for each machine,  
 1177 and then set  $\gamma_\ell = 3\eta_\ell^\mu$  (which in turn is also the same for every machine). Thus, in the setting we  
 1178 analyze, the iterations above are faithful to the structure of Algorithm 1.



1179 Note, of course, that this distributed data computes the Nyström approximation of

$$\begin{bmatrix} A & C \\ B & D \end{bmatrix},$$

1180 where  $C, D$  are as in the iteration, while

$$A = [A^1 \ A^2 \ \dots \ A^m], B = [B^1 \ B^2 \ \dots \ B^m].$$

1181 We will work in the noise-free regime, wherein there exists a matrix  $W_*$  such that

$$\begin{bmatrix} B & D \end{bmatrix} = W_* \begin{bmatrix} A & C \end{bmatrix},$$

1182 and  $C \in \text{column-span}(A)$ . Of course, this  $W_*$  also equals the Nyström parameter for this matrix  
 1183 (§B.1). The second condition ensures that the rank of this matrix is the same as that of  $A$ , and is  
 1184 needed to ensure that we can actually infer  $W_*$  in the directions constituted by the columns of  $C$   
 1185 (without which one cannot recover  $D$ ).

1186 **Per-machine and Global Signal Energies.** We begin with defining the energy and correlation  
 1187 matrices in analogy to the previous section. As before,  $A_0^\mu = A^\mu, B_0^\mu = B^\mu, C_0 = C, D_0 = 0$ .

1188 **Definition 4.** We define the per-machine objects

$$\mathcal{E}_\ell^\mu := A_\ell^\mu A_{\ell}^{\mu,\top} \text{ and } \mathcal{V}_\ell^\mu := B_\ell^\mu A_{\ell}^{\mu,\top},$$

1189 and the per-machine values

$$\bar{\lambda}_\ell^\mu = \lambda^1(\mathcal{E}_\ell^\mu), \underline{\lambda}_\ell^\mu = \lambda^{r(\mu)}(\mathcal{E}_\ell^\mu),$$

1190 where  $r(\mu)$  is the rank of  $\mathcal{E}_0^\mu$ . Further, we define the global signal energy

$$\bar{\mathcal{E}}_\ell := \frac{1}{m} \sum \mathcal{E}_\ell^\mu.$$

1191 Note that by our assumption,  $\|\mathcal{E}_0^\mu\|_2 = 1$  for all  $\mu$ . Further, the behaviour of  $\mathcal{E}_\ell^\mu, \mathcal{V}_\ell^\mu$  is identical to  
 1192 that in the centralised case, i.e.,

1193 **Lemma 11.** For all  $\mu, \ell$ ,

$$\mathcal{E}_{\ell+1}^\mu = \mathcal{E}_\ell^\mu (I - \eta_\ell^\mu \mathcal{E}_\ell^\mu)^2 \text{ and } \mathcal{V}_{\ell+1}^\mu = \mathcal{V}_\ell^\mu (I - \eta_\ell^\mu \mathcal{E}_\ell^\mu)^2.$$

1194 As a consequence, for every  $\ell, \mu$ , it holds that

$$\mathcal{V}_\ell^\mu = W_* \mathcal{E}_\ell^\mu.$$

1195 Finally, for any  $\ell$ ,

$$C_{\ell+1} = (I - \gamma_\ell \bar{\mathcal{E}}_\ell) C_\ell$$

1196 *Proof.* The iterations for  $\mathcal{E}_{\ell+1}^\mu, \mathcal{V}_{\ell+1}^\mu$  follow identically to the proof of Lemma 4. For the second  
 1197 claim, notice that we have

$$B^\mu = W_* A^\mu \implies \mathcal{V}_0^\mu = W_* \mathcal{E}_0^\mu$$

1198 by multiplying by  $A^{\mu,\top}$  on both sides. Then the claimed invariance follows inductively.

1199 Finally, noting that  $\mathcal{E}_\ell^\mu = A_\ell^\mu A_{\ell}^{\mu,\top}$ , we immediately have

$$C_{\ell+1} = \left( I - \gamma_\ell \cdot \frac{1}{m} \sum_{\mu} \mathcal{E}_\ell^\mu \right) C_\ell = (I - \gamma_\ell \bar{\mathcal{E}}_\ell) C_\ell \quad \square$$

1200 In analogy with the centralized case, this leads us to define the following matrices.

1201 **Definition 5.** We define

$$M_\ell^\mu = \prod_{l < \ell} (I - \eta_l^\mu \mathcal{E}_l^\mu)^2 \text{ and } N_\ell^\top := \prod_{l < \ell} (I - \gamma_l \bar{\mathcal{E}}_l),$$

1202 where again  $\prod$  is interpreted as multiplication from the right, and  $M_0^\mu = N_0 = I_d$ .

1203 Succinctly, then, we can write  $\mathcal{E}_\ell^\mu = \mathcal{E}_0^\mu M_\ell^\mu$  and  $C_\ell = N_\ell C_0$ .

1204 **Telescoping the Error.** This notation allows us to set up the following analogue of Lemma 5.

1205 **Lemma 12.** *Under the distributed iterations without noise, it holds for all  $\ell$  that*

$$D_\ell = W_*(I - N_\ell)C_0.$$

1206 *Proof.* Observe that

$$D_\ell = \sum_{l < \ell} \gamma_l \cdot \sum_{\mu} \frac{\mathcal{V}_l^\mu}{m} C_l.$$

1207 Now,  $\mathcal{V}_l^\mu = W_* \mathcal{E}_l^\mu$ , and so

$$\frac{1}{m} \sum_{\mu} \mathcal{V}_l^\mu = W_* \cdot \frac{1}{m} \sum_{\mu} \mathcal{E}_l^\mu = W_* \bar{\mathcal{E}}_l.$$

1208 Thus, we have

$$D_\ell = W_* \left( \sum_{l < \ell} \gamma_l \bar{\mathcal{E}}_l N_l \right) C_0.$$

1209 But

$$N_{l+1} = (I - \gamma_l \mathcal{E}_l) N_l \iff \gamma_l \bar{\mathcal{E}}_l N_l = N_l - N_{l+1},$$

1210 and so the term in the brackets telescopes to  $N_0 - N_\ell = I - N_\ell$ . □

1211 **Conditioning.** Of course, again,  $D = W_* C = W_*(I)C_0$ . So, to gain error control, we only need to  
 1212 argue (as before) that  $\|N_\ell\|_2$  vanishes quickly with  $\ell$ . As before, this relies strongly on the condition  
 1213 number of  $\bar{\mathcal{E}}_\ell$ . We note, again, that we will simply assume that  $\bar{\mathcal{E}}_0$  is full-rank, since rank-deficiency  
 1214 is rendered moot in this case by the fact that  $C$  lies in the column span of  $A$  (and hence, is orthogonal  
 1215 to the kernel of  $\bar{\mathcal{E}}_0$ ). However, the individual  $\mathcal{E}_0^\mu$  may not be full rank, and so the distinction between  
 1216  $\underline{\lambda}(\mathcal{E}^\mu)$  and the smallest eigenvalue of  $\mathcal{E}^\mu$  (which is usually 0) should not be forgotten, although it will  
 1217 not matter very much for our expressions.

1218 To begin with, we note that since the per-machine  $A^\mu, B^\mu$  iterations are identical to the central case,  
 1219 the corresponding  $\mathcal{E}^\mu$  are conditioned at the same quadratic rate we saw previously. We formally  
 1220 state this below.

1221 **Lemma 13.** *For all  $\ell$ , set  $\eta_\ell^\mu := (\bar{\lambda}_\ell^\mu)^{-1}/3$ , and define  $\kappa_\ell^\mu = \bar{\lambda}_\ell^\mu / \underline{\lambda}_\ell^\mu$ .*

*If  $\ell \geq L(\varepsilon) := 2 + \lceil 3/5 \log(\max_{\mu} \kappa_0^\mu) \rceil + \lceil \log_2(\log(4/3\varepsilon)) \rceil$ , then  $\max_{\mu} \kappa_\ell^\mu \leq 1 + \varepsilon$ .*

1222 *Proof.* Apply Lemma 9. □

1223 Of course,  $\bar{\mathcal{E}}_\ell$ , as an average, will not have the same conditioning behaviour. In fact, this is strongly  
 1224 sensitive to the diversity index  $\alpha$  from Definition 1, as captured below.

1225 **Lemma 14.** *Suppose that  $\lambda_0^\mu$  is constant across machines, and for all  $\mu, \ell$ ,  $\eta_\ell^\mu = (\bar{\lambda}_\ell^\mu)^{-1}/3$ . Define*

$$\bar{\kappa}_\ell = \bar{\lambda}(\bar{\mathcal{E}}_\ell) / \underline{\lambda}(\bar{\mathcal{E}}_\ell).$$

1226

$$\text{If } \ell \geq L(\varepsilon), \text{ then } \bar{\kappa}_\ell \leq \frac{1 + \varepsilon}{\alpha},$$

1227 where  $L(\varepsilon)$  is the expression in Lemma 13. Further, for the same range of  $\ell$ , for all  $\mu$ ,

$$\underline{\lambda}(\bar{\mathcal{E}}_\ell) \geq \frac{\bar{\lambda}_\ell^\mu \cdot \alpha}{1 + \varepsilon}.$$

1228 *Proof.* Firstly, by Weyl's inequality, notice that

$$\bar{\lambda}(\bar{\mathcal{E}}_\ell) \leq \frac{1}{m} \sum_{\mu} \bar{\lambda}(\mathcal{E}_\ell^\mu).$$

Further, recall that  $\bar{\lambda}(\mathcal{E}_0^\mu) = 1$  for all  $\mu$ , and we set  $\eta_\ell^\mu = (\bar{\lambda}_\ell^\mu)^{-1}/3$ . Thus, each of these  $\bar{\lambda}_\ell^\mu$ s are infact identical (and equal to  $(4/9)^\ell$ ). Thus,

$$\forall \mu, \bar{\lambda}(\bar{\mathcal{E}}_\ell) \leq \bar{\lambda}_\ell^\mu.$$

Now, let  $P^\mu$  be the projection onto the column space of  $A^\mu$ . Then we note that for any vector  $v$ ,

$$v^\top \mathcal{E}_\ell^\mu v = (P^\mu v)^\top \mathcal{E}_\ell^\mu (P^\mu v).$$

Indeed, any component in  $v$  orthogonal to the column space must lie in the kernel of  $\mathcal{E}_0^\mu = A^\mu A^{\mu,\top}$ , and we know that this kernel is invariant across the iterations. Further, note that since  $P^\mu v$  lies in this column space, it is orthogonal to any eigenvector of  $\mathcal{E}_\ell^\mu$  with zero eigenvalue, and so we can then conclude that

$$(P^\mu v)^\top \mathcal{E}_\ell^\mu (P^\mu v) \geq \lambda_\ell^\mu \|P^\mu v\|_2^2.$$

But then, if  $\ell \geq L(\varepsilon)$ , then for any unit vector  $v$ , we have (for any  $\mu$  in the last inequalities) that

$$\begin{aligned} v^\top \bar{\mathcal{E}}_\ell v &= \frac{1}{m} \sum_\mu v^\top \bar{\mathcal{E}}_\ell^\mu v \\ &\geq \frac{1}{m} \sum_\mu \lambda_\ell^\mu \|P^\mu v\|_2^2 = \frac{1}{m} \sum_\mu \frac{\bar{\lambda}_\ell^\mu}{(1 + \kappa_\ell^\mu)} \|P^\mu v\|_2^2 \\ &\geq \frac{\bar{\lambda}_\ell^\mu}{1 + \varepsilon} \frac{1}{m} \sum_\mu \|P^\mu v\|_2^2 \\ &\geq \frac{\bar{\lambda}_\ell^\mu}{1 + \varepsilon} \cdot \alpha, \end{aligned}$$

where we used the equality of the  $\bar{\lambda}_\ell^\mu$ , the fact that  $\kappa_\ell^\mu \leq 1 + \varepsilon$ , and finally the definition of  $\alpha$ . Since  $v$  is a unit vector, we conclude that any Rayleigh quotient of  $\bar{\mathcal{E}}_\ell$  is so lower bound, ergo

$$\forall \mu, \lambda(\bar{\mathcal{E}}_\ell) \geq \frac{\bar{\lambda}_\ell^\mu}{1 + \varepsilon} \cdot \alpha.$$

Putting this together with the upper bound on  $\bar{\lambda}(\bar{\mathcal{E}}_\ell)$ , we immediately conclude that

$$\bar{\kappa}_\ell \leq \frac{1 + \varepsilon}{\alpha}. \quad \square$$

At a high level,  $\alpha^{-1}$  is the limiting condition number of the  $\bar{\mathcal{E}}_\ell$ s, where deviation from perfect conditioning occurs only due to how the various  $\mathcal{E}_\ell^\mu$  energise distinct subspaces for large  $\mu$ . We note that this dependence is tight—if all the  $\mathcal{E}^\mu$  share the top eigenvector, then the upper bound on  $\bar{\lambda}(\bar{\mathcal{E}}_\ell)$  is exact, while the lowest nonzero eigenvector of  $\bar{\mathcal{E}}_\ell$  is precisely the direction that achieves the minimum in the definition of the diversity index.

**Concluding the argument.** With this in hand, we can argue the decay of  $N_\ell$ , and so attain error control.

*Proof of Theorem 2.* Recall that  $D - \hat{D}_\ell = W_* N_\ell C_0$ . We again assume that  $\bar{\lambda}_0^\mu = 1$  for all  $\mu$ , and that  $\eta_\ell^\mu = (\bar{\lambda}_\ell^\mu)^{-1}/3$ . We set  $\gamma_\ell = (\bar{\lambda}_\ell^\mu)^{-1} \leq \bar{\lambda}(\bar{\mathcal{E}}_\ell)^{-1}$ . Then, as before, we have (restricted to the appropriate subspace if  $\bar{\mathcal{E}}_0$  is not full rank)

$$\|I - \eta_\ell \bar{\mathcal{E}}_\ell\|_2 \leq (1 - 1/\bar{\kappa}_\ell) \leq \exp(-1/\bar{\kappa}_\ell).$$

By Lemma 14, for  $\ell \geq L(1)$ , we have  $\bar{\kappa}_\ell \leq 2/\alpha$ , and so

$$\|N_\ell\| \leq \exp\left(-\sum_{l=L(1)}^{\ell} 1/\bar{\kappa}_l\right) \leq \exp(-\alpha(\ell - L(1))/2),$$

which is smaller than  $\iota\varepsilon$  if

$$\ell \geq L(1) + \frac{2}{\alpha} \log \frac{1}{\iota\varepsilon}.$$

Of course,  $L(1) = 3 + \lceil 3/5 \log(\max_\mu \kappa_0^\mu) \rceil$ , and setting  $\iota$  so small that error to  $\varepsilon$  follows in the same way as the proof of Theorem 1 concludes the argument.  $\square$

**Comment on Rates.** We note that, in the distributed setting, within-machine computation is typically much cheaper than across-machine communication. As a result, the protocol we have analyzed above, wherein each machine begins with the same value of  $\bar{\lambda}_0^\mu$ , is cheap to follow by using power-iteration at each machine. Given this choice, the value  $\bar{\lambda}_\ell^\mu$  is simply equal to  $(4/9)^\ell$  for every machine, and setting the learning rate  $\eta_\ell^\mu = \rho_\ell^\mu/3$  for  $\rho_\ell^\mu = (9/4)^\ell$  is trivial, and does not require any communication. We also note that setting  $\gamma_\ell = \rho_\ell^\mu$  in each machine, and then averaging the results of the updates together is sufficient, since  $\rho_\ell^\mu$  is constant across all machines. Thus, we recover exactly the structure presented in Algorithm 1, with  $\gamma = 1, \eta = 1/3$ . Note, again, that for the sake of stability, instead of working with decaying  $A_\ell$  and exploding  $\eta_\ell, \gamma_\ell$ , we can again rescale each  $A_\ell^\mu, B_\ell^\mu$  by  $3/2$  after updates.

In general, if this normalisation is not carried out, the machines may actually set their values for  $\gamma_\ell$  as distinct  $\gamma_\ell^\mu$ s. The net effect would be that we need to analyze a slightly different version of  $\bar{\mathcal{E}}_\ell$  that is sensitive to inter-machine variations in  $\gamma_\ell^\mu$ , which in turn would rely on how strongly  $\bar{\lambda}_0^\mu$  varies across machines. We leave the study of such scenarios to future work.

### F.3 Sketched Analysis

We finally turn to the sketched iterations, corresponding to Algorithm 1 with orthogonal  $S$  and  $M = 1$ . Concretely, we have the iterations

$$\begin{aligned} S_\ell &\sim \mathcal{O}(n, r) \quad \tilde{A}_\ell = A_\ell S_\ell, \tilde{B}_\ell = B_\ell S_\ell, \\ A_{\ell+1} &= A_\ell - \eta_\ell \tilde{A}_\ell \tilde{A}_\ell^\top \tilde{A}_\ell S_\ell^\top, \\ B_{\ell+1} &= B_\ell - \eta_\ell \tilde{B}_\ell \tilde{A}_\ell^\top \tilde{A}_\ell, \\ C_{\ell+1} &= (I - \gamma_\ell \tilde{A}_\ell \tilde{A}_\ell^\top) C_\ell, \\ D_{\ell+1} &= D_\ell + \gamma_\ell \tilde{B}_\ell \tilde{A}_\ell^\top C_\ell, \end{aligned} \tag{8}$$

where  $\mathcal{O}(n, r)$  denotes the following law: we first sample a  $n \times n$  matrix according to the Haar measure on the group of orthogonal matrices, and then take the submatrix formed by the first  $r$  columns.

**Mean Iteration and Bias Correction.** It turns out that this iteration, per se, suffers from nontrivial biases during the sketching process, and we shall instead analyze a corrected version of the same. To see the origin of this bias, and also to develop the core observation behind the analysis, we present the following result.

**Lemma 15.** Let  $S \sim \mathcal{O}(n, r)$ . Define  $\alpha_{n,r} = \frac{(n-2)r+nr^2}{n(n-1)(n+2)}$  and  $\beta_{n,r} = \frac{nr-r^2}{n(n-1)(n+2)}$ . For any matrix  $J \in \mathbb{R}^{d \times n}$ ,

$$\mathbb{E}[SS^\top J^\top] = \frac{r}{n} J^\top, \text{ and } \mathbb{E}[SS^\top J^\top J SS^\top] = \alpha_{n,r} J^\top J + \beta_{n,r} \text{Tr}(J^\top J) I_n.$$

*Proof.* For the first claim, we simply note that  $SS^\top$  is a projection matrix onto an isotropically chosen  $r$ -dimensional subspace. Thus, in expectation,  $SS^\top = r/n I_n$ .

Let us start by writing the singular value decomposition of  $J^\top J = U \Lambda U^\top$ , where  $U$  is an  $n \times n$  orthogonal matrix. Then the expression we are interested in is

$$\begin{aligned} \mathbb{E}[SS^\top U \Lambda U^\top SS^\top] &= U \mathbb{E}[U^\top SS^\top U \Lambda U^\top SS^\top U] U^\top \\ &= U \mathbb{E}[(U^\top S)(U^\top S)^\top \Lambda (U^\top S)(S^\top U)] U^\top, \end{aligned}$$

where we used that  $UU^\top = I_n$ , and the fact that  $U$  is nonrandom. But note that for any fixed  $U$ ,  $U^\top S$  has exactly the same distribution as  $S$ , since the group of orthogonal matrices is invariant under left multiplication by a given orthogonal matrix. Thus, we conclude that

$$\mathbb{E}[SS^\top J^\top J SS^\top] = U \mathbb{E}[SS^\top \Lambda SS^\top] U^\top.$$

Going about this in the most straightforward way possible, we see that

$$(SS^\top \Lambda SS^\top)_{ij} = \sum_k \Lambda_{kk} (SS^\top)_{ik} (SS^\top)_{kj} = \sum_{k=1}^n \sum_{l=1}^r \sum_{m=1}^r \Lambda_{kk} S_{il} S_{kl} S_{km} S_{jm}.$$

1288 Thus, this calculation relies on the computation of the fourth moments  $\mathbb{E}[S_{il}S_{kl}S_{km}S_{jm}]$ .

1289 Through a simple but tedious calculation exploiting Lemma 2.22 of Meckes (2019), this works out to

$$\begin{aligned} \mathbb{E}[(SS^\top)\Lambda SS^\top]_{ij} &= \frac{\delta_{ij}}{n(n-1)(n+2)} \sum_k \Lambda_{kk} \left( (n+1) \left( \sum_{l,m:l=m} (1 + \delta_{ki}) + \sum_{l,m} \delta_{k,i} \right) \right. \\ &\quad \left. - \left( \sum_{l,m:l=m} (1 + 3\delta_{ki}) + \sum_{l,m} (1 + \delta_{ki}) \right) \right). \end{aligned}$$

1290 Evidently, only the diagonal,  $i = j$  terms are nonzero. To compute the sum, first consider the terms  
1291 without the  $\delta_{ki}$ , through which we pick up, for every  $k$ , a factor of

$$(n+1)(r) - (r+r^2) = nr - r^2.$$

1292 Now looking at terms varying with  $\delta_{ki}$ , we have

$$(n+1)(r+r^2) - 3r - r^2 = (n-2)r + nr^2.$$

1293 In total, then,

$$\mathbb{E}[(SS^\top)\Lambda(SS^\top)]_{ii} = \frac{(n-2)r + nr^2}{n(n-1)(n+2)} \Lambda_{ii} + \frac{\text{Tr}(\Lambda)(nr - r^2)}{n(n-1)(n+2)},$$

1294 or, lifting to matrix form, and multiplying by  $U$  and  $U^\top$  on the left and right,

$$\mathbb{E}[SS^\top J^\top J SS^\top] = \alpha_{n,r} J^\top J + \beta_{n,r} \text{Tr}(J^\top J) I,$$

1295 where  $\alpha_{n,r}, \beta_{n,r}$  are as in the statement.

1296 As a sanity check, observe that when  $J^\top J = I_n$ , we get the mean

$$(\alpha_{n,r} + \beta_{n,r}) I = \frac{(n-2)r + nr^2 + n^2r - nr^2}{n(n-1)(n+2)} = \frac{(n^2 + n - 2)r}{n(n-1)(n+2)} I = \frac{r}{n} I,$$

1297 which is consistent with the fact that  $\mathbb{E}[SS^\top SS^\top] = \mathbb{E}[SS^\top] = r/n I_n$ , since  $S^\top S = I_r$  due to the  
1298 column-orthogonality of  $S$ . Similarly, if  $r = n$ , then we get  $\alpha_{n,r} = 1, \beta_{n,r} = 0$ .  $\square$

1299 The trace term above acts as a bias in the iterations (8, and leads to accumulation of errors. To avoid  
1300 this, we will instead analyze the related bias-corrected iterations

$$\begin{aligned} S_\ell &\sim \mathcal{O}(n, r) \quad \tilde{A}_\ell = A_\ell S_\ell, \tilde{B}_\ell = B_\ell S_\ell, \\ A_{\ell+1} &= A_\ell - \eta_\ell \tilde{A}_\ell \tilde{A}_\ell^\top \tilde{A}_\ell S^\top + \eta_\ell \beta_{n,r} \text{Tr}(A_\ell^\top A_\ell) A_\ell, \\ B_{\ell+1} &= B_\ell - \eta_\ell \tilde{B}_\ell \tilde{A}_\ell^\top \tilde{A}_\ell + \eta_\ell \beta_{n,r} \text{Tr}(A_\ell^\top A_\ell) B_\ell, \\ C_{\ell+1} &= C_\ell - \gamma_\ell \tilde{A}_\ell \tilde{A}_\ell^\top C_\ell, \\ D_{\ell+1} &= D_\ell + \gamma_\ell \tilde{B}_\ell \tilde{A}_\ell^\top C_\ell, \end{aligned} \tag{9}$$

1301 Let  $\mathcal{F}_\ell$  be the filtration induced by the matrices  $((Z_l, S_l)_{l \leq \ell}, Z_\ell)$ . We will denote the condition  
1302 expectation  $\mathbb{E}[\cdot | \mathcal{F}_\ell]$  as simply  $\mathbb{E}_\ell[\cdot]$ . The bias-corrected dynamics sees the following central behaviour.  
1303

1304 **Lemma 16.** *Under the dynamic (9), we have*

$$\begin{aligned} \mathbb{E}_\ell[A_{\ell+1}] &= A_\ell - \eta_\ell \alpha_{n,r} A_\ell A_\ell^\top A_\ell, \\ \mathbb{E}_\ell[B_{\ell+1}] &= B_\ell - \eta_\ell \alpha_{n,r} B_\ell A_\ell^\top A_\ell, \\ \mathbb{E}_\ell[C_{\ell+1}] &= C_\ell - \frac{r\gamma_\ell}{n} A_\ell A_\ell^\top C_\ell, \\ \mathbb{E}_\ell[D_{\ell+1}] &= D_\ell + \frac{r\gamma_\ell}{n} B_\ell A_\ell^\top C_\ell. \end{aligned}$$

1305 *Proof.* We have

$$\mathbb{E}_\ell[A_{\ell+1}] = A_\ell - \eta_\ell \mathbb{E}_\ell[A_\ell S_\ell S_\ell^\top A_\ell^\top A_\ell S_\ell S_\ell^\top] + \eta_\ell \beta_{n,r} \text{Tr}(A_\ell^\top A_\ell) A_\ell.$$

1306 Pulling  $A_\ell$  outside of the expectation, and applying Lemma 15, we get

$$\begin{aligned} \mathbb{E}_\ell[A_\ell] &= A_\ell + \eta_\ell \beta_{n,r} \text{Tr}(A_\ell^\top A_\ell) A_\ell - \eta_\ell A_\ell (\alpha_{n,r} A_\ell^\top A_\ell + \beta_{n,r} \text{Tr}(A_\ell^\top A_\ell)) \\ &= A_\ell - \eta_\ell \alpha_{n,r} A_\ell A_\ell^\top A_\ell. \end{aligned}$$

1307 Similarly, for  $B_{\ell+1}$ , we can again pull out the  $B_\ell$  from  $\mathbb{E}_\ell[\tilde{B}_\ell \tilde{A}_\ell^\top \tilde{A}_\ell] = \mathbb{E}_\ell[B_\ell S S^\top A_\ell^\top A_\ell S S^\top]$ , and  
1308 obtain the same iteration.

1309 For  $C_{\ell+1}$ , the expression instead is

$$\mathbb{E}_\ell[C_{\ell+1}] = C_\ell - \mathbb{E}_\ell[\gamma_\ell \tilde{A}_\ell \tilde{A}_\ell^\top] C_\ell = C_\ell - \gamma_\ell A_\ell \mathbb{E}_\ell[S_\ell S_\ell^\top] A_\ell C_\ell,$$

1310 and similarly,

$$\mathbb{E}_\ell[D_{\ell+1}] = D_\ell + \gamma_\ell B_\ell \mathbb{E}_\ell[S_\ell S_\ell^\top] A_\ell^\top C_\ell,$$

1311 and invoking the fact that  $\mathbb{E}_\ell[S_\ell S_\ell^\top] = r/nI$  finishes the argument.  $\square$

1312 Up to appropriately inflating the rates  $\eta_\ell, \gamma_\ell$ , this mean dynamics recovers the behaviour of the  
1313 centralised iterations. Specifically, throughout the rest of the proof we fix

$$\boxed{\eta_\ell := \frac{2}{3\alpha_{n,r}} \|A_\ell\|_2^{-2}, \quad \gamma_\ell := \frac{n}{r}.} \quad (\star)$$

1314 Note that  $\|A_0\|_2 = 1$  and the factor  $1 - \frac{2}{3}\sigma^2 \leq 1 - \frac{2}{3}$  in  $\mathbb{E}_\ell[A_{\ell+1}]$  implies  $\|A_\ell\|_2 \leq 1$  for all  $\ell$ , so  
1315 the operator- and Frobenius-norm terms that appear in the variance bounds are uniformly bounded  
1316 by 1; no additional renormalisation or clipping is required.

1317 Naturally, the next step is controlling the size of the noise fluctuations. We now briefly sketch how  
1318 this fluctuation control occurs through standard arguments.

### 1319 **High-probability convergence of the sketched iteration**

1320 Recall the iteration

$$A_{\ell+1} = A_\ell - \eta_\ell \tilde{A}_\ell \tilde{A}_\ell^\top \tilde{A}_\ell S_\ell^\top, \quad \tilde{A}_\ell = A_\ell S_\ell, \quad S_\ell \sim U(\mathcal{O}(n, r)),$$

1321 together with the coupled updates for  $B_\ell, C_\ell, D_\ell$  in (5). Set the filtration  $\mathcal{F}_\ell :=$   
1322  $\sigma(A_0, \dots, A_\ell, S_0, \dots, S_{\ell-1})$ . Write

$$\Delta_\ell := A_{\ell+1} - \mathbb{E}[A_{\ell+1} \mid \mathcal{F}_\ell].$$

1323 **Step 1 (one-step contraction in expectation).** By Lemma 16 and the choice  $\eta_\ell$  we have

$$\mathbb{E}[A_{\ell+1} \mid \mathcal{F}_\ell] = \left(I - \frac{2}{3} A A^\top\right) A_\ell \implies \|\mathbb{E}[A_{\ell+1} \mid \mathcal{F}_\ell]\|_2 \leq \frac{2}{3} \|A_\ell\|_2.$$

1324 **Step 2 (variance control).** Apply Lemma 15 *again*, this time to the centred matrix  $J := A_\ell - \frac{r}{n} A_\ell$ .  
1325 Because  $\mathbb{E}_\ell[T_\ell - \frac{r}{n} I] = 0$  and  $\|T_\ell - \frac{r}{n} I\|_2 \leq 1$ ,

$$\mathbb{E}_\ell[\Delta_\ell \Delta_\ell^\top] = \eta_\ell^2 \mathbb{E}_\ell[(T_\ell - \frac{r}{n} I) A_\ell^\top A_\ell (T_\ell - \frac{r}{n} I)] \preceq \frac{cr}{n} \|A_\ell\|_2^2 I,$$

1326 for an absolute constant  $c$ . This is the conditional variance term that enters the matrix Freedman  
1327 inequality in Step 3.

1328 **Step 3 (matrix Azuma).** Summing the martingale differences and using the matrix Freedman/Azuma  
1329 inequality yields, for every  $t > 0$ ,

$$\Pr\left[\max_{\ell \leq L} \|A_\ell - \mathbb{E} A_\ell\|_2 \geq t\right] \leq 2d \exp\left(-\frac{t^2}{2crL/n + 2t\|A_0\|_2/3}\right).$$

1330 **Step 4 (from  $A_\ell$  to  $D_\ell$ ).** Because  $B_\ell, C_\ell, D_\ell$  are linear in the same  $\tilde{A}_\ell \tilde{A}_\ell^\top \tilde{A}_\ell$  term, the same  
 1331 martingale control applies. Telescoping the error series and employing the spectral-norm bound from  
 1332 Steps 1–3 gives

$$\|D_L - \hat{D}_*\|_F \leq \sum_{\ell \geq L} \|\mathbb{E}[\Delta_\ell \mid \mathcal{F}_\ell]\|_F \leq \|B\|_F \|C\|_F \left(\frac{2}{3}\right)^{L-L_0},$$

1333 where  $L_0 = O(\frac{n}{r} \log \kappa(A_0))$  covers the “geometric phase” until  $\kappa(A_\ell) \leq 2$ . Choosing

$$L = L_0 + \left\lceil \log_{3/2} \left( \frac{\|B\|_F \|C\|_F}{\varepsilon} \right) \right\rceil + \left\lceil \log \left( \frac{2}{\delta} \right) \right\rceil$$

1334 and inserting the tail bound from Step 3 delivers

$$\Pr[\|D_L - \hat{D}_*\|_F \leq \varepsilon] \geq 1 - \delta.$$

1335 This is exactly the iteration bound stated in Theorem 3. □