Acceleration via Fractal Learning Rate Schedules

Naman Agarwal [1]   **Surbhi Goel** [2]   **Cyril Zhang** [2]

Abstract

In practical applications of iterative first-order optimization, the learning rate schedule remains notoriously difficult to understand and expensive to tune. We demonstrate the presence of these subtleties even in the innocuous case when the objective is a convex quadratic. We reinterpret an iterative algorithm from the numerical analysis literature as what we call the *Chebyshev learning rate schedule* for accelerating vanilla gradient descent, and show that the problem of mitigating instability leads to a *fractal* ordering of step sizes. We provide some experiments to challenge conventional beliefs about stable learning rates in deep learning: the fractal schedule enables training to converge with locally unstable updates which make negative progress on the objective.

1. Introduction

In the current era of large-scale machine learning models, a single deep neural network can cost millions of dollars to train. Despite the sensitivity of gradient-based training to the choice of learning rate schedule, no clear consensus has emerged on how to select this high-dimensional hyperparameter, other than expensive end-to-end model training and evaluation. Prior literature indirectly sheds some light on this mystery, showing that the learning rate schedule governs tradeoffs between accelerated convergence and various forms of algorithmic stability.

In this work, we highlight the surprising consequences of these tradeoffs in a very simple setting: first-order optimization of a convex quadratic function. We start by pointing out the existence of a non-adaptive step size schedule, derived from the roots of Chebyshev polynomials, which allows plain gradient descent to obtain accelerated convergence rates without momentum. These learning rates overshoot the region of guaranteed local progress, resulting in unsta-
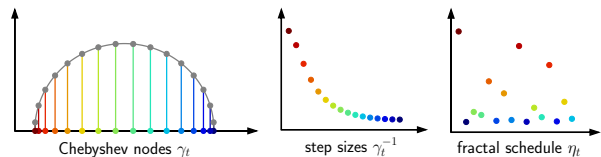

Figure 1: Visualization of the Chebyshev nodes γ_t, their corresponding step sizes γ_t^{-1}, and the fractal permutation (Lebedev & Finogenov, 1971) studied in this paper.

ble optimization trajectories. Extending a relatively obscure line of work motivated by numerical imprecision in PDE solvers (Lebedev & Finogenov, 1971), we show that stable acceleration is achieved by selecting a *fractal* permutation of the Chebyshev step sizes.

Acceleration via large step sizes may provide an useful alternative to momentum: it is less stable according to our worst-case bounds, but inherits the memory-efficiency and statelessness of vanilla gradient descent. More broadly, we discuss how this form of acceleration might implicitly present itself in settings like deep learning, introducing hidden entanglements and experimental confounds. We hope that these ideas will lead to new adaptive algorithms which overstep the "edge of stability" (the largest constant learning rate at which model training converges) (Giladi et al., 2019; Cohen et al., 2021), and accelerate training via carefully scheduled negative progress. We provide some supporting experiments towards bridging the theory-practice gap, as well as open questions for future investigation.

1.1. Our contributions

Provably stable acceleration without momentum. We revisit an oft-neglected variant of the Chebyshev iteration method for accelerating gradient descent on convex quadratics. In lieu of momentum, it uses a recursively-defined sequence of large step sizes derived from Chebyshev polynomials, which we call the fractal Chebyshev schedule. We prove a new stability guarantee for this algorithm: under bounded perturbations to all the gradients, no iterate changes by more than $O(\text{poly}(\kappa))$, where κ is the condition number of the problem. We also some provide theoretically-grounded practical variants of the schedule, and negative results for function classes beyond convex quadratics.

---

*Equal contribution [1]Google AI Princeton, Princeton, NJ, USA [2]Microsoft Research, New York, NY, USA. Correspondence to: Cyril Zhang <cyrilzhang@microsoft.com>.

Empirical insights on stable oscillating schedules. We demonstrate empirically that the fractal Chebyshev schedule stabilizes gradient descent on objectives beyond convex quadratics. We observe accelerated convergence on an instance of multiclass logistic regression, and convergent training of deep neural networks at unstable learning rates. These experiments highlight the power of optimizing the "microstructure" of the learning rate schedule (as opposed to global features like warmup and decay). We discuss how these findings connect to other implicit behaviors of SGD and learning rate schedules.

1.2. Related work

The predominant algorithms for accelerated first-order optimization are the momentum methods of Polyak (1964b) and Nesterov (1983). The former, known as the *heavy-ball* method, only achieves provable acceleration on quadratic objectives. The latter achieves minimax optimal convergence rates for general smooth convex objectives. Both are widely used in practice, far beyond their theoretical scope; for instance, they are the standard options available in deep learning frameworks.

Empirical challenges and tradeoffs. (Bottou & Bousquet, 2007) discuss the competing objectives of stability, acceleration, and computation in large-scale settings, where one cannot afford to consider a single asymptotically dominant term. Devolder et al. (2014); Chen et al. (2018); Agarwal et al. (2020b) study this specifically for acceleration. Optimizing the learning rate schedule remains a ubiquitous challenge; see Section 6.2 and Appendix G.2 for references.

Numerical methods and extremal polynomials. There are many connections between algorithm design and approximation theory (Vishnoi, 2012; Sachdeva & Vishnoi, 2013). We emphasize that the beautiful idea of the fractal permutation of Chebyshev nodes is an innovation by Lebedev & Finogenov (1971; 1973; 1976); our technical results are generalizations and refinements of the ideas therein. We give an overview of this line of work in Appendix G.1.

Learning rate schedules in stochastic optimization. Bias-variance tradeoffs in optimization are studied in various theoretical settings, including quadratics with additive and multiplicative noise (Lan, 2012; Ge et al., 2019; Gorbunov et al., 2020). Many of them also arrive at theoretically principled learning rate schedules; see Appendix G.3. On the more empirical side, Zhang et al. (2019) use a noisy quadratic model to make coarse predictions about the dynamics of large-scale neural net training. Cyclic learning rate schedules have been employed in deep learning, with various heuristic justifications (Loshchilov & Hutter, 2016; Smith, 2017; Fu et al., 2019). In parallel work, (Oymak, 2021) considers a cyclic "1 high, n low" schedule, which gives $\log(\kappa)$ convergence rates in the special case of convex quadratics whose Hessians have bimodal spectra. We discuss in Appendix E.5 why this approach does not provide acceleration in the general case; the MNIST experiments in Appendix F.4 include a comparison with this schedule.

2. Preliminaries

2.1. Gradient descent

We consider the problem of iterative optimization of a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$, with a first-order oracle $\nabla f : \mathbb{R}^d \to \mathbb{R}^d$ which computes the gradient of f at a query point. The simplest algorithm in this setting is gradient descent, which takes an arbitrary initial iterate $x_1 \in \mathbb{R}^d$ and executes T update steps

$$\{x_{t+1} \leftarrow x_t - \eta_t \nabla f(x_t)\}_{t=1}^T \tag{1}$$

according to a learning rate schedule $(\eta_1, \ldots, \eta_T)$, producing a final iterate $x_{\text{out}} := x_{T+1}$. When the $\{\eta_t\}$ do not depend on T, an analogous infinite sequence of iterates $\{x_t\}_{t \in \mathbb{N}}$ can be defined.

There are many ways to choose the learning rate schedule, depending on the structure of f and uncertainty in the gradient oracle. Some schedules are static (non-adaptive): $(\eta_1, \ldots, \eta_T)$ are chosen before the execution of the algorithm. For instance, when f is an M-smooth convex function, $\eta_t = 1/M$ achieves the classical convergence rates.

Adaptive choices of η_t are allowed to depend on the observed feedback from the current execution (including x_t and $\nabla f(x_t)$), and are considerably more expressive. For example, η_t can be chosen adaptively via line search, adaptive regularization, or curvature estimation.

2.2. The special case of quadratics

Consider the case where the objective is of the form

$$f(x) = \frac{1}{2}x^\top A x - b^\top x,$$

where $A \in \mathbb{R}^{d \times d}$ is symmetric and positive definite, and $b \in \mathbb{R}^d$, so that $\nabla f(x) = Ax - b$ is an affine function of the query point x. Then, the mapping $\mathcal{G} : x_t \mapsto x_{t+1}$ induced by gradient descent is also affine. Let $x^* := \min f$ (a fixed point of $\mathcal{G}$). Then,

$$x_{t+1} - x^* = \mathcal{G}(x_t) - x^* = \mathcal{G}(x_t) - \mathcal{G}(x^*)$$
$$= (I - \eta_t A)(x_t - x^*).$$

By induction, we conclude that

$$x_{\text{out}} - x^* = \left[\prod_{t=1}^T (I - \eta_t A) \right] (x_1 - x^*).$$

Thus, the residual after T steps of gradient descent is given by a degree-T matrix polynomial times the initial residual:

Definition 1 (Residual polynomial). Fix a choice of non-adaptive $(\eta_1, \ldots, \eta_T)$. Then, define the *residual polynomial* $p : \mathbb{R}^{d \times d} \to \mathbb{R}^{d \times d}$ as

$$p(A) := \prod_{t=1}^{T} (I - \eta_t A).$$

When clear, we will interchange to denote scalar and matrix polynomials with the same coefficients. Thus, overloading $p : \mathbb{R} \to \mathbb{R}$, we have $p(0) = 1$, and $p(1/\eta_t) = 0$ for each t.

Remark 2. The matrices in the above product all commute. Thus, when f is quadratic, $p(A)$ (and thus x_{out} given x_1) does not depend on the permutation of $(\eta_1, \ldots, \eta_T)$.

2.3. Chebyshev polynomials and Chebyshev methods

The problem of choosing $p(A)$ to optimize convergence for least-squares has roots in numerical methods for differential equations (Richardson, 1911). The Chebyshev polynomials, which appear ubiquitously in numerical methods and approximation theory (Chebyshev, 1853; Mason & Handscomb, 2002), provide a minimax-optimal solution (Flanders & Shortley, 1950; Gavurin, 1950; Young, 1953)[1]: choose positive real numbers $m \leq M$, and set

$$p(\lambda) = \frac{\mathcal{T}_T(z)}{\mathcal{T}_T(\theta)},$$

where $z := \frac{M+m-2\lambda}{M-m}$, $\theta := \frac{M+m}{M-m} = 1 + \frac{2m}{M-m}$, and $\mathcal{T}_n(\cdot)$ is the degree-n Chebyshev polynomial of the first kind. One of many equivalent definitions is $\mathcal{T}_n(z) = \cos(n \arccos z)$ for $|z| \leq 1$. From this definition it follows that the roots of p occur at the *Chebyshev nodes*

$$\gamma_t := \frac{M+m}{2} - \frac{M-m}{2} \cos \frac{(t - \frac{1}{2})\pi}{T}, \quad t = 1, \ldots, T.$$

Setting $\{\eta_t\}$ to be any permutation of $\{1/\gamma_t\}$ suffices to realize this choice of p. Note that $1/\gamma_t$ is decreasing in t. The limiting case $m = M$ is gradient descent with a constant learning rate, and $p(\lambda) = (1 - \lambda/m)^T$.

Let $\lambda_{\min}, \lambda_{\max}$ denote the smallest and largest eigenvalues of A, so that the *condition number* of A is $\kappa := \lambda_{\max}/\lambda_{\min}$. Viewing m, M as estimates for the spectrum, we define

$$\widehat{\kappa} := \frac{M}{m} \geq \frac{\lambda_{\max}}{\lambda_{\min}} = \kappa.$$

We state a classic end-to-end convergence rate for Chebyshev iteration (proven in Appendix B for completeness):

Theorem 3 (Convergence rate of Chebyshev iteration). *Choose spectral estimates $m \leq M$ such that $0 < m \leq \lambda_{\min} \leq \lambda_{\max} \leq M$. Then, setting $\{\eta_t\}$ to be any permutation of $\{1/\gamma_t\}$, the final iterate of gradient descent x_{out} satisfies the following:*

$$\|x_{\text{out}} - x^*\| \leq \frac{2\rho^T}{1 + \rho^{2T}} \|x_1 - x^*\|$$
$$\leq e^{-\Omega(T)/\sqrt{\widehat{\kappa}}} \|x_1 - x^*\|,$$

where $\rho := \frac{\sqrt{M} - \sqrt{m}}{\sqrt{M} + \sqrt{m}} \leq 1 - \Omega\left(\frac{1}{\sqrt{\widehat{\kappa}}}\right)$.

Thus, accelerated methods like Chebyshev iteration get ε-close to the minimizer in $O(\sqrt{\widehat{\kappa}} \log(1/\varepsilon))$ iterations, a quadratic improvement over the $O(\widehat{\kappa} \log(1/\varepsilon))$ rate of gradient descent with a constant learning rate. Theorem 3 is proven using approximation theory: show that $|p(\lambda)|$ is small on an interval containing the spectrum of A.

Definition 4 (Uniform norm on an interval). Let $p : \mathbb{R} \to \mathbb{R}$, and $m \leq M \in \mathbb{R}$. Define the norm

$$\|p\|_{[m,M]} := \|p\|_{L_\infty([m,M])} = \max_{\lambda \in [m,M]} |p(\lambda)|.$$

Then, any upper bound on this norm gives rise to a convergence rate like Theorem 3:

$$\|x_{\text{out}} - x^*\| \leq \|p\|_{[m,M]} \cdot \|x_1 - x^*\|.$$

These can be converted into optimality gaps on f by considering the polynomial $\lambda p^2(\lambda)$.

Moving beyond infinite-precision arithmetic, the optimization literature typically takes the route of Stiefel (1958), establishing a higher-order recurrence which "semi-iteratively" (iteratively, but keeping some auxiliary state) constructs the same final polynomial p. This is the usual meaning of the Chebyshev iteration method, and coincides with Polyak's momentum on quadratics.

This is where we depart from the conventional approach.[2] We revisit the idea of *working directly with the Chebyshev step sizes*, giving a different class of algorithms with different trajectories and stability properties.

3. The fractal Chebyshev schedule

In this section, we work in the strongly[3] convex quadratic setting from Section 2.2. Our new contributions on top of the existing theory address the following questions:

---

[1] For a modern exposition, see the blogpost http://fa.bianp.net/blog/2021/no-momentum/.

[2] For instance, this is not found in references on acceleration (Bubeck, 2017; d'Aspremont et al., 2021), or in textbooks on Chebyshev methods (Gottlieb & Orszag, 1977; Higham, 2002).

[3] Accelerated rates in this paper have $O(1/T^2)$ analogues when $\lambda_{\min} = 0$ (Allen-Zhu & Hazan, 2016).
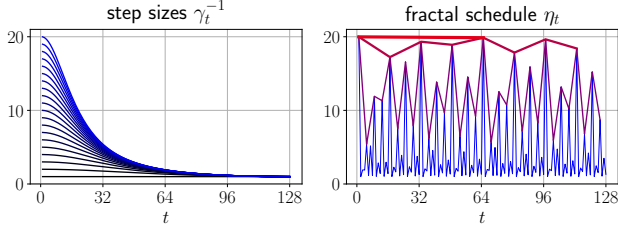
Figure 2: Shapes of the Chebyshev step sizes and fractal permutations. *Left:* Step sizes in sorted order for $M = 1$, and $m = 1, \frac{1}{2}, \ldots, \frac{1}{20}$ (black to blue). *Right:* Permuted schedule with $M = 1, m = \frac{1}{20}, T = 128$ (red). Subsequences with strides $\{1, 4, 16, 64\}$ are overlaid, demonstrating self-similarity arising from the interlacing construction.

(1) How noise-tolerant is gradient descent with Chebyshev learning rates, beyond numerical imprecision?

(2) How do we choose the ordering of steps?

We first introduce the construction originally motivated by numerical error, which provides an initial answer to (2). Then, our extended robustness analysis provides an answer to (1), and subsequently a more refined answer to (2).

3.1. Construction

We begin with the construction from (Lebedev & Finogenov, 1971), defined below and visualized in Figure 2.

Definition 5 (Fractal Chebyshev schedule). Let $\sigma_1 := [1]$, and for each $T \geq 1$ a power of 2, define

$$\sigma_{2T} := \operatorname{interlace}(\sigma_T, 2T + 1 - \sigma_T),$$

where

$$\operatorname{interlace}([a_1 \ldots a_n], [b_1 \ldots b_n]) := [a_1\ b_1\ a_2\ b_2 \ldots a_n\ b_n].$$

Then, for given $m \leq M$, and T a power of 2, the *fractal Chebyshev schedule* is the sequence of learning rates

$$\eta_t := 1/\gamma_{\sigma_T(t)}, \quad t = 1, \ldots, T.$$

Below are the first few nontrivial permutations σ_T:

$$\sigma_2 = [1\ 2],$$

$$\sigma_4 = [1\ 4\ 2\ 3],$$

$$\sigma_8 = [1\ 8\ 4\ 5\ 2\ 7\ 3\ 6],$$

$$\sigma_{16} = [1\ 16\ 8\ 9\ 4\ 13\ 5\ 12\ 2\ 15\ 7\ 10\ 3\ 14\ 6\ 11].$$

3.2. Basic properties

We first list some basic facts about the unordered step sizes:

Proposition 6. *For all $m < M$ and T, the fractal Chebyshev step sizes $\{\gamma_t^{-1}\}$ satisfy the following:*

(i) $\frac{1}{M} < \gamma_t^{-1} < \frac{1}{m} = \frac{\widehat{\kappa}}{M}$.

(ii) The number of step sizes greater than $\frac{2}{M}$ is $\left(\frac{1}{2} - \varepsilon\right) T$, where $0 \leq \varepsilon \leq O(1/\widehat{\kappa})$ as $\widehat{\kappa} \to \infty$.

(iii) For $t \leq \frac{T}{2}$, we have $\gamma_t^{-1} < \frac{1}{m + \frac{2(M-m)t^2}{T^2}}$, and

$$\frac{1}{T}\sum_{t=1}^{T} \gamma_t^{-1} = \frac{\tanh\left(T \operatorname{acosh}\left(\frac{2m}{M-m}\right)\right)}{\sqrt{Mm}} < \frac{1}{\sqrt{Mm}} = \frac{\sqrt{\widehat{\kappa}}}{M}.$$

Interpreting m, M as estimates for $\lambda_{\min}, \lambda_{\max}$:

(i) *Every* step size in the schedule exceeds the classic fixed learning rate of $1/\lambda_{\max}$. As T gets large, the largest step approaches $1/\lambda_{\min}$, a factor of κ larger.

(ii) For large κ, close to half of the step sizes *overshoot* the stable regime $\eta \in [0, 2/\lambda_{\max}]$, where local progress on f is guaranteed.

(iii) The large steps are neither highly clustered nor dispersed. The largest γ_t^{-1} overshoots the stable regime by a factor of $\Theta(\kappa)$, but the average factor is only $O(\sqrt{\kappa})$.

Next, some basic observations about the fractal schedule:

Proposition 7 (Hierarchy and self-similarity). *For all m, M, T and $0 \leq i \leq \log_2 T$:*

(i) The largest $\frac{T}{2^i}$ steps η_t in the fractal Chebyshev schedule occur when $t = 1 + 2^i(\tau - 1)$, with $\tau = 1, \ldots, \frac{T}{2^i}$.

(ii) The subsampled sequence $\{\eta_{1+2^i(\tau-1)}\}$ has the same ordering as the fractal permutation of the same length:

$$\eta_{1+2^i\tau} = \gamma_{1+2^i(\tau'-1)}^{-1}, \quad \text{where } \tau' = \sigma_{T/2^i}(\tau).$$

Figure 2 visualizes these observations, while Appendix D.1 contains formal statements and proofs.

3.3. Self-stabilization via infix polynomial bounds

Now, let us examine why the fractal ordering is needed. As discussed, in the noiseless infinite-precision setting, the final iterate x_{out} is invariant to the permutation of $\{\eta_t\}$. However, the intermediate iterates x_t depend on a sequence of *partial* products, which depend very sensitively on the permutation; Figure 3 illustrates these tradeoffs; details are found in Appendix F.1.
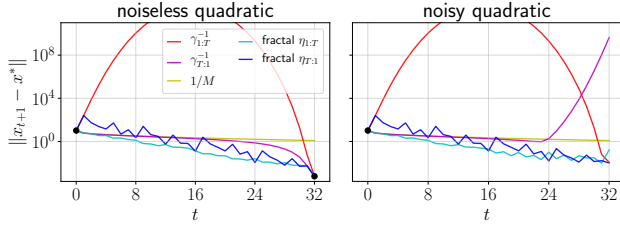
Figure 3: The optimization trajectories of various permutations of the Chebyshev step sizes. *Left:* In the noiseless case, the final iterates coincide, but x_t can wander exponentially far away. *Right:* With (i.i.d. Gaussian) noise, there is a tradeoff between $\|x_t\|$ and the stability of x_{out}.

We motivate our first new results using an additive noise model; this is a refinement of (Lebedev & Finogenov, 1971; 1973; 1976), which are only concerned with preventing exponential blowup of negligible perturbations at the numerical noise floor. We consider adding a sequence of perturbations $(\xi_1, \ldots, \xi_T)$ to gradient descent (Equation 1):

$$\{x_{t+1} \leftarrow x_t - \eta_t \nabla f(x_t) + \xi_t\}_{t=1}^{T}. \tag{2}$$

Note that this captures an inexact (e.g. stochastic) gradient oracle $\widetilde{\nabla} f(\cdot)$, in which case

$$\xi_t = \eta_t(\nabla f(x_t) - \widetilde{\nabla} f(x_t)). \tag{3}$$

Unrolling the recursion, we get:

$$x_2 - x^* = (I - \eta_1 A)(x_1 - x^*) + \xi_1,$$
$$x_3 - x^* = (I - \eta_2 A)\left[(I - \eta_1 A)(x_1 - x^*) + \xi_1\right] + \xi_2,$$
$$\cdots$$
$$x_t - x^* = p_{1:t-1}(A)(x_1 - x^*) + \sum_{t'=2}^{t} p_{t':t-1}(A)\xi_{t'-1},$$

where we have defined the *infix polynomial* as the (possibly empty) product

$$p_{s:t}(A) := \prod_{\tau=s}^{t}(I - \eta_\tau A).$$

Lebedev & Finogenov (1971) give bounds on the norms of the *prefix polynomials* $p_{1:t}$ and *suffix polynomials* $p_{s:T}$:

Theorem 8 (Prefix and suffix bounds). *For a fractal Chebyshev schedule with m, M, T, and all $1 \leq s \leq t \leq T$:*

(i) $\|p_{1:t}\|_{[m,M]} \leq \frac{\widehat{\kappa}-1}{4^{\min(\text{bits}(t))}} \prod_{j \in \text{bits}'(t)} \frac{2}{1+\mathcal{T}_{2^j}(\theta)};$

(ii) $\|p_{s:T}\|_{[m,M]} \leq \prod_{j \in \text{bits}(T+1-s)} \frac{2}{1+\mathcal{T}_{2^j}(\theta)},$

where $\text{bits}(n)$ *denotes the sequence* $j_1 > j_2 > \ldots > j_k$ *of indices in the binary expansion of n, and* $\text{bits}'(n) :=$

$\text{bits}(n)\backslash j_k$. *For example, when $n = 6 = 2^2 + 2^1$, $\text{bits}(n) = \{2,1\}$, and $\text{bits}'(n) = \{2\}$.*

Let $\mathcal{V}(\cdot), \mathcal{V}'(\cdot)$ denote the bounds from Theorem 8, so that $\|p_{1:t}\|_{[m,M]} \leq \mathcal{V}'(t)$, and $\|p_{s:T}\|_{[m,M]} \leq \mathcal{V}(T+1-s)$. Notice that $\mathcal{V}(t) \leq \frac{2}{1+\mathcal{T}_{\lfloor t/2 \rfloor}(\theta)} \leq e^{-\Omega(t)/\sqrt{\widehat{\kappa}}}$ for all $t \geq 1$, and $\mathcal{V}'(t) \leq \widehat{\kappa}\mathcal{V}(t)$.

To fully understand the propagation of ξ_t through Equation 2, we provide bounds on the infix polynomial norms:

Theorem 9 (Infix polynomial bounds). *For the fractal Chebyshev schedule with m, M, T, and all $1 \leq s \leq t \leq T$:*

$$\|p_{s:t}\|_{[m,M]} \leq \mathcal{V}(\zeta + 1 - s) \cdot \mathcal{V}'(t - \zeta),$$

where ζ is the index such that $s - 1 \leq \zeta \leq t$ and $\zeta, \zeta + 1$ differ at the most significant bit.

Then, analyzing the decay of $\mathcal{V}, \mathcal{V}'$, we derive cumulative error bounds:

Theorem 10 (Infix series bounds). *For a fractal Chebyshev schedule with m, M, T, and all $1 \leq s \leq t \leq T$:*

$$\sum_{t'=s}^{t} \|p_{t':t}\|_{[m,M]} \leq O\left(\widehat{\kappa}^{1+\frac{1}{\ln 4}} \log \widehat{\kappa}\right) = o\left(\widehat{\kappa}^{1.73}\right).$$

This bound, a sum of up to T terms, is independent of T.

These require generalizations of the combinatorial proofs for Theorem 8, presented (along with more precise statements) in Appendices D.2 and D.3.

3.4. Implications for gradient descent

Theorem 10 translates to the following end-to-end statement about gradient descent with the fractal schedule:

Corollary 11. *Suppose $0 < m \leq \lambda_{\min} \leq \lambda_{\max} \leq M$. Then, gradient descent with the fractal Chebyshev schedule of length T, and perturbations (as in Equation 2) such that $\|\xi_t\| \leq \varepsilon$, outputs iterates x_t satisfying*

$$\|x_{t+1} - x^*\| \leq \|p_{1:t}\|_{[m,M]} \cdot \|x_1 - x^*\| + o(\widehat{\kappa}^{1.73}) \cdot \varepsilon.$$

Recall that Theorems 8 and 3 guarantee

$$\|p_{1:t}\|_{[m,M]} \leq e^{-\Omega(T) \cdot \log(\widehat{\kappa})/\sqrt{\widehat{\kappa}}};$$
$$\|p_{1:T}\|_{[m,M]} \leq e^{-\Omega(T)/\sqrt{\widehat{\kappa}}}.$$

The fractal schedule allows the stability factor to be independent of T. When the perturbations arise from noisy gradients (as in Equation 3), so that each ξ_t is $\eta_t \varepsilon$-bounded, this factor becomes $o(\widehat{\kappa}^{2.73})$.

Provable benefit of negative progress. A striking fact about the fractal Chebyshev schedule is that this *non-adaptive* method provably beats the minimax convergence rate of line search, the most fundamental *adaptive* algorithm in this setting (Boyd & Vandenberghe, 2004):

$$\eta_t^{(\mathrm{ls})} := \underset{\eta \, \geq \, 0}{\arg\min} \, f(x_t - \eta \nabla f(x_t)). \tag{4}$$

Proposition 12 (No acceleration from line search). *On a strongly convex quadratic objective $f(x) = \frac{1}{2}x^\top A x + b^\top x$, let $\{x_t\}$ be the sequence of iterates of gradient descent with the adaptive learning rate schedule $\eta_t^{(\mathrm{ls})}$ from Equation 4. Then, for each A, b, there exists a setting of x_1 such that*

$$\|x_{t+1} - x^*\| \; \geq \; \left(1 - \frac{1}{\Omega(\kappa)}\right)^T \cdot \|x_1 - x^*\|, \quad \forall t \; \geq \; 1.$$

This is a classic fact; for a complete treatment, see Section 3.2.2 of (Kelley, 1999). In the context of our results, it shows that greedily selecting the locally optimal learning rates is provably suboptimal, even compared to a feedback-independent policy.

Adaptive estimation of the local loss curvature is an oft-attempted approach, amounting to finding the best conservative step size $\frac{1}{M}$. Proposition 12 suggests that although such methods have numerous advantages, greedy local methods can miss out on acceleration. The fact that acceleration can be obtained from carefully scheduled overshooting is reminiscent of simulated annealing (Aarts & Korst, 1989), though we could not find any rigorous connections.

Comparison with momentum. We stress that this form of acceleration does not replace or dominate momentum. The dependence of the stability term on $\widehat{\kappa}$ is suboptimal (Devolder et al., 2014). In exchange, we get a *memory-less* acceleration algorithm: gradient descent has no auxiliary variables or multi-term recurrences, so that x_t fully specifies the state. This bypasses the subtleties inherent in restarting stateful optimizers (O'Donoghue & Candes, 2015; Loshchilov & Hutter, 2016).

Finally, our theory (especially Theorem 14) implies that experiments attempting to probe the acceleration benefits of momentum might be confounded by the learning rate schedule, even in the simplest of settings (thus, certainly also in more complicated settings, like deep learning).

3.5. Brief overview of proof ideas

Figure 3 suggests that there is a tradeoff between taking large $\Omega(1/m)$ steps for acceleration vs. small $O(1/M)$ steps for stability. To get acceleration, we must take all of the large steps in the schedule. However, we must space them out: taking $k = o(T)$ of the largest steps consecutively

incurs an exponential blowup in the infix polynomial:

$$\prod_{i=1}^{k} \left\| \left(1 - \frac{\lambda}{\gamma_i}\right) \right\|_{[m,M]} \approx \left\| \left(1 - \frac{\lambda}{m}\right)^k \right\|_{[m,M]} = (\widehat{\kappa} - 1)^k.$$

The difficulty arises from the fact that there are not enough small steps in the schedule, so that a large step will need to be stabilized by *internal copies of Chebyshev iteration*. This is why the fractal schedule is necessary. Theorem 9 shows that this is surprisingly possible: the fractal schedule is only as unstable as the largest single step.

This intuition does not get us very far towards an actual proof: the internal copies of Chebyshev iteration, which form a complete binary tree, are "skewed" in a way that is sometimes better, sometimes worse. Isolating a combinatorial *tree exchange lemma* used to prove Theorem 8, we can iteratively swap two special infix polynomials with two others, and localize "bad skewness" to only one large step. Theorem 9 follows from decomposing each infix into two infixes amenable to the tree exchange procedure. Theorem 10 follows by combining Theorem 9 with sharpened generalizations of the original paper's series bounds.

The proofs involve delicate trigonometric inequalities and various interesting facts about the geometry of polynomials. Appendices B, C, and D build up to self-contained proofs.

4. Extensions and variants

Next, we explore some theoretically justified variants.

4.1. Useful transformations of the fractal schedule

Reversing the schedule. Notice that the first step η_1 is the largest step in the schedule. This might not be desirable when ξ_t is proportional to $\|x - x^*\|$ (like in linear regression with minibatch SGD noise). It is a simple consequence of the symmetries in the main theorems that reversing the fractal Chebyshev schedule produces a contractive variant:

Proposition 13. *Suppose we run gradient descent with the reversed fractal Chebyshev schedule $\sigma_T(T + 1 - t)$. Then:*

(i) *For any $1 \, \leq \, t < t' \, \leq \, T$, we have*

$$\overline{\|p_{1:t}\|}_{[m,M]} \; \leq \; \overline{\|p_{1:t'}\|}_{[m,M]} \; \leq \; 1,$$

where $\overline{\|\cdot\|}$ denotes the corresponding suffix norm bound from Theorem 8 (ii).

(ii) *The bounds from Theorem 8 are swapped: replace $(p_{1:t}, p_{s:T}) \to (p_{T+1-t:T}, p_{1:T+1-s})$.*

(iii) *Theorem 9 holds, swapping $\mathcal{V} \leftrightarrow \mathcal{V}'$. Theorem 10 holds.*

Concatenating schedules. One can also repeat the fractal Chebyshev schedule indefinitely.[4] Note that each infix polynomial of a repeated schedule can be written as a product of one prefix $p_{1:t}$, one suffix $p_{s:T}$, and a power of $p_{1:T}$, so stability bounds analogous to Theorems 9 and 10 follow straightforwardly. It is also possible to concatenate schedules with different lengths T. Choosing T to be successive powers of 2, one obtains an infinitely long schedule suitable for unknown time horizons.

4.2. Conservative overstepping and partial acceleration

In this section, we decouple the eigenvalue range $[\lambda_{\min}, \lambda_{\max}]$ from the Chebyshev node range $[m, M]$ used in constructing the schedule. This can simply arise from an incorrect estimation of the eigenvalue range. However, more interestingly, if we think of $[m, M]$ as purposefully omitting the lower spectrum of A (and thus taking smaller large steps), this allows us to interpolate between the fractal Chebyshev schedule and the vanilla constant learning rate.

Easy cases. If $m < \lambda_{\min}$ or $M > \lambda_{\max}$, then $[m, M]$ is still an interval containing the spectrum of A; it is simply the case that convergence rates and stability bounds will depend on a worse $\widehat{\kappa} > \kappa$. On the other hand, if $M < \lambda_{\max}$, the residual blows up exponentially.

The subtle case is when $m > \lambda_{\min}$, when we are overstepping with restraint, trading off acceleration for stability via more conservative step sizes. This requires us to reason about $\|p\|_{[\lambda_{\min}, M]}$ when p was constructed to shrink $\|p\|_{[m, M]}$. Analyzing this case, we get *partial* acceleration:

Theorem 14. *Given a quadratic objective with matrix A and $0 < \lambda_{\min} \leq m \leq \lambda_{\max} \leq M$, gradient descent with the Chebyshev step sizes results in the following convergence guarantee:*

$$\|x_{\text{out}} - x^*\| \leq 2 \left(1 - \phi^{-1}(\lambda_{\min}, m, M)\right)^T \cdot \|x_1 - x^*\|,$$

with

$$\phi^{-1}(\lambda_{\min}, m, M)$$
$$:= 2 \cdot \frac{\lambda_{\min} + \sqrt{Mm} - \sqrt{(M - \lambda_{\min})(m - \lambda_{\min})}}{(\sqrt{M} + \sqrt{m})^2}.$$

This is an interpolation between the standard and accelerated convergence rates of $O(\kappa \log(1/\varepsilon))$ and $O(\sqrt{\kappa} \log(1/\varepsilon))$. Figure 4 shows the shape of ϕ for $m \in [\lambda_{\min}, M]$, as it ranges from $\sim \sqrt{\kappa} \to \kappa$.

---

[4]This is known as a cyclic iterative method, and was in fact the original motivation for (Lebedev & Finogenov, 1971).
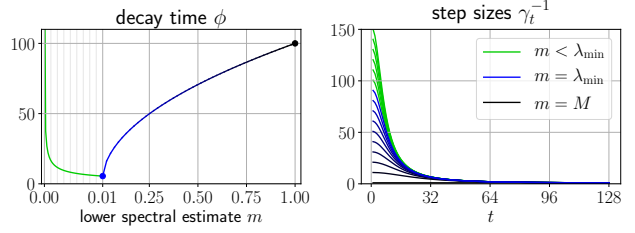

Figure 4: Summary of the discussion in Section 4.2. Suboptimal decay times $\phi(\lambda_{\min} = 0.01, m, M = 1)$ interpolate between the standard and accelerated rates. Green curves correspond to settings of $m < \lambda_{\min}$ where Theorem 3 applies; notice the distorted horizontal scale.

4.3. Existence of clairvoyant non-adaptive schedules

Finally, we present one more view on the provable power of tuning (i.e. searching globally for) a learning rate schedule on a fixed problem instance. An ambitious benchmark is the conjugate gradient method (Hestenes & Stiefel, 1952), which is optimal for *every* (rather than the worst-case) choice of A, b. That is, at iteration t, it outputs

$$x_{t+1} := \arg\min_{\substack{\deg p \leq t \\ p(0)=1}} \|p(A)(x_1 - x^*)\|_A,$$

where $\|x\|_A := \sqrt{x^\top A x}$. This can be much stronger than the guarantee from Theorem 3 (e.g. when the eigenvalues of A are clustered). In Appendix E.3, we prove that there are non-adaptive (but instance-dependent) learning rate schedules that compete with conjugate gradient:

Theorem 15 (Conjugate gradient schedule; informal)**.** *For every problem instance (A, b), there is a learning rate schedule $\{\eta_t\}$ for gradient descent, with each $\eta_t \in [\frac{1}{\lambda_{\max}}, \frac{1}{\lambda_{\min}}]$, such that x_{out} is the output of conjugate gradient.*

5. Beyond convex quadratics

5.1. General convex objectives: a counterexample

A mysterious fact about acceleration is that some algorithms and analyses transfer from the quadratic case to general convex functions, while others do not. (Lessard et al., 2016) exhibit a smooth and strongly convex non-quadratic f for which Polyak's momentum gets stuck in a limit cycle.

For us, $f(x) = \log \cosh(x) + 0.01x^2$ serves as a one-dimensional "proof by simulation" that gradient descent with the fractal Chebyshev schedule can fail to converge. This is shown in Appendix F.2; note that this is a tiny instance of ridge logistic regression.

5.2. Non-convex objectives: a no-go

None of this theory carries over to worst-case non-convex f: the analogue of Theorem 15 is vacuously strong. We point out that global optimization of the learning rate schedule is information-theoretically intractable.

Proposition 16 (Non-convex combination lock; informal)**.** *For every "passcode" $\{\eta_1^*, \ldots, \eta_T^*\}$ and $\delta > 0$, there is a smooth non-convex optimization problem instance $(f(\cdot), x_1)$ for which the final iterate x_{out} of gradient descent is an 1-approximate global minimum only if*

$$|\eta_t - \eta_t^*| \leq \delta, \quad \forall t = 1, \ldots, T.$$

A formal statement and proof are given in Appendix E.4.

5.3. More heuristic building blocks

With Polyak momentum as the most illustrious example, an optimizer can be very useful beyond its original theoretical scope. We present some more ideas for heuristic variants (unlike the theoretically justified ones from Section 4):

Cheap surrogates for the fractal schedule. The worst-case guarantees for Chebyshev methods depend sensitively on the choice of nodes. However, beyond worst-case objectives, it might suffice to replace $\{\gamma_t^{-1}\}$ with any similarly-shaped distribution (like the triangular one considered by (Smith, 2017)), and σ with any sequence that sufficiently disperses the large steps. We show in Appendix E.5 that acceleration cannot arise from the simple cyclic schedule from (Oymak, 2021). An intriguing question is whether adaptive gradient methods or the randomness of SGD implicitly causes partial acceleration, alongside other proposed "side effect" mechanisms (Keskar et al., 2016; Jin et al., 2017; Staib et al., 2019).

Inserting slow steps. We can insert any number of steps $\eta \in [0, \frac{2}{M}]$ at any point in a schedule without worsening stability or convergence, because $\|(1 - \eta\lambda)\|_{[m,M]} \leq 1$. That is, $\|p_{s':t'}\|$ in the supersequence is bounded by the corresponding $\|p_{s:t}\|$ in the original schedule, and Theorems 9 and 10 apply. A special case of this is *warmup* or *burn-in*: take any number of small steps at the beginning.

Another option is to insert the small steps cyclically: notice from Propositions 6 (ii) and 7 (i) that the steps $\{\eta_t\}$ come in "fast-slow" pairs: an odd step overshoots, and an even step corrects it. This suggests further heuristics, like the following "Chebyshevian waltz": in minibatch SGD, run triplets of iterations with step sizes $(\eta_{2t-1}, \eta_{2t}, \frac{1}{M})$.[5] In

---

[5]In non-GPU-bound regimes (Choi et al., 2019; Agarwal et al., 2020a) and deep RL, one can sometimes take these steps for free, without causing a time bottleneck.
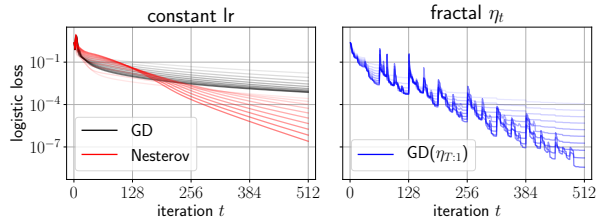

Figure 5: Logistic regression/MNIST training loss curves. *Left:* Standard algorithms, with constant (more opaque = larger) learning rates. *Right:* A fractal Chebyshev schedule.
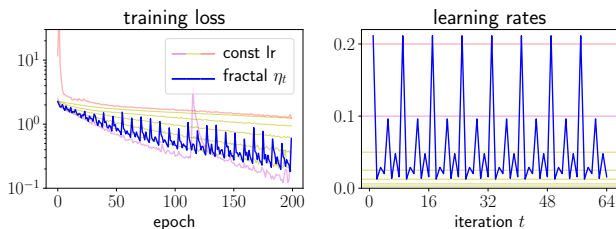

Figure 6: ResNet-18/CIFAR-10 training with batch size 8192 and a repeated $T = 8$ fractal Chebyshev schedule. *Left:* Training loss curves. *Right:* Learning rates; the schedule pokes through the edge of stability (magenta and red) without destabilizing training.

theory, this degrades the worst-case convergence rate by a constant factor, but improves stability by a constant factor.

6. Experiments

6.1. Convex problems and non-local progress

In spite of the simple negative result in Section 5.1, we find that the fractal Chebyshev schedule can exhibit accelerated convergence beyond quadratic objectives. Figure 5 shows training curves for logistic regression for MNIST classification; details are in Appendix F.3. We leave a theoretical characterization of the schedule's acceleration properties on general convex functions to future work; this may require further assumptions on "natural" problem instances beyond minimax bounds.

6.2. Beyond the edge of stability in deep learning

We provide a small set of deep learning experiments, finding that the fractal Chebyshev schedule can overstep the empirical "edge of stability" (i.e. the largest constant multiplier on the learning rate for which training does not diverge). Figure 6 gives an overview of these findings; details are in Appendix F.4.

Estimating the scale of $\lambda_{\max}(\nabla^2 f)$ is an old paradigm for selecting learning rates (LeCun et al., 1992; Schaul et al.,

2013); there are many proposed mechanisms for the success of larger learning rates. Our theory (especially Theorem 14) and experiments point to the possibility of *time-varying* schedules to enable larger learning rates, on a much finer scale than cyclic restarts (Loshchilov & Hutter, 2016; Smith, 2017; Fu et al., 2019). A nascent line of work also challenges the classical $\eta_t \sim 1/\lambda_{\max}$ wisdom from an empirical angle (Cohen et al., 2021), finding a phenomenon dubbed *progressive sharpening* during normal (smooth η_t) training.

End-to-end improvements on training benchmarks are outside the scope of this work: the learning rate schedule interacts with generalization (Jiang et al., 2020), batch normalization + weight decay (Li & Arora, 2019), batch size (Smith et al., 2018), adaptive preconditioners (Agarwal et al., 2020a) and now (from this work) acceleration. This adds yet one more perspective on why it is so difficult to standardize experimental controls and ablations in this space. Analogously, it has been proposed that momentum acts as a variance reduction mechanism (Li et al., 2017; Cutkosky & Orabona, 2019), alongside its classical role in acceleration.

As an invitation to try these ideas in various experimental settings, we provide in Appendix A some Python code to generate Chebyshev learning rates and fractal schedules.

7. Conclusion

We have revisited a lesser-known acceleration algorithm which uses a fractal learning rate schedule of reciprocal Chebyshev nodes, proved a stronger stability guarantee for its iterates, and developed some practical variants. Our experiments demonstrate promising empirical behaviors of the schedule beyond low-noise quadratics. We hope that this work provides new foundations towards investigating local optimization algorithms which take carefully scheduled "leaps of faith".

Open questions. We conclude with some natural followup questions for future work:

- Find "reasonable"[6] (computationally efficient, oracle-efficient, and perturbation-stable) adaptive learning rate schedulers with accelerated convergence rates. What are the acceleration properties of commonly-used adaptive step size heuristics (Duchi et al., 2011; Kingma & Ba, 2014; Ward et al., 2019)?

- Do there exist learning rate schedules (adaptive or non-adaptive) which obtain the accelerated rate for general strongly convex f, as opposed to only quadratics?

---

[6]One example which is unreasonable in every way: run conjugate gradient ahead of time, maintaining monomial-basis expansions of the A-orthogonal basis. Compute the roots of the final polynomial, and use their inverses as a learning rate schedule.

References

Aarts, E. and Korst, J. *Simulated annealing and Boltzmann machines: a stochastic approach to combinatorial optimization and neural computing*. John Wiley & Sons, Inc., 1989.

Agarwal, N., Anil, R., Hazan, E., Koren, T., and Zhang, C. Disentangling adaptive gradient methods from learning rates. *arXiv preprint arXiv:2002.11803*, 2020a.

Agarwal, N., Anil, R., Koren, T., Talwar, K., and Zhang, C. Stochastic optimization with laggard data pipelines. In *Advances in Neural Information Processing Systems*, volume 33, 2020b.

Aitken, A. C. XXV.—On Bernoulli's numerical solution of algebraic equations. *Proceedings of the Royal Society of Edinburgh*, 46:289–305, 1927.

Allen-Zhu, Z. and Hazan, E. Optimal black-box reductions between optimization objectives. *arXiv preprint arXiv:1603.05642*, 2016.

Allen-Zhu, Z. and Orecchia, L. Linear coupling: An ultimate unification of gradient and mirror descent. *arXiv preprint arXiv:1407.1537*, 2014.

Anderson, D. G. Iterative procedures for nonlinear integral equations. *Journal of the ACM (JACM)*, 12(4):547–560, 1965.

Bach, F. Machine learning research blog, 2020. URL https://francisbach.com/acceleration-without-pain/.

Barré, M., Taylor, A., and d'Aspremont, A. Complexity guarantees for polyak steps with momentum. In *Conference on Learning Theory*, pp. 452–478. PMLR, 2020.

Bello, I., Zoph, B., Vasudevan, V., and Le, Q. V. Neural optimizer search with reinforcement learning. In *International Conference on Machine Learning*, pp. 459–468. PMLR, 2017.

Bottou, L. and Bousquet, O. The tradeoffs of large scale learning. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, pp. 161–168, 2007.

Bousquet, O. and Elisseeff, A. Stability and generalization. *The Journal of Machine Learning Research*, 2:499–526, 2002.

Boyd, S. and Vandenberghe, L. *Convex optimization*. Cambridge University Press, 2004.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Bubeck, S. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 8, 2017.

Bubeck, S. Nemirovski's acceleration (blog post), 2019. URL https://blogs.princeton.edu/imabandit/2019/01/09/nemirovskis-acceleration/.

Bubeck, S., Lee, Y. T., and Singh, M. A geometric alternative to nesterov's accelerated gradient descent. *arXiv preprint arXiv:1506.08187*, 2015.

Bubeck, S., Jiang, Q., Lee, Y. T., Li, Y., and Sidford, A. Near-optimal method for highly smooth convex optimization. In *Conference on Learning Theory*, pp. 492–507. PMLR, 2019.

Chebyshev, P. L. *Théorie des mécanismes connus sous le nom de parallélogrammes*. Imprimerie de l'Académie impériale des sciences, 1853.

Chen, Y., Jin, C., and Yu, B. Stability and convergence tradeoff of iterative optimization algorithms. *arXiv preprint arXiv:1804.01619*, 2018.

Choi, D., Passos, A., Shallue, C. J., and Dahl, G. E. Faster neural network training with data echoing. *arXiv preprint arXiv:1907.05550*, 2019.

Cohen, J., Kaur, S., Li, Y., Kolter, J. Z., and Talwalkar, A. Gradient descent on neural networks typically occurs at the edge of stability. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=jh-rTtvkGeM.

Cutkosky, A. and Orabona, F. Black-box reductions for parameter-free online learning in banach spaces. In *Conference On Learning Theory*, pp. 1493–1529. PMLR, 2018.

Cutkosky, A. and Orabona, F. Momentum-based variance reduction in non-convex sgd. *arXiv preprint arXiv:1905.10018*, 2019.

d'Aspremont, A., Scieur, D., and Taylor, A. Acceleration methods. *arXiv preprint arXiv:2101.09545*, 2021.

Devolder, O., Glineur, F., and Nesterov, Y. First-order methods of smooth convex optimization with inexact oracle. *Mathematical Programming*, 146(1):37–75, 2014.

Dozat, T. Incorporating nesterov momentum into adam. 2016.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

Flanders, D. A. and Shortley, G. Numerical determination of fundamental modes. *Journal of Applied Physics*, 21 (12):1326–1332, 1950.

Fu, H., Li, C., Liu, X., Gao, J., Celikyilmaz, A., and Carin, L. Cyclical annealing schedule: A simple approach to mitigating kl vanishing. *arXiv preprint arXiv:1903.10145*, 2019.

Gavurin, M. K. The use of polynomials of best approximation for improving the convergence of iterative processes. *Uspekhi Matematicheskikh Nauk*, 5(3):156–160, 1950.

Ge, R., Kakade, S. M., Kidambi, R., and Netrapalli, P. The step decay schedule: A near optimal, geometrically decaying learning rate procedure for least squares. *Advances in Neural Information Processing Systems*, 32: 14977–14988, 2019.

Giladi, N., Nacson, M. S., Hoffer, E., and Soudry, D. At stability's edge: How to adjust hyperparameters to preserve minima selection in asynchronous training of neural networks? In *International Conference on Learning Representations*, 2019.

Gorbunov, E., Hanzely, F., and Richtárik, P. A unified theory of sgd: Variance reduction, sampling, quantization and coordinate descent. In *International Conference on Artificial Intelligence and Statistics*, pp. 680–690. PMLR, 2020.

Gottlieb, D. and Orszag, S. A. *Numerical analysis of spectral methods: theory and applications*. SIAM, 1977.

Hardt, M., Recht, B., and Singer, Y. Train faster, generalize better: Stability of stochastic gradient descent. In *International Conference on Machine Learning*, pp. 1225–1234. PMLR, 2016.

Hazan, E. and Kakade, S. Revisiting the polyak step size. *arXiv preprint arXiv:1905.00313*, 2019.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016.

Hestenes, M. R. and Stiefel, E. Methods of conjugate gradients for solving linear systems. *Journal cf Research of the National Bureau of Standards*, 49(6), 1952.

Higham, N. J. *Accuracy and stability of numerical algorithms*. SIAM, 2002.

Jiang, Z., Zhang, C., Talwar, K., and Mozer, M. C. Characterizing structural regularities of labeled data in overparameterized models. *arXiv e-prints*, pp. arXiv–2002, 2020.

Jin, C., Ge, R., Netrapalli, P., Kakade, S. M., and Jordan, M. I. How to escape saddle points efficiently. In *International Conference on Machine Learning*, pp. 1724–1732. PMLR, 2017.

Kelley, C. T. *Iterative methods for optimization*. SIAM, 1999.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Lan, G. An optimal method for stochastic composite optimization. *Mathematical Programming*, 133(1-2):365–397, 2012.

Lebedev, V. and Finogenov, S. Solution of the parameter ordering problem in chebyshev iterative methods. *USSR Computational Mathematics and Mathematical Physics*, 13(1):21–41, 1973.

Lebedev, V. and Finogenov, S. Utilization of ordered chebyshev parameters in iterative methods. *USSR Computational Mathematics and Mathematical Physics*, 16(4): 70–83, 1976.

Lebedev, V. and Finogenov, S. On construction of the stable permutations of parameters for the chebyshev iterative methods. part i. *Russian Journal of Numerical Analysis and Mathematical Modelling*, 17(5):437–456, 2002.

Lebedev, V. and Finogenov, S. On construction of the stable permutations of parameters for the chebyshev iterative methods. part ii. *Russian Journal of Numerical Analysis and Mathematical Modelling*, 19(3):251–263, 2004.

Lebedev, V. I. and Finogenov, S. The order of choice of the iteration parameters in the cyclic Chebyshev iteration method. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 11(2):425–438, 1971.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

LeCun, Y., Simard, P. Y., and Pearlmutter, B. Automatic learning rate maximization by on-line estimation of the hessian's eigenvectors. In *Proceedings of the 5th International Conference on Neural Information Processing Systems*, pp. 156–163, 1992.

Lessard, L., Recht, B., and Packard, A. Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM Journal on Optimization*, 26(1):57–95, 2016.

Li, Q., Tai, C., and Weinan, E. Stochastic modified equations and adaptive stochastic gradient algorithms. In *International Conference on Machine Learning*, pp. 2101–2110. PMLR, 2017.

Li, Y., Wei, C., and Ma, T. Towards explaining the regularization effect of initial large learning rate in training neural networks. *arXiv preprint arXiv:1907.04595*, 2019.

Li, Z. and Arora, S. An exponential learning rate schedule for deep learning. *arXiv preprint arXiv:1910.07454*, 2019.

Li, Z. and Li, J. A fast anderson-chebyshev acceleration for nonlinear optimization. In *International Conference on Artificial Intelligence and Statistics*, pp. 1047–1057. PMLR, 2020.

Li, Z., Lyu, K., and Arora, S. Reconciling modern deep learning with traditional optimization analyses: The intrinsic learning rate. *arXiv preprint arXiv:2010.02916*, 2020.

Lin, H., Mairal, J., and Harchaoui, Z. Catalyst acceleration for first-order convex optimization: from theory to practice. *Journal of Machine Learning Research*, 18(1): 7854–7907, 2018.

Liu, D. C. and Nocedal, J. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

Loshchilov, I. and Hutter, F. SGDR: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Mason, J. C. and Handscomb, D. C. *Chebyshev polynomials*. CRC press, 2002.

Monteiro, R. D. and Svaiter, B. F. An accelerated hybrid proximal extragradient method for convex optimization and its implications to second-order methods. *SIAM Journal on Optimization*, 23(2):1092–1125, 2013.

Nesterov, Y. Accelerating the cubic regularization of newton's method on convex problems. *Mathematical Programming*, 112(1):159–181, 2008.

Nesterov, Y. E. A method of solving a convex programming problem with convergence rate o(k^2). In *Doklady Akademii Nauk*, volume 269, pp. 543–547. Russian Academy of Sciences, 1983.

Orabona, F. and Tommasi, T. Training deep networks without learning rates through coin betting. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 2157–2167, 2017.

Oymak, S. Super-convergence with an unstable learning rate. *arXiv preprint arXiv:2102.10734*, 2021.

O'Donoghue, B. and Candes, E. Adaptive restart for accelerated gradient schemes. *Foundations of computational mathematics*, 15(3):715–732, 2015.

Pedregosa, F. and Scieur, D. Acceleration through spectral density estimation. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7553–7562. PMLR, 13–18 Jul 2020. URL http://proceedings.mlr.press/v119/pedregosa20a.html.

Polyak, B. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964a. ISSN 0041-5553.

Polyak, B. T. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964b.

Polyak, B. T. Introduction to optimization. optimization software. *Inc., Publications Division, New York*, 1, 1987.

Richardson, L. F. The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 210(459-470):307–357, 1911.

Sachdeva, S. and Vishnoi, N. K. Faster algorithms via approximation theory. *Theoretical Computer Science*, 9(2):125–210, 2013.

Schaul, T., Zhang, S., and LeCun, Y. No more pesky learning rates. In *International Conference on Machine Learning*, pp. 343–351. PMLR, 2013.

Scieur, D. and Pedregosa, F. Universal asymptotic optimality of polyak momentum. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8565–8572. PMLR, 13–18 Jul 2020. URL http://proceedings.mlr.press/v119/scieur20a.html.

Shallue, C. J., Lee, J., Antognini, J., Sohl-Dickstein, J., Frostig, R., and Dahl, G. E. Measuring the effects of data parallelism on neural network training. *Journal of Machine Learning Research*, 20:1–49, 2019.

Sidi, A., Ford, W. F., and Smith, D. A. Acceleration of convergence of vector sequences. *SIAM Journal on Numerical Analysis*, 23(1):178–196, 1986.

Smith, L. N. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pp. 464–472. IEEE, 2017.

Smith, S. L., Kindermans, P.-J., Ying, C., and Le, Q. V. Don't decay the learning rate, increase the batch size. In *International Conference on Learning Representations*, 2018.

Staib, M., Reddi, S., Kale, S., Kumar, S., and Sra, S. Escaping saddle points with adaptive gradient methods. In *International Conference on Machine Learning*, pp. 5956–5965. PMLR, 2019.

Stiefel, E. L. Kernel polynomial in linear algebra and their numerical applications. *NBS Applied Math. Ser.*, 49:1–22, 1958.

Su, W., Boyd, S. P., and Candes, E. J. A differential equation for modeling nesterov's accelerated gradient method: Theory and insights. In *NIPS*, volume 14, pp. 2510–2518, 2014.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147. PMLR, 2013.

Vishnoi, N. K. Laplacian solvers and their algorithmic applications. *Theoretical Computer Science*, 8(1-2):1–141, 2012.

Ward, R., Wu, X., and Bottou, L. Adagrad stepsizes: Sharp convergence over nonconvex landscapes. In *International Conference on Machine Learning*, pp. 6677–6686. PMLR, 2019.

Wibisono, A. and Wilson, A. C. On accelerated methods in optimization. *arXiv preprint arXiv:1509.03616*, 2015.

Wynn, P. On a device for computing the e m (s n) transformation. *Mathematical Tables and Other Aids to Computation*, pp. 91–96, 1956.

You, Y., Li, J., Reddi, S., Hseu, J., Kumar, S., Bhojanapalli, S., Song, X., Demmel, J., Keutzer, K., and Hsieh, C.-J. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*, 2019.

Young, D. On richardson's method for solving linear systems with positive definite matrices. *Journal of Mathematics and Physics*, 32(1-4):243–255, 1953.

Zhang, G., Li, L., Nado, Z., Martens, J., Sachdeva, S., Dahl, G. E., Shallue, C. J., and Grosse, R. Which algorithmic choices matter at which batch sizes? Insights from a noisy quadratic model. *arXiv preprint arXiv:1907.04164*, 2019.

A. Code snippets

Below, we provide some Python code to compute the Chebyshev step sizes $\{1/\gamma_t\}$, and the permutation σ_T that generates the fractal Chebyshev schedule $\{\eta_t\}$.

```python
import numpy as np

def cheb_steps(m, M, T):
    C, R = (M+m)/2., (M-m)/2.
    thetas = (np.arange(T)+0.5)/T * np.pi
    return 1./(C - R*np.cos(thetas))

def cheb_perm(T):
    perm = np.array([0])
    while len(perm) < T:
        perm = np.vstack([perm, 2*len(perm)-1-perm]).T.flatten()
    return perm

steps = cheb_steps(0.1, 1, 8)   # [9.20, 5.69 ... 1.01]
perm = cheb_perm(8)             # [0, 7, 3, 4, 1, 6, 2, 5]
steps[perm]                     # [9.20, 1.01 ... 1.25]
```

B. Notation and background on Chebyshev polynomials

First, we gather the notation and classic results on Chebyshev polynomials that will be useful for the proofs in Appendices C, D, and E.

B.1. Definitions

To review the notation in Section 2.3, for given values of m, M, T, we have defined the following:

- The shifted and scaled Chebyshev polynomial construction: set

$$p(\lambda) := \frac{\mathcal{T}_T(z)}{\mathcal{T}_T(\theta)} \qquad \text{where } z := \frac{M + m - 2\lambda}{M - m}, \quad \theta := \frac{M + m}{M - m} = 1 + \frac{2m}{M - m}.$$

We will keep using the auxiliary notation from above, which is useful in switching between different coordinate systems using the bijection $\lambda \leftrightarrow z$ which allows us to switch between the horizontal scales of $p(\lambda)$ and its corresponding $\mathcal{T}(z)$. This bijection maps $\lambda \in [m, M]$ to $z \in [-1, 1]$. Note that θ is the value of z corresponding to applying the above bijection to $\lambda = 0$.

- A characterization of p by its roots:

$$\gamma_t := \frac{M + m}{2} - \frac{M - m}{2} \cos \frac{(t - \frac{1}{2})\pi}{T}, \quad t = 1, \dots, T.$$

If $m = \lambda_{\min}(A)$ and $M = \lambda_{\max}(A)$, then M/m coincides with $\kappa := \lambda_{\max}(A)/\lambda_{\min}(A)$, the condition number of the matrix A. Then, we can think of $\theta = 1 + \frac{2}{\kappa - 1}$.

Next, we review some well-known facts about Chebyshev polynomials beginning with the definition.

Definition 17 (Chebyshev polynomials (Chebyshev, 1853))**.** For each $n \geq 0$, the Chebyshev polynomials of the first kind $\mathcal{T}_n(z)$ are defined by the following recurrence:

- $\mathcal{T}_0(z) = 1$.

- $\mathcal{T}_1(z) = z$.

- $\mathcal{T}_n(z) = 2z\mathcal{T}_{n-1}(z) - \mathcal{T}_{n-2}(z)$, for all $n \geq 2$.

The equivalence of the above with the alternate definition ($\mathcal{T}_n(z) = \cos(n \arccos z)$ for $|z| \leq 1$) follows by verifying the base cases and applying the cosine sum-of-angles formula.

B.2. Basic lemmas

We gather some basic lemmas used to prove the results in (Lebedev & Finogenov, 1971), which are classic.

Lemma 18 (Alternative characterizations of the Chebyshev polynomials). *The following are true for all non-negative integers n and $|z| \geq 1$:*

(i) $T_n(z) = \pm \cosh(n \operatorname{acosh}(z))$. *The sign is $+1$ when z is positive, and $(-1)^n$ when z is negative.*

(ii) We have

$$\mathcal{T}(z) = \frac{\left(z - \sqrt{z^2 - 1}\right)^n + \left(z + \sqrt{z^2 - 1}\right)^n}{2}.$$

Proof. (i) follows from recursively applying the identities $\cosh(a+b) = \sinh(a)\sinh(b) + \cosh(a)\cosh(b)$ and $\sinh^2(a) = \cosh^2(a) - 1$. (ii) follows from (i), performing the hyperbolic substitution $z = \cosh(a)$, noticing that all odd-powered $\sinh(a)$ terms cancel, and verifying the base cases and recurrence relation. $\square$

Combining the $\cos$ and $\cosh$ characterizations of $\mathcal{T}_n(z)$, we obtain the composition property: for all integers $k, n \geq 0$ and all $z \in \mathbb{R}$,

$$\mathcal{T}_{kn}(z) = \mathcal{T}_k(\mathcal{T}_n(z)).$$

The half-angle cosine formulas will be useful: for any $\alpha, z \in \mathbb{R}$, and positive even n,

$$\cos(\theta) = 2\cos^2(\theta/2) - 1 = 1 - 2\sin^2(\theta/2); \qquad \mathcal{T}_n(z) = 2\mathcal{T}_{n/2}^2(z) - 1.$$

The third statement is true for all z, since it is the composition property with $k = 2$.

The key reason why we are interested in the Chebyshev polynomials is their *extremal* property: outside the range of their roots, they expand faster than any other polynomial of the same degree. There are various ways to formalize this. We will only need the following:

Lemma 19 (Expansion lower bounds). *For all $\delta \geq 0$, each $\mathcal{T}_n$ satisfies the following:*

(i) $\mathcal{T}_n(1+\delta) = \frac{(1+\delta+\sqrt{2\delta+\delta^2})^{2n}+1}{2(1+\delta+\sqrt{2\delta+\delta^2})^n} \geq \frac{(1+\sqrt{2\delta})^n}{2}.$

(ii) $\mathcal{T}_n(1+\delta) \geq 1 + n^2\delta.$

Proof. To prove (i) note that, using Lemma 18(ii), we have that

$$\mathcal{T}_n(1+\delta) = \frac{(1+\delta+\sqrt{2\delta+\delta^2})^n + (1+\delta-\sqrt{2\delta+\delta^2})^n}{2} = \frac{(1+\delta+\sqrt{2\delta+\delta^2})^{2n}+1}{2(1+\delta+\sqrt{2\delta+\delta^2})^n},$$

where the last equality follows by noticing that $(1+\delta-\sqrt{2\delta+\delta^2})^{-1} = (1+\delta+\sqrt{2\delta+\delta^2})$. The inequality in (i) is concluded by noticing that

$$\frac{(1+\delta+\sqrt{2\delta+\delta^2})^n + (1+\delta-\sqrt{2\delta+\delta^2})^n}{2} \geq \frac{(1+\sqrt{2\delta})^n}{2},$$

by dropping the positive terms. To conclude (ii), we perform a series expansion upto degree 2 to get,

$$\frac{(1+\delta+\sqrt{2\delta+\delta^2})^n + (1+\delta-\sqrt{2\delta+\delta^2})^n}{2} \geq \frac{2 + 2n\delta + \frac{n(n-1)}{2}(4\delta + 4\delta^2)}{2}$$

$$= 1 + n^2\delta.$$

$\square$

Classic convergence rate of Chebyshev iteration. Though Theorem 3 is classic, the exact statement of the convergence rate has several variants. For sake of completeness, we give a quick proof of the convergence rate of Chebyshev iteration implied by the exact formula in Lemma 19 (i):

Theorem 3. *Choose spectral estimates* $m \leq M$ *such that* $0 < m \leq \lambda_{\min} \leq \lambda_{\max} \leq M$. *Then, setting* $\{\eta_t\}$ *to be any permutation of* $\{1/\gamma_t\}$, *the final iterate of gradient descent* x_{out} *satisfies the following:*

$$\|x_{\text{out}} - x^*\| \leq \frac{2\rho^T}{1+\rho^{2T}} \|x_1 - x^*\| \leq e^{-\Omega(T)/\sqrt{\widehat{\kappa}}} \|x_1 - x^*\|,$$

where $\rho := \frac{\sqrt{M}-\sqrt{m}}{\sqrt{M}+\sqrt{m}} \leq 1 - \Omega\left(\frac{1}{\sqrt{\widehat{\kappa}}}\right)$.

Proof. First, assume $m < M$. Then, we have

$$\|x_{\text{out}} - x^*\| \leq \|p\|_{[\lambda_{\min}, \lambda_{\max}]} \cdot \|x_{\text{out}} - x^*\| \leq \|p\|_{[m,M]} \cdot \|x_{\text{out}} - x^*\|.$$

So, we need to bound $\|p\|_{[m,M]}$. Setting $\delta = \theta - 1 = \frac{2m}{M-m}$, notice that $1 + \delta + \sqrt{2\delta + \delta^2} = 1/\rho$. Then, using Lemma 19 (i), we have

$$\|p\|_{[m,M]} = \max_{|z| \leq 1} \frac{\mathcal{T}_T(z)}{\mathcal{T}_T(\theta)} = \frac{1}{\mathcal{T}_T(\theta)} = \frac{2\rho^{-T}}{1+\rho^{-2T}} = \frac{2\rho^T}{1+\rho^{2T}},$$

as desired. When $m = M$, the inequality is trivially true because ρ and $\|p\|_{[m,M]}$ are both 0.

$\square$

C. Theorems and proofs from (Lebedev & Finogenov, 1971)

In the hope of bridging old algorithmic ideas from numerical methods with modern optimization for machine learning, we present a self-contained exposition of the results and proofs from (Lebedev & Finogenov, 1971) used in this paper. This is far from an exact translation from the original Russian-language manuscript, whose exposition is somewhat terse. We provide some more intuitive proofs, fix some small (inconsequential) typos, change some notation to match this paper, isolate lemmas which are useful for proving our other results, and omit some weaker and irrelevant results.

C.1. Skewed Chebyshev polynomials

First, we show an equivalent divide-and-conquer root-partitioning construction of the fractal permutation σ_T. To review, this construction defines $\sigma_1 := [1]$, and for each $T \geq 1$ a power of 2, uses the recurrence

$$\sigma_{2T} := \text{interlace}(\sigma_T, 2T + 1 - \sigma_T),$$

where

$$\text{interlace}([a_1 \ldots a_n], [b_1 \ldots b_n]) := [a_1 \ b_1 \ a_2 \ b_2 \ldots a_n \ b_n].$$

Some examples are below:

$$\sigma_2 = [1 \ 2],$$
$$\sigma_4 = [1 \ 4 \ 2 \ 3],$$
$$\sigma_8 = [1 \ 8 \ 4 \ 5 \ 2 \ 7 \ 3 \ 6],$$
$$\sigma_{16} = [1 \ 16 \ 8 \ 9 \ 4 \ 13 \ 5 \ 12 \ 2 \ 15 \ 7 \ 10 \ 3 \ 14 \ 6 \ 11].$$

Let us formalize the sense in which σ contains internal copies of Chebyshev iteration. For positive integers n and $\alpha \in (0, \pi)$, let us define the *skewed Chebyshev polynomials*

$$\mathcal{P}_{n,\alpha}(z) := \frac{\mathcal{T}_n(z) - \cos(\alpha)}{\mathcal{T}_n(\theta) - \cos(\alpha)},$$

noting that $\mathcal{P}_{T,\frac{\pi}{2}}(z) = p(\lambda)$. If $\alpha \in (0, \frac{\pi}{2}]$, then call $\mathcal{P}_{n,\alpha}$ good and if $\alpha \in [\frac{\pi}{2}, \pi)$, then call $\mathcal{P}_{n,\alpha}$ bad. We use the colours blue and red to highlight good and bad polynomials for clarity in our proofs. Note that $p(\lambda)$ is both good and bad. Next, we note some additional facts:

Lemma 20 (Properties of skewed Chebyshev polynomials). *The following are true for all $n \in \mathbb{N}$ and $\alpha \in (0, \pi)$:*

(i) $\mathcal{P}_{n,\alpha}(z)$ *has n real roots.*

(ii) If $\mathcal{P}_{n,\alpha}$ is good, then

$$\|\mathcal{P}_{n,\alpha}\|_{L_\infty([-1,1])} \leq \frac{2}{1 + \mathcal{T}_n(\theta)}.$$

(iii) If $\mathcal{P}_{n,\alpha}$ is bad, then

$$\|\mathcal{P}_{n,\alpha}\|_{L_\infty([-1,1])} \leq \frac{2}{n^2(\theta - 1)}.$$

(iv) If n is even, then for all $-1 \leq z \leq 1$,

$$\mathcal{P}_{n,\alpha}(z) = \mathcal{P}_{\frac{n}{2},\frac{\alpha}{2}}(z) \cdot \mathcal{P}_{\frac{n}{2},\pi-\frac{\alpha}{2}}(z).$$

Proof. (i) follows from the fact that $\mathcal{T}_n(z_i) = (-1)^i$ at $z_i = \arccos(i\pi/n), i = 0, \ldots, n$.

(ii) follows from the fact that $|\mathcal{T}(z)| \leq 1, \mathcal{T}(\theta) > 1$, and $\cos(\alpha) \leq 0$, so that for $u = 1 + \cos(\alpha) \geq 0$,

$$\|\mathcal{P}_{n,\alpha}\|_{L_\infty([-1,1])} \leq \frac{1 - \cos(\alpha)}{\mathcal{T}_n(\theta) - \cos(\alpha)} \leq \frac{1 - \cos(\alpha) + u}{\mathcal{T}_n(\theta) - \cos(\alpha) + u} = \frac{2}{1 + \mathcal{T}_n(\theta)},$$

where the last inequality uses the mediant inequality.

(iii) is a weaker bound than the above, because we can't use the mediant inequality, as $-\cos(\alpha)$ is negative. Using part (ii) of Lemma 19 and $\cos(\alpha) \leq 1$, we get

$$\|\mathcal{P}_{n,\alpha}\|_{L_\infty([-1,1])} \leq \frac{2}{\mathcal{T}_n(\theta) - \cos(\alpha)} \leq \frac{2}{(1 + n^2(\theta - 1)) - 1}.$$

(iv) follows from half-angle formulas and factorizing differences of squares:

$$\mathcal{P}_{n,\alpha}(z) = \frac{\mathcal{T}_n(z) - \cos(\alpha)}{\mathcal{T}_n(\theta) - \cos(\alpha)} = \frac{(2\mathcal{T}_{n/2}^2(z) - 1) - (2\cos^2(\frac{\alpha}{2}) - 1)}{(2\mathcal{T}_{n/2}^2(\theta) - 1) - (2\cos^2(\frac{\alpha}{2}) - 1)} = \frac{\mathcal{T}_{n/2}^2(z) - \cos^2(\frac{\alpha}{2})}{\mathcal{T}_{n/2}^2(\theta) - \cos^2(\frac{\alpha}{2})}$$

$$= \frac{\mathcal{T}_{n/2}(z) - \cos(\frac{\alpha}{2})}{\mathcal{T}_{n/2}(\theta) - \cos(\frac{\alpha}{2})} \cdot \frac{\mathcal{T}_{n/2}(z) + \cos(\frac{\alpha}{2})}{\mathcal{T}_{n/2}(\theta) + \cos(\frac{\alpha}{2})} = \mathcal{P}_{\frac{n}{2},\frac{\alpha}{2}}(z) \cdot \mathcal{P}_{\frac{n}{2},\pi-\frac{\alpha}{2}}(z),$$

as claimed. $\square$

C.2. The fractal schedule splits skewed Chebyshev polynomials

In this section, we connect the skewed polynomials $\mathcal{P}_{n,\alpha}$ to the construction of the fractal permutation σ_T, obtained via recursive binary splitting. This construction will provide the basis for all the proofs regarding the fractal schedule. The starting point for the construction is Lemma 20, which shows that when $T \geq 2$ is a power of 2,

$$p(\lambda) = \mathcal{P}_{T,\frac{\pi}{2}}(z) = \mathcal{P}_{\frac{T}{2},\frac{\pi}{4}}(z) \cdot \mathcal{P}_{\frac{T}{2},\frac{3\pi}{4}}(z).$$

The above splitting procedure can be recursively repeated (since T is a power of 2) on the pieces produced till we reach degree 1 polynomials of the form $\mathcal{P}_{1,\alpha}$ for some α. Such splitting can easily be visualized via the construction of a complete binary tree of depth $\log_2(T)$ (see Figure 7), by associating to every node a polynomial of the form $\mathcal{P}_{n,\alpha}$ and setting its left child to be $\mathcal{P}_{n/2,\alpha/2}$ and right child to be $\mathcal{P}_{n/2,\pi-\alpha/2}$. Note that every non-leaf node is a product of its children by Lemma 20. The root node corresponds to the polynomial $p(\lambda)$ and the leaf nodes correspond to one degree polynomials which can be equivalently identified by its root, which are by construction the roots of polynomial $p(\lambda)$.

The key fact regarding this construction is the following.

Fact 21. *The fractal schedule corresponds to the ordering of the roots as generated by a pre-order traversal of the tree.*

To see this note that every time a split is made in the tree, a constraint on the pre-order traversal is placed, i.e. the roots of the left child polynomial $\mathcal{P}_{n/2,\alpha/2}$ precede that of the right child polynomial $\mathcal{P}_{n/2,\pi-\alpha/2}$. It can be easily verified that the procedure for generating σ_T produces an ordering of the roots γ_t satisfying all of these constraints; the corresponding learning rate schedule η_t is by definition the fractal Chebyshev schedule for each T, a power of 2.

Using the above, it can be seen that every node in the tree also corresponds to a particular infix polynomial $p_{s:t}(x)$ which includes all the roots corresponding to all the leaves underneath the node.
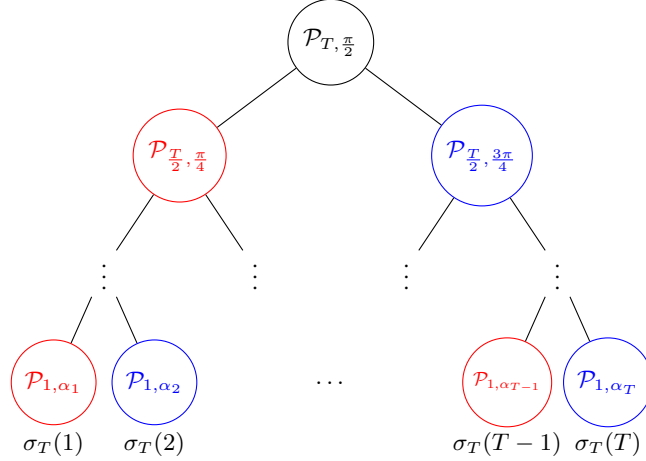

Figure 7: The binary tree construction for the decomposition of skewed Chebyshev polynomials. Each non-leaf node corresponds to the product of the polynomials corresponding to its children. The root corresponds to the entire polynomial $p_{1:T}(\lambda)$. Leaf nodes correspond to 1 degree polynomials and the pre-order traversal on the leaves induces the ordering given by σ_T.

C.3. Tree partitions and tree exchanges

In this section we collect some observations regarding the tree construction, which are essential to the proofs for the bounds on the substring polynomials.

Firstly note that in the binary tree, a node/polynomial is good (bad) if and only if it is the left (right) child of its parent. We begin by analyzing the special case of suffix polynomials $p_{s:T}$ and prefix polynomials $p_{1:s}$ respectively.

Suffix polynomials: The following key observation follows from the tree construction.
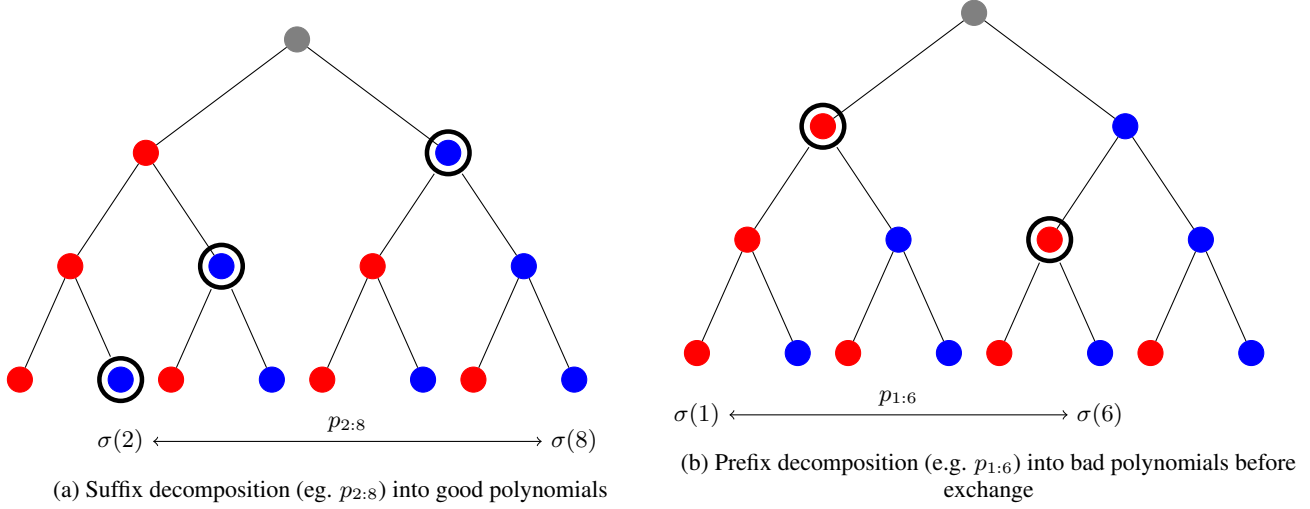
Fact 22. *Every suffix polynomial $p_{s:T}$ (for any s) can be written as a product of good polynomials*

To see the above, consider the binary expansion of the number $T + 1 - s$, $\text{bits}(T + 1 - s) = \{s_1, s_2 \ldots s_k\}$ defined by the unique decomposition, $T + 1 - s = 2^{s_1} + 2^{s_2} + \ldots + 2^{s_k}$ such that $s_1 > s_2 > \ldots s_k \geq 0$. We now perform the following iterative decomposition of the polynomial $p_{s:T}$,

$$
\begin{aligned}
p_{s:T} &= p_{s:T_1} \cdot p_{T_1+1:T} && \text{where } T_1 := T - 2^{s_1}, \\
&= p_{s:T_2} \cdot p_{T_2:T_1} \cdot p_{T_1+1:T} && \text{where } T_2 := T_1 - 2^{s_2}, \\
&\cdots,
\end{aligned}
$$

until we reach $s : T_k$, which is the empty interval. It can be seen that every intermediate polynomial $p_{T_i+1:T_{i-1}}$ produced is a good polynomial because each one is the rightmost node at level s_i (i.e. with distance $\log_2 T - s_i$ from the root node), restricted to the subtree rooted at the lowest common ancestor of roots s through T_{i-1} (setting $T_0 := T$). An example of the above decomposition is highlighted in Figure 8a. Combining with statement (ii) in Lemma 20, we get

$$
\|p_{s:T}\|_{L_\infty([-1,1])} \leq \prod_{i=1}^{k} \frac{2}{1 + \mathcal{T}_{2^{s_i}}(\theta)} \tag{5}
$$

(a) Suffix decomposition (eg. $p_{2:8}$) into good polynomials

(b) Prefix decomposition (e.g. $p_{1:6}$) into bad polynomials before exchange

Prefix polynomials: The prefix polynomials $p_{1:s}$ are more challenging to analyze, as the immediate approach of reversing the above construction gives a decomposition into bad polynomials only.

To this end, consider the binary expansion of $s = 2^{s_1} + 2^{s_2} + \ldots + 2^{s_k}$ such that $s_1 > s_2 > \ldots s_1 \geq 0$. We decompose $p_{1:s}$ into products in the following manner: starting with $\{1, \ldots, s-1\}$, we iteratively partition

$$
\begin{aligned}
p_{1:s} &= p_{1:T_1} \cdot p_{T_1+1:s} & \text{where } T_1 &:= 2^{s_1}, \\
&= p_{1:T_1} \cdot p_{T_1+1:T_2} \cdot p_{T_2+1:s} & \text{where } T_2 &:= T_1 + 2^{s_2}, \\
&\ldots,
\end{aligned}
$$

until we reach $T_k + 1 : s$, which is the empty interval. Note that this partition results in all bad polynomials. An example of the above decomposition is highlighted in Figure 8b. We can in fact exactly characterize these polynomials. Define the angle recurrence $\alpha_1 = \frac{2^{s_1}}{T} \cdot \frac{\pi}{2}$ and $\alpha_{i+1} = \frac{\pi - \alpha_i}{2^{s_i - s_{i+1}}}$. It can be seen that

$$
p_{1:s} = \prod_{i=1}^{k} \mathcal{P}_{2^{s_i}, \alpha_i}. \tag{6}
$$

To get a tight bound for the norms of these polynomials, we require another innovation from (Lebedev & Finogenov, 1971). This innovation can be captured as a *tree exchange property* that allows us to switch two bad polynomials for a good and bad polynomial. Starting with a partition of the roots of the polynomial we want to analyze, applying this tree switching repeatedly allows us to convert a partition with multiple bad polynomials into a set containing only one bad polynomial (and the rest good).

To further elucidate this tree exchange trick, let us establish some notation. We will be manipulating upper bounds for norms of $\mathcal{P}_{n,a}$, the product of which will serve as an upper bound for a prefix or suffix polynomial. Let

$$
\mathcal{B}_{n,\alpha} := \|\mathcal{P}_{n,\alpha}\|_{L_\infty([-1,1])} = \max_{z \in [-1,1]} \left| \frac{\mathcal{T}_n(z) - \cos(\alpha)}{\mathcal{T}_n(\theta) - \cos(\alpha)} \right|.
$$

Note that the denominator of this fraction is positive independent of z. In the numerator, the maximum is achieved when $\mathcal{T}_n(z)$ is either $+1$ or -1, depending on the sign of $\cos(\alpha)$. When $\mathcal{P}_{n,\alpha}$ is good, $\cos(\alpha) \leq 0$, thus we have

$$
\mathcal{B}_{n,\alpha} = \frac{1 - \cos(\alpha)}{\mathcal{T}_n(\theta) - \cos(\alpha)} = \frac{2\sin^2\left(\frac{\alpha}{2}\right)}{\mathcal{T}_n(\theta) - \cos(\alpha)}.
$$

When $\mathcal{P}_{n,\alpha}$ is bad, $\cos(\alpha) \geq 0$, thus we have

$$
\mathcal{B}_{n,\alpha} = \frac{1 + \cos(\alpha)}{\mathcal{T}_n(\theta) - \cos(\alpha)} = \frac{2\cos^2\left(\frac{\alpha}{2}\right)}{\mathcal{T}_n(\theta) - \cos(\alpha)}.
$$

With this notation, using (6), we get that,

$$\|p_{1:s}\|_{L_\infty([-1,1])} \le \prod_{i=1}^{k} \mathcal{B}_{2^{s_i}, \alpha_i}.$$

Now, we introduce the key tool which will allow us to handle products of bad polynomials.

Lemma 23 (Tree Exchange Property). *For any $0 < \alpha < \frac{\pi}{2}$, and integers $n \ge 2$, $r \ge 1$, we have*

$$\mathcal{B}_{nr, \alpha} \cdot \mathcal{B}_{r, \frac{\pi - \alpha}{n}} \le \mathcal{B}_{nr, \pi - \alpha} \cdot \mathcal{B}_{r, \frac{\alpha}{n}}.$$

If we view the arguments as indexing the corresponding subtrees in our construction, then the right hand side can be viewed as exchanging a (bad) $\mathcal{P}_{nr, \alpha}$ with its (good) sibling $\mathcal{P}_{nr, \pi - \alpha}$ at the cost of exchanging $\mathcal{P}_{r, \frac{\pi - \alpha}{n}}$ with the leftmost degree-r polynomial $\mathcal{P}_{r, \frac{\alpha}{n}}$ under $\mathcal{P}_{nr, \alpha}$ (see Figure 9).

Proof. Using the derived bounds on $\mathcal{B}$, our claim reduces to proving the inequality

$$\frac{\cos^2\left(\frac{\alpha}{2}\right)}{\mathcal{T}_{nr}(\theta) - \cos(\alpha)} \cdot \frac{\cos^2\left(\frac{\pi - \alpha}{2n}\right)}{\mathcal{T}_r(\theta) - \cos\left(\frac{\pi - \alpha}{n}\right)} \le \frac{\cos^2\left(\frac{\alpha}{2}\right)}{\mathcal{T}_{nr}(\theta) + \cos(\alpha)} \cdot \frac{\cos^2\left(\frac{\alpha}{2n}\right)}{\mathcal{T}_r(\theta) - \cos\left(\frac{\alpha}{n}\right)},$$

which is equivalent to the following (since all terms are positive).

$$\frac{\mathcal{T}_{nr}(\theta) - 1 + 2\sin^2\left(\frac{\alpha}{2}\right)}{\cos^2\left(\frac{\alpha}{2}\right)} \cdot \frac{\mathcal{T}_r(\theta) - 1 + 2\sin^2\left(\frac{\pi - \alpha}{2n}\right)}{\cos^2\left(\frac{\pi - \alpha}{2n}\right)} \ge \frac{\mathcal{T}_{nr}(\theta) - 1 + 2\cos^2\left(\frac{\alpha}{2}\right)}{\cos^2\left(\frac{\alpha}{2}\right)} \cdot \frac{\mathcal{T}_r(\theta) - 1 + 2\sin^2\left(\frac{\alpha}{2n}\right)}{\cos^2\left(\frac{\alpha}{2n}\right)}.$$

For ease of exposition, let us set $\Delta_i(\theta) = \mathcal{T}_i(\theta) - 1$. Note that by definition of θ, we have $\Delta_i(\theta) \ge 0$. Observe that the second inequality holds if the following three inequalities are true,

$$\frac{\Delta_{nr}(\theta)\Delta_r(\theta)}{\cos^2\left(\frac{\alpha}{2}\right)} \left(\frac{1}{\cos^2\left(\frac{\pi - \alpha}{2n}\right)} - \frac{1}{\cos^2\left(\frac{\alpha}{2n}\right)} \right) \ge 0 \tag{7}$$

$$\tan^2\left(\frac{\alpha}{2}\right)\tan^2\left(\frac{\pi - \alpha}{2n}\right) - \tan^2\left(\frac{\alpha}{2n}\right) \ge 0 \tag{8}$$

$$\Delta_{nr}(\theta) \left(\frac{\sin^2\left(\frac{\pi - \alpha}{2n}\right)}{\cos^2\left(\frac{\pi - \alpha}{2n}\right)\cos^2\left(\frac{\alpha}{2}\right)} - \frac{\sin^2\left(\frac{\alpha}{2n}\right)}{\cos^2\left(\frac{\alpha}{2n}\right)\cos^2\left(\frac{\alpha}{2}\right)} \right) + \Delta_r(\theta) \left(\frac{\sin^2\left(\frac{\alpha}{2}\right)}{\cos^2\left(\frac{\alpha}{2}\right)\cos^2\left(\frac{\pi - \alpha}{2n}\right)} - \frac{1}{\cos^2\left(\frac{\alpha}{2n}\right)} \right) \ge 0 \tag{9}$$

Note that Equation (7) follows from the fact that $\cos^2\left(\frac{\alpha}{2n}\right) \ge \cos^2\left(\frac{\pi - \alpha}{2n}\right) > 1/2$ since $\frac{\pi}{4} \ge \frac{\pi - \alpha}{2n} \ge \frac{\alpha}{2n} \ge 0$.

To prove Equation (8), observe that it is equivalent to proving

$$\sin\left(\frac{\alpha}{2}\right)\sin\left(\frac{\pi - \alpha}{2n}\right)\cos\left(\frac{\alpha}{2n}\right) - \cos\left(\frac{\alpha}{2}\right)\cos\left(\frac{\pi - \alpha}{2n}\right)\sin\left(\frac{\alpha}{2n}\right) \ge 0$$

Further simplifying the left hand side, we get

$$2\left(\sin\left(\frac{\alpha}{2}\right)\sin\left(\frac{\pi - \alpha}{2n}\right)\cos\left(\frac{\alpha}{2n}\right) - \cos\left(\frac{\alpha}{2}\right)\cos\left(\frac{\pi - \alpha}{2n}\right)\sin\left(\frac{\alpha}{2n}\right) \right)$$

$$= \sin\left(\frac{\alpha}{2}\right)\left(\sin\left(\frac{\pi}{2n}\right) + \sin\left(\frac{\pi - 2\alpha}{2n}\right) \right) - \cos\left(\frac{\alpha}{2}\right)\left(\sin\left(\frac{\pi}{2n}\right) - \sin\left(\frac{\pi - 2\alpha}{2n}\right) \right)$$

$$= \sin\left(\frac{\pi}{2n}\right)\left(\sin\left(\frac{\alpha}{2}\right) - \cos\left(\frac{\alpha}{2}\right) \right) + \sin\left(\frac{\pi - 2\alpha}{2n}\right)\left(\sin\left(\frac{\alpha}{2}\right) + \cos\left(\frac{\alpha}{2}\right) \right)$$

$$= \sqrt{2}\left(\sin\left(\frac{\pi - 2\alpha}{2n}\right)\cos\left(\frac{\pi - 2\alpha}{4}\right) - \sin\left(\frac{\pi}{2n}\right)\sin\left(\frac{\pi - 2\alpha}{4}\right) \right).$$

Observe that $\frac{\sin(x)}{x}$ is a decreasing function for $0 < x \le \frac{\pi}{4}$, therefore we have $\frac{\sin\left(\frac{\pi-2\alpha}{2n}\right)}{\frac{\pi-2\alpha}{2n}} \ge \frac{\sin\left(\frac{\pi}{2n}\right)}{\frac{\pi}{2n}}$. Substituting this back, we get

$$\sqrt{2}\left(\sin\left(\frac{\pi-2\alpha}{2n}\right)\cos\left(\frac{\pi-2\alpha}{4}\right) - \sin\left(\frac{\pi}{2n}\right)\sin\left(\frac{\pi-2\alpha}{4}\right)\right)$$

$$\ge \sqrt{2}\sin\left(\frac{\pi}{2n}\right)\left(\frac{\pi-2\alpha}{\pi}\cos\left(\frac{\pi-2\alpha}{4}\right) - \sin\left(\frac{\pi-2\alpha}{4}\right)\right) \ge 0.$$

Here the last inequality follows from observing that $\tan(x) \le \frac{4x}{\pi}$ for $0 \le x \le \frac{\pi}{4}$. This proves Equation (8). Now it remains to prove Equation (9). Further simplifying the equation, it is equivalent to,

$$\frac{\Delta_{nr}(\theta)}{\Delta_r(\theta)}\left(\sin^2\left(\frac{\pi-\alpha}{2n}\right)\cos^2\left(\frac{\alpha}{2n}\right) - \sin^2\left(\frac{\alpha}{2n}\right)\cos^2\left(\frac{\pi-\alpha}{2n}\right)\right) + \sin^2\left(\frac{\alpha}{2}\right)\cos^2\left(\frac{\alpha}{2n}\right) - \cos^2\left(\frac{\alpha}{2}\right)\cos^2\left(\frac{\pi-\alpha}{2n}\right) \ge 0$$

Let us prove this inequality. We have

$$\frac{\Delta_{nr}(\theta)}{\Delta_r(\theta)}\left(\sin^2\left(\frac{\pi-\alpha}{2n}\right)\cos^2\left(\frac{\alpha}{2n}\right) - \sin^2\left(\frac{\alpha}{2n}\right)\cos^2\left(\frac{\pi-\alpha}{2n}\right)\right) + \sin^2\left(\frac{\alpha}{2}\right)\cos^2\left(\frac{\alpha}{2n}\right) - \cos^2\left(\frac{\alpha}{2}\right)\cos^2\left(\frac{\pi-\alpha}{2n}\right)$$

$$= \frac{\Delta_{nr}(\theta)}{\Delta_r(\theta)}\left(\sin^2\left(\frac{\pi-\alpha}{2n}\right) - \sin^2\left(\frac{\alpha}{2n}\right)\right) + \cos^2\left(\frac{\alpha}{2}\right)\left(\cos^2\left(\frac{\alpha}{2n}\right) - \cos^2\left(\frac{\pi-\alpha}{2n}\right)\right) + \left(\sin^2\left(\frac{\alpha}{2}\right) - \cos^2\left(\frac{\alpha}{2}\right)\right)\cos^2\left(\frac{\alpha}{2n}\right)$$

$$= \frac{\Delta_{nr}(\theta)}{\Delta_r(\theta)}\sin\left(\frac{\pi-2\alpha}{2n}\right)\sin\left(\frac{\pi}{2n}\right) + \cos^2\left(\frac{\alpha}{2}\right)\left(\cos^2\left(\frac{\alpha}{2n}\right) - \cos^2\left(\frac{\pi-\alpha}{2n}\right)\right) - \cos(\alpha)\cos^2\left(\frac{\alpha}{2n}\right)$$

$$\ge \frac{\Delta_{nr}(\theta)}{\Delta_r(\theta)}\sin\left(\frac{\pi-2\alpha}{2n}\right)\sin\left(\frac{\pi}{2n}\right) - \sin\left(\frac{\pi-2\alpha}{2}\right).$$

Here the last inequality follows from the fact that $\cos\left(\frac{\alpha}{2n}\right) \ge \cos\left(\frac{\pi-\alpha}{2n}\right)$ since $\alpha \le \frac{\pi}{2}$ and $\cos(\alpha)\cos^2\left(\frac{\alpha}{2n}\right) \le \cos(\alpha) = \sin\left(\frac{\pi-2\alpha}{2}\right)$.

By the composition property and part (ii) of Lemma 19, we have

$$\frac{\Delta_{nr}(\theta)}{\Delta_r(\theta)} = \frac{\mathcal{T}_n(\mathcal{T}_r(\theta)) - 1}{\mathcal{T}_r(\theta) - 1} \le \frac{1 + n^2(\mathcal{T}_r(\theta) - 1) - 1}{\mathcal{T}_r(\theta) - 1} \le n^2.$$

We also know from before that $\frac{\sin(x)}{x}$ is decreasing in $0 < x \le \frac{\pi}{4}$, therefore, we have $\frac{\sin\left(\frac{\pi}{2n}\right)}{\frac{\pi}{2n}} \ge \frac{\sin\left(\frac{\pi}{4}\right)}{\frac{\pi}{4}}$ and $\frac{\sin\left(\frac{\pi-2\alpha}{2n}\right)}{\frac{\pi-2\alpha}{2n}} \ge \frac{\sin\left(\frac{\pi-2\alpha}{2}\right)}{\frac{\pi-2\alpha}{2}}$. Combining these and substituting back, we get

$$\frac{\Delta_{nr}(\theta)}{\Delta_r(\theta)}\sin\left(\frac{\pi-2\alpha}{2n}\right)\sin\left(\frac{\pi}{2n}\right) - \sin\left(\frac{\pi-2\alpha}{2}\right) \ge n^2 \cdot \frac{\sin\left(\frac{\pi-2\alpha}{2}\right)}{n} \cdot \frac{2\sin\left(\frac{\pi}{4}\right)}{n} - \sin\left(\frac{\pi-2\alpha}{2}\right)$$

$$= (\sqrt{2}-1)\sin\left(\frac{\pi-2\alpha}{2}\right) \ge 0.$$

This completes the proof of the tree exchange lemma. $\qquad\square$

C.4. Completing the main theorems

Theorem 8 (Prefix and suffix bounds). *For a fractal Chebyshev schedule with m, M, T, and all $1 \le s, t \le T$:*

(i) $\|p_{1:t}\|_{L_\infty([m,M])} \le \left(\frac{M}{m} - 1\right)\prod_{j\in\text{bits}'(t)}\frac{2}{1+\mathcal{T}_{2^j}(\theta)};$

(ii) $\|p_{s:T}\|_{L_\infty([m,M])} \le \prod_{j\in\text{bits}(T+1-s)}\frac{2}{1+\mathcal{T}_{2^j}(\theta)},$

where $\text{bits}(n)$ *denotes the indices in the binary expansion of n, formally defined as the unique sequence $\{j_1, j_2 \dots\}$ with $j_1 > j_2 > j_3 \dots$ such that $n = \sum_{j_k\in\text{bits}(n)} 2^{j_k}$. Further we define $\text{bits}'(n) := \text{bits}(n) \setminus j_1$. For example, when $n = 6 = (110)_2$, $\text{bits}(n) = \{2, 1\}$, and $\text{bits}'(n) = \{1\}$.*
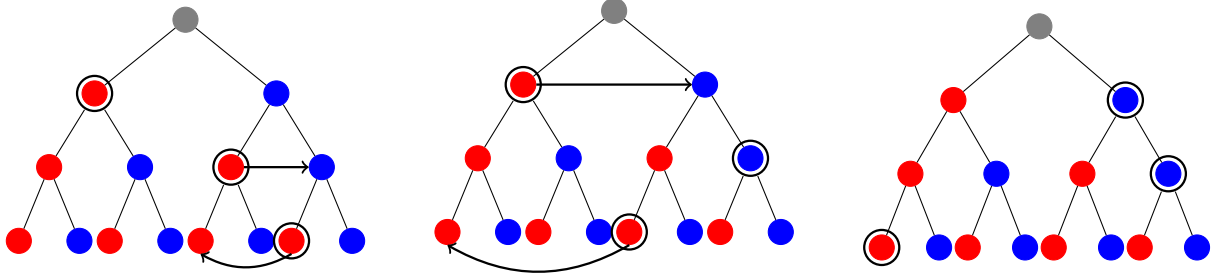
Figure 9: An example of successive exchanges to fix a prefix polynomial ($p_{1:7}$). In every exchange, the product of two bad nodes are exchanged with a product of a good node and a bad node. Eventually, one is left with exactly one bad node and the remaining good nodes.

Starting with a prefix decomposition and repeatedly applying Lemma 23:

$$\|p_{1:s}\|_{L_\infty([-1,1])} = \max_{-1 \leq z \leq 1} \left| \prod_{i=1}^{k} \mathcal{P}_{2^{s_i},\alpha_i}(z) \right| \tag{Using (6)}$$

$$\leq \prod_{i=1}^{k} \max_{-1 \leq z \leq 1} |\mathcal{P}_{2^{s_i},\alpha_i}(z)|$$

$$= \prod_{i=1}^{k} \mathcal{B}_{2^{s_i},\alpha_i}$$

$$= \left(\prod_{i=1}^{k-2} \mathcal{B}_{2^{s_i},\alpha_i} \right) \cdot \mathcal{B}_{2^{s_{k-1}},\alpha_{k-1}} \cdot \mathcal{B}_{2^{s_k}, \frac{\pi - \alpha_{k-1}}{2^{s_{k-1} - s_k}}}$$

$$\leq \mathcal{B}_{2^{s_{k-1}},\pi - \alpha_{k-1}} \cdot \left(\prod_{i=1}^{k-2} \mathcal{B}_{2^{s_i},\alpha_{k-1}} \right) \cdot \mathcal{B}_{2^{s_k}, \frac{\alpha_{k-1}}{2^{s_{k-1} - s_k}}} \tag{using Lemma 23}$$

$$= \mathcal{B}_{2^{s_{k-1}},\pi - \alpha_{k-1}} \cdot \left(\prod_{i=1}^{k-2} \mathcal{B}_{2^{s_i},\alpha_{k-1}} \right) \cdot \mathcal{B}_{2^{s_k}, \frac{\pi - \alpha_{k-2}}{2^{s_{k-2} - s_k}}} \tag{using recurrence angle relation}$$

$$= \mathcal{B}_{2^{s_{k-1}},\pi - \alpha_{k-1}} \cdot \left(\prod_{i=1}^{k-3} \mathcal{B}_{2^{s_i},\alpha_{k-1}} \right) \cdot \mathcal{B}_{2^{s_{k-2}},\alpha_{k-2}} \cdot \mathcal{B}_{2^{s_k}, \frac{\pi - \alpha_{k-2}}{2^{s_{k-2} - s_k}}}$$

$$= \mathcal{B}_{2^{s_{k-1}},\pi - \alpha_{k-1}} \cdot \mathcal{B}_{2^{s_{k-2}},\pi - \alpha_{k-2}} \left(\prod_{i=1}^{k-3} \mathcal{B}_{2^{s_i},\alpha_{k-1}} \right) \cdot \mathcal{B}_{2^{s_k}, \frac{\alpha_{k-2}}{2^{s_{k-2} - s_k}}} \tag{using Lemma 23}$$

$$\vdots \tag{repeating this switching argument iteratively}$$

$$\leq \left(\prod_{i=1}^{k-1} \mathcal{B}_{2^{s_i},\pi - \alpha_i} \right) \cdot \mathcal{B}_{2^{s_k}, \frac{\alpha_1}{2^{s_1 - s_k}}}$$

$$= \left(\prod_{i=1}^{k-1} \mathcal{B}_{2^{s_i},\pi - \alpha_i} \right) \cdot \mathcal{B}_{2^{s_k}, \frac{2^{s_k}\pi}{2T}} \cdot$$

Note that this repeated exchange leads to only one bad polynomial and rest all good polynomials. Using (ii) and (iii) of Lemma 20, we get

$$\|p_{1:s}\|_{L_\infty([-1,1])} \leq \prod_{i=1}^{k-1} \frac{2}{1 + \mathcal{T}_{2^{s_i}}(\theta)} \cdot \frac{2}{4^{s_k}(\theta - 1)}$$

$$\leq \frac{\frac{M}{m} - 1}{4^{s_k}} \cdot \prod_{i=1}^{k-1} \frac{2}{1 + \mathcal{T}_{2^{s_i}}(\theta)} \qquad \text{(using the definition of } \theta\text{)}$$

D. Proofs for Section 3

D.1. Basic facts about the schedule

We provide full statements and proofs of Proposition 6:

Proposition 6. *For all m, M, T, the fractal Chebyshev step sizes γ_t^{-1} satisfy the following:*

(i) $\frac{1}{M} < \gamma_t^{-1} < \frac{1}{m}$.

(ii) *The number of step sizes greater than $\frac{2}{M}$ is $\left(\frac{1}{2} - \varepsilon\right) T$, where $0 \leq \varepsilon \leq O(1/\widehat{\kappa})$ as $\widehat{\kappa} \to \infty$.*

(iii) *For $t \leq \frac{T}{2}$, we have $\gamma_t^{-1} < \frac{1}{m + \frac{2(M-m)t^2}{T^2}}$. Further,*

$$\frac{1}{T} \sum_{t=1}^{T} \gamma_t^{-1} = \frac{\tanh\left(T \operatorname{acosh}\left(\frac{2m}{M-m}\right)\right)}{\sqrt{Mm}} < \frac{1}{\sqrt{Mm}}.$$

(i) is obvious from the construction of γ_t, keeping in the mind the fact that $-1 < \cos(x) < 1$ for $x \in (0, \pi)$.

Proof of (ii). It is obvious that $\gamma_t^{-1} \geq 2/M$ for at most half of the indices $t = 1, \ldots, T$: the nodes γ_t are symmetric with respect to reflection around the axis $\frac{M+m}{2} < \frac{M}{2}$, so at least half of them are greater than or equal to $\frac{M}{2}$. Now, let us establish the lower bound on the number of these steps. We have $\gamma_t^{-1} \leq 2/M$ if and only if

$$\gamma_t = \frac{M+m}{2} - \frac{M-m}{2} \cos \frac{(t - 1/2)\pi}{T} \geq \frac{M}{2}$$
$$\Leftrightarrow \quad \cos \frac{(t-1/2)\pi}{T} \leq \frac{m}{M-m}$$
$$\Leftrightarrow \quad t \geq \frac{1}{2} + \frac{\arccos(\frac{m}{M-m})}{\pi} \cdot T.$$

Call the right hand side the *threshold t^**. Then, using the fact that $\arccos(x) \leq \frac{\pi}{2} - x$ for $x \in [0, 1]$, we have

$$t^* \leq \frac{T}{2} + \frac{T}{\pi} \cdot \frac{m}{M-m} + \frac{1}{2},$$

as required. $\qquad\square$

Proof of (iii). The first statement follows from the upper bound $\cos \alpha \leq 1 - \left(\frac{2\alpha}{\pi}\right)^2$, which is valid for $\alpha \leq \frac{\pi}{2}$:

$$\gamma_t^{-1} = \frac{1}{\frac{M+m}{2} - \frac{M-m}{2} \cos\left(\frac{t - \frac{1}{2}}{T}\pi\right)} < \frac{1}{\frac{M+m}{2} - \frac{M-m}{2} \cos\left(\frac{t}{T}\pi\right)} \leq \frac{1}{\frac{M+m}{2} - \frac{M-m}{2}\left(1 - \left(\frac{2t}{T}\right)^2\right)},$$

from which the claim follows.

A cheap version of the second statement, which is tight up to a constant, can be proven by viewing the summation above as a Riemann sum for a continuous integral. Interestingly, this also provides a way to prove bounds on all moments: for example, using the identity

$$\int_0^1 \frac{1}{(c + x^2)^2} dx = \frac{1}{2}\left(\frac{\operatorname{arccot}(\sqrt{c})}{c^{3/2}} + \frac{1}{c^2 + c}\right) \leq O(1/c^{3/2}),$$

one can verify that the root-mean-square $\sqrt{\frac{1}{T} \sum_{t=1}^{T} \gamma_t^{-2}}$ is bounded by $O(\widehat{\kappa}^{3/4}/M)$.

The exact statement in (iii) is subtler. We suspect that this is known in the literature on Chebyshev spectral methods, but could not find a reference. Consider $q(\lambda) = x^T p(1/\lambda)$, which is $p(\lambda)$ with its coefficients reversed. Then, the sum of

reciprocal roots of $p(\lambda)$ we want is the sum of roots of $q(\lambda)$. By Viète's formula, this is $-a_{T-1}/a_T$, where a_i is the λ^i coefficient of $q(\lambda)$. By the coefficient reversal, a_T is the constant term of $p(\lambda)$, and a_{T-1} is its linear term. Thus, we have:

$$\sum_{t=1}^{T} \gamma_t^{-1} = \frac{\frac{d}{d\lambda}p(\lambda)|_{\lambda=0}}{p(0)} = \frac{2}{M-m} \frac{\frac{d}{dz}\mathcal{T}_T(z)|_{z=\theta}}{\mathcal{T}_T(\theta)}.$$

To reason about the derivatives, we need to introduce the Chebyshev polynomials of the *second* kind, $\mathcal{U}_n(z)$. They can be defined as the unique polynomial satisfying

$$\mathcal{U}_n(\cos\alpha)\sin\alpha = \sin((n+1)\alpha).$$

From this, the cosine characterization of $\mathcal{T}_n(z)$, and the trigonometric substitution $z = \cos\alpha$, we have

$$\frac{d}{dz}\mathcal{T}_n(z) = n\mathcal{U}_{n-1}(z).$$

By definition we have that

$$\mathcal{U}_{n-1}(z) = \frac{(z+\sqrt{z^2-1})^n - (z-\sqrt{z^2-1})^n}{2\sqrt{z^2-1}}, \quad \mathcal{T}_{n-1}(z) = \frac{(z+\sqrt{z^2-1})^n + (z-\sqrt{z^2-1})^n}{2}.$$

Therefore,

$$\frac{\mathcal{U}_{n-1}(z)}{\mathcal{T}_n(z)} = \frac{(z+\sqrt{z^2-1})^n - (z-\sqrt{z^2-1})^n}{(z+\sqrt{z^2-1})^n + (z-\sqrt{z^2-1})^n} \cdot \frac{1}{\sqrt{z^2-1}} = \frac{\tanh(n\,\mathrm{acosh}(z))}{\sqrt{z^2-1}}.$$

Substituting these back, we get that

$$\sum_{t=1}^{T} \gamma_t^{-1} = \frac{2}{M-m} \frac{\frac{d}{dz}\mathcal{T}_T(z)|_{z=\theta}}{\mathcal{T}_T(\theta)} = \frac{2T\tanh(T\,\mathrm{acosh}(\theta))}{(M-m)\sqrt{\theta^2-1}} = \frac{T\tanh(T\,\mathrm{acosh}(\theta))}{\sqrt{Mm}}.$$

$\square$

Finally, we prove the simple observations about the fractal permutation:

Proposition 7. *For all m, M, T and $0 \le i \le \log_2 T$:*

(i) *The largest $\frac{T}{2^i}$ steps η_t in the fractal Chebyshev schedule occur when $t = 1 + 2^i(\tau-1)$, with $\tau = 1, \ldots, \frac{T}{2^i}$.*

(ii) *The subsampled sequence $\{\eta_{1+2^i(\tau-1)}\}$ has the same ordering as the fractal permutation of the same length:*

$$\eta_{1+2^i\tau} = \gamma_{1+2^i(\tau'-1)}^{-1}, \quad \text{where } \tau' = \sigma_{T/2^i}(\tau).$$

Proof. (i) and (ii) are true by the recursive interlacing construction. Let $T_i := \frac{T}{2^i}$. The interlacing step makes it true that the indices in σ_{T_i} are every other element of σ_{2T_i}, the indices in σ_{2T_i} are every other element in σ_{4T_i}, and so forth. Note that γ_t^{-1} are decreasing in t. $\square$

D.2. Infix polynomial bounds

Theorem 9. *For the fractal Chebyshev schedule with m, M, T, and all $1 \le s \le t \le T$:*

$$\|p_{s:t}\|_{[m,M]} \le \left(\frac{M}{m}-1\right) \cdot \prod_{i\in\mathrm{bits}(\zeta+1-s)} \frac{2}{1+\mathcal{T}_{2^i}(\theta)} \cdot \prod_{i\in\mathrm{bits}'(t-\zeta)} \frac{2}{1+\mathcal{T}_{2^i}(\theta)},$$

where ζ is the index such that $s - 1 \le \zeta \le t$ and $\mathrm{lca}(\zeta, \zeta+1)$ is maximized, where

$$\mathrm{lca}(a,b) := \max\{j : j \in \mathrm{bits}(a) \ xor \ j \in \mathrm{bits}(b)\}$$

is the index of the most significant bit at which the binary decompositions of a, b differ.
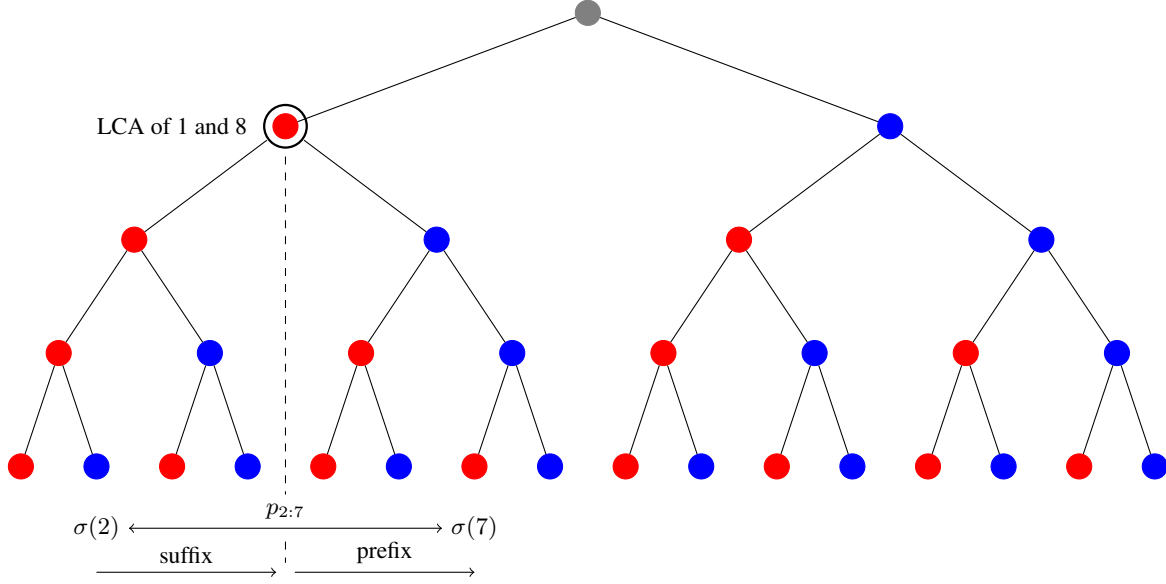
Figure 10: A schematic for the decomposition of the infix $p_{2:7}$ into a suffix and a prefix polynomial corresponding to the child subtrees of the Lowest Common Ancestor

Proof. We have previously shown how to bound the norms of the prefix and suffix polynomial. In this section, we will extend these arguments to bound any infix polynomial $p_{s:t}$ for $1 < s < t < T$. To bound the norm of this polynomial, we will decompose the polynomial into two polynomials lying in disjoint subtrees. On one part we will use the suffix argument whereas on the other part we will use the prefix argument.

To formalize this, we split based on the lowest common ancestor of node $s-1$ and $t+1$.[7] Consider the binary expansions of $s-1 = 2^{s_1} + 2^{s_2} + \ldots + 2^{s_k}$ such that $s_1 > s_2 > \ldots s_k \geq 0$ and $t+1 = 2^{t_1} + 2^{t_2} + \ldots + 2^{t_l}$ such that $t_1 > t_2 > \ldots t_l \geq 0$. Let h be the minimum index such that $s_h \neq t_h$ and let $\zeta = \sum_{i=1}^{h} 2^{t_h}$. Note that h is the level of the lowest common ancestor in the tree, and $\zeta, \zeta + 1$ are the indices splitting the infix between the lowest common ancestor's two subtrees. We will perform the following decomposition based on this,

$$p_{s:t} = p_{s:\zeta} \cdot p_{\zeta+1:t}.$$

It is not hard to see that this decomposition puts the two polynomials in disjoint subtrees: a subtree corresponding to $p_{\zeta-2^{t_h}+1:\zeta}$ and $p_{\zeta+1:\zeta+2^{t_h}}$. Observe that $p_{s:\zeta}$ is a suffix in the left subtree and $p_{\zeta+1:t}$ is a prefix in the right subtree. See Figure 10 for a schematic depiction of the above decomposition.

Let us first analyze $p_{s:\zeta}$. Note that ζ can be $< s$, in that case the polynomial is the empty product with norm 1. Thus, let us assume $\zeta \geq s$. More formally, consider the binary expansions of $M \overset{\text{def}}{=} \zeta + 1 - s = 2^{r_1} + 2^{r_2} + \ldots + 2^{r_j}$ such that $r_1 > r_2 > \ldots r_j \geq 0$. We now perform the following iterative decomposition of the polynomial $p_{s:M}$,

$$
\begin{aligned}
p_{s:M} &= p_{s:T_1} \cdot p_{T_1+1:\zeta} && \text{where } T_1 := M - 2^{r_1}, \\
&= p_{s:T_2} \cdot p_{T_2:T_1} \cdot p_{T_1+1:M} && \text{where } T_2 := T_1 - 2^{r_2}, \\
&\ldots,
\end{aligned}
$$

As in the suffix argument, we will decompose the polynomial into good polynomials starting from the right. Recall that the right child of every node is a good polynomial. It can be seen that every intermediate polynomial $p_{T_i+1:T_{i-1}}$ produced is a good polynomial because each one is the rightmost node at level r_i (i.e. with distance $\log_2 \zeta - r_i$ from the root node of the subtree), restricted to the subtree rooted at the lowest common ancestor of roots s through T_{i-1} (setting $T_0 := \zeta$).

---

[7]Note that we include $s-1$ and $t+1$ when doing this split to ensure that the polynomials do not cover the corresponding subtrees completely and are in fact prefixes and suffixes.

Combining with statement (ii) in Lemma 20, we get

$$\|p_{s:\zeta}\|_{[m,M]} \le \prod_{i=1}^{j} \frac{2}{1 + \mathcal{T}_{2^{r_i}}(\theta)} = \prod_{i \in \text{bits}(\zeta+1-s)} \frac{2}{1 + \mathcal{T}_{2^i}(\theta)} \tag{10}$$

Let us now look at polynomial $p_{\zeta+1:t}$. We will use the prefix argument as before on this polynomial. Consider the binary expansion of $t - \zeta = 2^{q_1} + 2^{q_2} + \ldots + 2^{q_j}$ such that $q_1 > q_2 > \ldots q_j \ge 0$. We decompose $p_{\zeta+1:t}$ into products in the following manner: starting with $\{\zeta + 1, \ldots, t\}$, we iteratively partition

$$
\begin{aligned}
p_{\zeta+1:t} &= p_{\zeta+1:T_1} \cdot p_{T_1+1:t} & \text{where } T_1 := 2^{q_1}, \\
&= p_{\zeta+1:T_1} \cdot p_{T_1+1:T_2} \cdot p_{T_2+1:t} & \text{where } T_2 := T_1 + 2^{q_2}, \\
&\ldots,
\end{aligned}
$$

until we reach $T_j + 1 : t$, which is the empty interval. Note that this partition results in all bad polynomials. We can in fact exactly characterize these polynomials. Define the angle recurrence with α_1 being the angle corresponding to the subtree $p_{\zeta+1:\zeta-2^{t_j}}$ and $\alpha_{i+1} = \frac{\pi - \alpha_i}{2^{q_i - q_{i+1}}}$. It can be seen that

$$p_{\zeta+1:t} = \prod_{i=1}^{j} \mathcal{P}_{2^{q_i}, \alpha_i} \implies \|p_{\zeta+1:t}\|_{[m,M]} \le \prod_{i=1}^{j} \mathcal{B}_{2^{q_i}, \alpha_i}.$$

Using Lemma 23 iteratively on this, we can see that

$$
\begin{aligned}
\|p_{\zeta+1:t}\|_{[m,M]} &\le \mathcal{B}_{2^{q_j}, \frac{\alpha_1}{2^{q_1-q_j}}} \cdot \prod_{i=1}^{k-1} \mathcal{B}_{2^{q_i}, \pi - \alpha_i} \\
&\le \frac{2}{4^{q_j}(\theta - 1)} \cdot \prod_{i=1}^{j-1} \frac{2}{1 + \mathcal{T}_{2^{q_i}}(\theta)} & \text{(using Lemma 20 (ii) and (iii))} \\
&\le \left(\frac{M}{m} - 1\right) \cdot \prod_{i \in \text{bits}'(t-\zeta)} \frac{2}{1 + \mathcal{T}_{2^i}(\theta)}. & \text{(using the definition of } \theta)
\end{aligned}
$$

This gives us the final bound on the infix as,

$$\|p_{s:t}\|_{[m,M]} \le \left(\frac{M}{m} - 1\right) \cdot \prod_{i \in \text{bits}(\zeta+1-s)} \frac{2}{1 + \mathcal{T}_{2^i}(\theta)} \cdot \prod_{i \in \text{bits}'(t-\zeta)} \frac{2}{1 + \mathcal{T}_{2^i}(\theta)}. \tag{11}$$

$\square$

D.3. Infix series bounds

First, we provide a useful bound for an infix series contained entirely within a subtree:

Lemma 24. *For any $N = 2^k$ for some $k > 0$ and any $\delta > 0$,*

$$\sum_{i=1}^{N} \prod_{i \in \text{bits}(N+1-i)} \frac{2}{1 + \mathcal{T}_{2^i}(1 + \delta)} \le \exp\left(\frac{1}{1 + \delta}\right) \cdot \left(\frac{1 + \delta}{\delta}\right)^{1/\log(4)}.$$

Proof. Define $P_N := \sum_{i=1}^{N} \prod_{i \in \text{bits}(N+1-i)} \frac{2}{1 + \mathcal{T}_{2^i}(\theta)}$. Since N is a power of 2, it is not hard to see that

$$P_N = \left(\frac{2}{1 + T_{N/2}(\theta)}\right) \cdot P_{N/2} + P_{N/2} = \left(1 + \frac{2}{1 + T_{N/2}(\theta)}\right) \cdot P_{N/2}.$$

Recursively applying the above, we have

$$P_N = \prod_{i=0}^{k-1} \left(1 + \frac{2}{1 + T_{2^i}(\theta)}\right).$$

To bound the above, let us take $\log$ on both sides. This gives us

$$
\begin{aligned}
\log(P_N) &= \sum_{i=0}^{k-1} \log\left(1 + \frac{2}{1 + T_{2^i}(\theta)}\right) \\
&\leq \sum_{i=0}^{k-1} \frac{2}{1 + T_{2^i}(\theta)} \\
&\leq \sum_{i=0}^{k-1} \frac{1}{1 + 4^i \delta} \\
&\leq \frac{1}{1 + \delta} + \int_0^{k-1} \frac{1}{1 + 4^x \delta} dx \\
&\leq \frac{1}{1 + \delta} + \frac{\log(1 + \delta) - \log(\delta)}{\log 4}.
\end{aligned}
$$

Substituting back gives us the desired result. □

Now, we are ready to prove the main theorem about infix series sums.

Theorem 10. *For a fractal Chebyshev schedule with m, M, T, and all $1 \leq s \leq t \leq T$:*

$$\sum_{t'=s}^{t} \|p_{t':t}\|_{[m,M]} \leq 18\left(\frac{M}{m} - 1\right)\left(\left(\frac{M+m}{2m}\right)^{1/\log(4)}\right)\left(1 + \log\left(\frac{M+m}{2m}\right)\right).$$

Proof. We prove the theorem only for $s = 1$ which subsumes all the cases. To prove the statement we will first consider the binary expansion of t, $\text{bits}(t)$ which is the unique sequence of numbers $\{t_1 \ldots t_k\}$ such that $t = \sum_{j=1}^{k} 2^{t_i}$ and $t_1 > t_2 > t_3 \ldots$. Further, define the following the sequence

$$\bar{t}_0 = 0; \qquad \bar{t}_j = \sum_{j'=1}^{j} 2^{t_{j'}} \quad \forall j \in [k].$$

We will break the sum and analyze it in the following manner:

$$\sum_{j=1}^{k} \underbrace{\sum_{t'=\bar{t}_{j-1}+1}^{\bar{t}_j} \|p_{t':t}\|_{[m,M]}}_{:=T_j}.$$

Before analyzing T_j, we establish some calculations, which will be useful. Firstly, note that for all j, the roots in the range $[\bar{t}_j + 1, t]$ are all contained inside a subtree of height 2^{t_j} in the tree representation. Therefore the polynomial $p_{\bar{t}_j+1,t}$ forms a prefix polynomial within the subtree for which we can apply the bounds in Theorem 8 (see usage in the proof of Theorem 9 on how Theorem 8 applies to prefix of any subtree). Using the above, we get the following bounds:

$$\|p_{\bar{t}_j+1:t}\|_{[m,M]} \leq \left(\frac{M}{m} - 1\right) \prod_{r=j+2}^{k} \frac{2}{1 + \mathcal{T}_{2^{t_r}}(\theta)}. \tag{12}$$

Note that for the above and the rest of this section if the sum Σ and product $\prod$ is over an empty set then they are assumed to be 0 and 1 respectively. Furthermore, note that for any j and $t' \in [\bar{t}_{j-1} + 2, \bar{t}_j]$, the range of roots $[t', \bar{t}_j]$, belongs to a

subtree of height 2^{t_j} in the tree representation. In particular the infix polynomial $p_{t',\bar{t}_j}$ is a suffix polynomial within the tree and the bound from Theorem 8 can be invoked to give

$$\|p_{t',\bar{t}_j}\|_{[m,M]} \leq \prod_{r \in \text{bits}(\bar{t}_j+1-t')} \frac{2}{1 + \mathcal{T}_{2^r}(\theta)}. \tag{13}$$

We are now ready to analyze the terms T_j for any j as follows.

$$
\begin{aligned}
T_j &\leq \|p_{\bar{t}_{j-1}+1:t}\|_{[m,M]} + \sum_{t'=\bar{t}_{j-1}+2}^{\bar{t}_j} \|p_{t':\bar{t}_j}\|_{[m,M]} \|p_{\bar{t}_j+1:t}\|_{[m,M]} \\
&\leq \left(\frac{M}{m}-1\right)\left(\prod_{r=j+1}^{k}\frac{2}{1+\mathcal{T}_{2^{t_r}}(\theta)} + \prod_{r=j+2}^{k}\frac{2}{1+\mathcal{T}_{2^{t_r}}(\theta)}\left(\sum_{t'=\bar{t}_{j-1}+2}^{\bar{t}_j}\left(\prod_{r\in\text{bits}(\bar{t}_j+1-t')}\frac{2}{1+\mathcal{T}_{2^r}(\theta)}\right)\right)\right) \\
&\leq \left(\frac{M}{m}-1\right)\left(\prod_{r=j+1}^{k}\frac{2}{1+\mathcal{T}_{2^{t_r}}(\theta)} + \prod_{r=j+2}^{k}\frac{2}{1+\mathcal{T}_{2^{t_r}}(\theta)}\left(3\left(\frac{\theta}{\theta-1}\right)^{1/\log(4)}\right)\right) \\
&\leq 6\left(\frac{M}{m}-1\right)\left(\prod_{r=j+2}^{k}\frac{2}{1+\mathcal{T}_{2^{t_r}}(\theta)}\left(\left(\frac{\theta}{\theta-1}\right)^{1/\log(4)}\right)\right).
\end{aligned}
$$

In the above, the first inequality follows from triangle inequality, the second from (12),(13) and the third from Lemma 19. Summing over j now gives us the bound as follows:

$$
\begin{aligned}
\sum_{j=1}^{k} T_j &\leq 6\left(\frac{M}{m}-1\right)\left(\sum_{j=1}^{k-2}\left(\prod_{r=j+2}^{k}\frac{2}{1+\mathcal{T}_{2^{t_r}}(\theta)}\right)+2\right)\left(\left(\frac{\theta}{\theta-1}\right)^{1/\log(4)}\right). \\
&\leq 6\left(\frac{M}{m}-1\right)\left(\sum_{j=1}^{k-2}\left(\frac{2}{1+\mathcal{T}_{2^{t_{j+2}}}(\theta)}\right)+2\right)\left(\left(\frac{\theta}{\theta-1}\right)^{1/\log(4)}\right). \\
&\leq 6\left(\frac{M}{m}-1\right)\left(\log\left(\frac{\theta}{\theta-1}\right)+3\right)\left(\left(\frac{\theta}{\theta-1}\right)^{1/\log(4)}\right). \\
&\leq 18\left(\frac{M}{m}-1\right)\left(\left(\frac{M+m}{2m}\right)^{1/\log(4)}\right)\left(1+\log\left(\frac{M+m}{2m}\right)\right).
\end{aligned}
$$

This concludes the theorem.

$\square$

E. Proofs for Section 3

E.1. Modifications to the fractal schedule

Proof of Proposition 13 (i). For any $1 \leq t \leq T-1$, consider $p_{1:t}$ and $p_{1:t+1}$ (under the reversed permutation). Both decompose into a product of good polynomials, whose levels in the tree are given by $\text{bits}(t)$ and $\text{bits}(t+1)$. Notice that $\text{bits}(t+1) \setminus \text{bits}(t)$ contains exactly one element (the index where the carrying operation stops in binary addition); call it c. Then, $\text{bits}(t) \setminus \text{bits}(t+1) = \{1, \ldots, c-1\}$. Thus, it suffices to prove that

$$\frac{2}{1+\mathcal{T}_{2^c}(\theta)} \leq \prod_{i=0}^{c-1}\frac{2}{1+\mathcal{T}_{2^i}(\theta)}.$$

Notice that when $i \geq 1$, we have

$$\frac{2}{1+\mathcal{T}_{2^i}(\theta)} = \frac{1}{\mathcal{T}_{2^{i-1}}^2(\theta)},$$

so when $c \geq 1$, the statement we wish to prove reduces to

$$\mathcal{T}_{2^{c-1}}^2(\theta) \geq \frac{1+\theta}{2} \cdot \prod_{i=0}^{c-2} \mathcal{T}_{2^i}^2(\theta).$$

This is true because $\frac{1+\theta}{2} \leq \theta^2 = \mathcal{T}_1^2(\theta)$, and we can recursively apply the following inequality:

$$\mathcal{T}_{2n}(\theta) = 2\mathcal{T}_n^2(\theta) - 1 \geq \mathcal{T}_n^2(\theta),$$

which holds because $\theta \geq 1$. $\qquad\square$

The other proofs in Section 3 follow immediately from the definitions.

E.2. Overstepping with conservative parameters

Proof of Theorem 14. With $p = p_{1:T}$ as the shifted Chebyshev polynomial with parameters m, M, we wish to bound $\|p\|_{[\lambda_{\min}, M]}$. In the range $[m, M]$, the bounds from Theorem 3 hold; moreover, the cosine formula for the Chebyshev polynomials implies that the inequality is tight at the boundary: the maximum is achieved at $p(m)$. In the remaining part $[\lambda_{\min}, M]$, which is outside the range of the roots of p, $p(\lambda)$ grows monotonically as λ decreases in this interval. Thus, it will suffice to derive a bound for $p(\lambda_{\min})$.

When $m < M$, we use the notation $\theta = \frac{M+m}{M-m}$, and define $z_{\max} := \frac{M+m-2\lambda_{\min}}{M-m}$ (the image of $\lambda_{\min}$ under the bijection).

$$\frac{\mathcal{T}_T(z_{\max})}{\mathcal{T}_n(\theta)} = \frac{\cosh(T \operatorname{acosh}(z_{\max}))}{\cosh(T \operatorname{acosh}(\theta))} = \frac{e^{T \operatorname{acosh}(z_{\max})} + e^{-T \operatorname{acosh}(z_{\max})}}{e^{T \operatorname{acosh}(\theta)} + e^{-T \operatorname{acosh}(\theta)}} \leq \frac{2e^{T \operatorname{acosh}(z_{\max})}}{e^{T \operatorname{acosh}(\theta)}}$$

$$= 2 \left(\frac{z_{\max} + \sqrt{z_{\max}^2 - 1}}{\theta + \sqrt{\theta^2 - 1}} \right)^T = 2 \left(1 - \frac{\theta + \sqrt{\theta^2 - 1} - z_{\max} - \sqrt{z_{\max}^2 - 1}}{\theta + \sqrt{\theta^2 - 1}} \right)^T.$$

The quantity in the fraction is equal to

$$\frac{\frac{M+m+2\sqrt{Mm}}{M-m} - \frac{M+m+2\lambda_{\min}+2\sqrt{(M-\lambda_{\min})(m-\lambda_{\min})}}{M-m}}{\frac{M+m+2\sqrt{Mm}}{M-m}} = 2 \cdot \frac{\lambda_{\min} + \sqrt{Mm} - \sqrt{(M-\lambda_{\min})(m-\lambda_{\min})}}{(\sqrt{M} + \sqrt{m})^2},$$

as required. The same is concluded for $m = M$ by taking the limit $m \to M$.

$\qquad\square$

E.3. Conjugate gradient schedule

The simplest definition of the conjugate gradient algorithm, without having to worry about how to implement the iterations in linear time, is the non-iterative formula

$$x_{t+1} := \min_{\substack{\deg p \leq t \\ p(0)=1}} \|p(A)(x_1 - x^*)\|_A, \qquad (14)$$

where $\|x\|_A := \sqrt{x^\top A x}$, and the minimization is over polynomials with real coefficients.

Theorem 15 (Conjugate gradient schedule). *For all positive definite matrices $A \in \mathbb{R}^{d \times d}$ and $b \in \mathbb{R}^d$, there exists a multiset of real numbers $\{\eta_t\}$, all in the interval $[\frac{1}{\lambda_{\max}(A)}, \frac{1}{\lambda_{\min}(A)}]$, such that x_{T+1} as defined by the conjugate gradient algorithm (Equation (14)) is equal to x_{T+1} as defined by gradient descent (Equation (1)).*

Proof. Let

$$p^* \in \arg\min_{\substack{\deg p \leq t \\ p(0)=1}} \|p(A)(x_1 - x^*)\|_A.$$

By the fundamental theorem of algebra applied to $p^* : \mathbb{R} \to \mathbb{R}$, and noting that 0 cannot be a root of p^*, this is obviously true if the step sizes η_t are allowed to be arbitrary complex numbers. We will show that there exists a minimal real-rooted polynomial that achieves the minimum in Equation (14), with all roots lying in the specified interval. To do this, we will start with a p^* with possibly complex roots, and transform it to fit our conditions, without increasing the residual norm.

Let $\mathcal{F}(p)$ denote the functional that returns the squared residual of a residual polynomial:

$$\mathcal{F}(p) := \|p(A)(x_1 - x^*)\|_A^2 = \sum_{(\lambda_i, u_i) \in \mathrm{eigs}(A)} \lambda_i \left[p(\lambda_i)\right]^2 (u_i^\top (x_1 - x^*))^2,$$

where $\mathrm{eigs}(A)$ denotes the eigendecomposition of A. Define a partial ordering on functions $p : [\lambda_{\min}(A), \lambda_{\max}(A)] \to \mathbb{R}$:

$$p \succcurlyeq q \quad \Leftrightarrow \quad |p(\lambda)| \geq |q(\lambda)| \quad \forall \lambda \in [\lambda_{\min}(A), \lambda_{\max}(A)].$$

Notice that $\mathcal{F}(p)$ is monotone with respect to ($\succcurlyeq$). That is,

$$p \succcurlyeq q \quad \to \quad \mathcal{F}(p) \succcurlyeq \mathcal{F}(q).$$

Now we can complete the proof.

Roots are real w.l.o.g. By the complex conjugate root theorem, if p^* has any complex roots, they come in conjugate pairs $(a \pm bi)$ with matching multiplicities. Multiplying these in pairs gives us quadratic factors $q(a, b) := (x - a)^2 + b^2$. But $|q(a, 0)| \leq |q(a, b)|$, so we can construct a real-rooted polynomial p' with the same degree as p^* such that $p' \preccurlyeq p^*$, by deleting the complex parts of each root.

Roots lie within the eigenvalue range w.l.o.g. By the above, p^* is real-rooted; write $p^*(\lambda) = \prod_{i=1}^{\deg p}(1 - \lambda/\alpha_i)$. Split the real line into intervals $I_1 = (-\infty, 0)$, $I_2 = (0, \lambda_{\min}(A))$, $I_3 = [\lambda_{\min}(A), \lambda_{\max}(A)]$, and $I_4 = (\lambda_{\max}(A), \infty)$. We will show that we can move all the roots of p^* into I_3 without increasing $\mathcal{F}$. For roots $\alpha \in I_1$ and $\alpha \in I_4$, notice that $(1 - \lambda/\lambda_{\max}(A)) \preccurlyeq (1 - \lambda/\alpha)$, so we can change those roots to $\lambda_{\max}(A)$. For roots $\alpha \in I_2$, notice that $(1 - \lambda/\lambda_{\min}(A)) \preccurlyeq (1 - \lambda/\alpha)$, so that we can change those roots to $\lambda_{\min}(A)$. By making these changes, we have obtained a polynomial p'' with the desired properties, such that $\mathcal{F}(p'') \leq \mathcal{F}(p^*)$.

Finally, the roots of p^* give the reciprocal step sizes needed for the final iterate of gradient descent to match that of conjugate gradient. If $\deg p < T$, then we have more step sizes to assign than roots, and we can simply assign the remaining $T - \deg p$ step sizes to 0. This completes the proof. $\square$

E.4. Non-convex combination lock

This construction is a simple variant of the "needle-in-haystack" construction for global non-convex optimization with a first-order oracle. This statement can be strengthened, but we optimize for brevity.

Proposition 16. *Let $(\eta_1^*, \ldots, \eta_T^*)$ be any sequence of positive real numbers, and $0 < \delta \leq \frac{1}{2} \min_t \eta_t^*$. Then, there exists a function $f : \mathbb{R}^T \to \mathbb{R}$ for which:*

- *f is infinitely differentiable. All of its derivatives are $O(1/\delta)$.*

- *$-1 \leq f(x) \leq 2$ for all $x \in \mathbb{R}^T$, and $\min_{x \in \mathbb{R}^T} f(x) = -1$, where $\eta_{\min} := \min_t \eta_t^*$. The minimizer is unique.*

- *Let x_{out} be the final iterate of gradient descent, starting from $x_1 = 0$ and with learning rate schedule $(\eta_1, \ldots, \eta_t)$. Then, if $\eta_t = \eta_t^*$ for each t, then $x_{\mathrm{out}} = -1$. Furthermore, for any t we have $|\eta_t - \eta_t^*| \geq \delta$, then $f(x_{\mathrm{out}}) \geq 0$.*

Proof. We will start by constructing a non-smooth such function. For all $z \in \mathbb{R}$, $\eta > 0$, define

$$g_{\eta_T^*}^{(T)}(z) = \begin{cases} 2 & z \in (-\infty, -\delta/2] \\ 1 - z & z \in [-\delta/2, \eta_T^* - \delta/2) \\ -1 & z \in [\eta_T^* - \delta/2, \eta_T^* + \delta/2] \\ 0 & z \in (\eta_T^* + \delta/2, \infty) \end{cases}.$$

Starting at $z = 0$, one step of gradient descent on $g^{(T)}$ with learning rate η reaches a global minimizer only if $\eta = \eta_T^*$.

Now, for each $t = T - 1, \ldots, 1$, define

$$g_{\eta_t^*}^{(t)}(z, z_{t+1}, \ldots, z_T) = \begin{cases} 2 & z \in (-\infty, -\delta/2] \\ 1 - z & z \in [-\delta/2, \eta_t^* - \delta/2) \\ g_{\eta_{t+1}^*}^{(t+1)}(z_{t+1}, \ldots, z_T) & z \in [\eta_t^* - \delta/2, \eta_t^* + \delta/2] \\ 0 & z \in (\eta_t^* + \delta/2, \infty) \end{cases},$$

and so forth. Then let $g := g_{\eta_1^*}^{(1)}$ be our unsmoothed function of choice: define $f = g * \psi(2x/\delta)$, where

$$\psi(x) = \begin{cases} \frac{1}{Z} e^{-\frac{1}{1 - \|x\|^2}} & \|x\| < 1 \\ 0 & \text{otherwise} \end{cases},$$

where $Z = \int_{x \in \mathbb{R}^T} \psi(2x/\delta)\, dx \leq O(1/\delta)$. Then:

- f is infinitely differentiable because g is bounded and $\psi(2x/\delta)$ is infinitely differentiable.

- $-1 \leq f(x) \leq 2$ by Young's inequality. Since the support of $\psi(2x/\delta)$ is $\delta/2$ times the unit sphere, and $g = -1$ exactly on the ℓ_∞ ball of radius $\delta/2$ centered at $(\eta_1^*, \ldots, \eta_T^*)$, the f has a unique minimum at $(\eta_1^*, \ldots, \eta_T^*)$.

- By the construction, gradient descent with learning rates $\{\eta_t^*\}$, starting at 0, encounters the gradient sequence $\{e_t\}$, the elementary unit vectors, so it outputs the minimizer. At each iteration t, x_{t+1} must lie in the span of $\{e_1, \ldots, e_t\}$ in order for x_{out} to reach the minimizer. If $|\eta_{t'} - \eta_{t'}^*| \geq \delta$ at any iteration t, this invariant cannot hold, since the next gradient is in the span of e_t.

$\square$

It may not be overly pessimistic to think of tuning the learning rate schedule in deep learning as a "needle-in-haystack" search problem. Learning rate schedules have been observed to affect generalization behavior in practice (Jiang et al., 2020; Agarwal et al., 2020a), so that restarting training with a new schedule is the only way to escape poor local optima.

E.5. No acceleration from the simple spiky schedule

A natural choice of self-stabilizing learning rate schedule is that which takes one large step of size η^+ to make progress in directions with shallow curvature, then several small steps of size η^- to correct for the overshooting of the large step. This is the cyclic schedule considered by (Oymak, 2021), which is shown to obtain a $\log(\kappa)$ "super-convergent" rate under the assumption that the eigenvalues of A lie in two clusters. In this section, we provide a brief note on why this cannot obtain the $\sqrt{\kappa}$ rate on general strongly convex quadratics.

Proposition 25. *Let* $\eta^+, \eta^- \in [1/\lambda_{\max}, 1/\lambda_{\min}]$, *and suppose* $\eta^+ \geq 10\eta^-$. *Let* n *be a positive integer. Consider the polynomial*

$$p(\lambda) := (1 - \eta^+ \lambda)(1 - \eta^- \lambda)^n.$$

Then, if $n \leq 0.1 \eta^+/\eta^-$, *it must be true that*

$$\|p^m\|_{[\lambda_{\min}, \lambda_{\max}]} > 1.34^m$$

for all positive integers m.

Proof. We have

$$\frac{dp}{d\lambda} = -\eta^+(1 - \eta^- \lambda)^n - n\eta^-(1 - \eta^+ \lambda)(1 - \eta^- \lambda)^{n-1} = -(1 - \eta^- \lambda)^{n-1}\left(n\eta^-(1 - \eta^+ \lambda) + \eta^+(1 - \eta^- \lambda)\right),$$

which has a root at $\lambda^* := \frac{\eta^+ + n\eta^-}{(n+1)\eta^+\eta^-}$. Then,

$$
\begin{aligned}
\|p\|_{[\lambda_{\min}, \lambda_{\max}]} \;\geq\; |p(\lambda^*)| &= \frac{\frac{\eta^+}{\eta^-} - 1}{n+1}\left(\left(1 - \frac{\eta^-}{\eta^+}\right)\left(1 - \frac{1}{n+1}\right)\right)^n \\
&\geq \frac{1}{e}\frac{\frac{\eta^+}{\eta^-} - 1}{n+1}\left(1 - \frac{\eta^-}{\eta^+}\right)e^{-n\eta^-/\eta^+} \geq \frac{1}{e}\frac{\frac{\eta^+}{\eta^-} - 1}{0.1\frac{\eta^+}{\eta^-} + 1}\left(1 - \frac{\eta^-}{\eta^+}\right)e^{-n\eta^-/\eta^+} \\
&\geq \frac{1}{e}\cdot\frac{9}{2}\cdot 0.9\cdot e^{-0.1} > 1.34.
\end{aligned}
$$

$\square$

Note that p^m is the residual polynomial associated with repeating this cyclic schedule m times. Thus, if the unstable step size in this schedule is κ^α times larger than the stable step size, the number of small steps required to prevent exponential blowup of the residual polynomial norm is $\Omega(\kappa^\alpha)$. For any $\alpha \in [0,1]$, we have $\|p^m\| \geq |p(\lambda_{\min})|^m \geq (\exp(-O(\sqrt{\kappa})))^m$, so that $m \geq \Omega(\kappa^{1-\alpha}\log(1/\varepsilon))$ cycles are required to make the residual norm at most ε. But we have shown that each cycle requires $\Omega(\kappa^\alpha)$ steps; thus, no choice of η^+, η^-, n can get a better unconditional convergence rate than $O(\kappa\log(1/\varepsilon))$.

F. Experimental details and supplements

F.1. Visualization of the quadratic (theoretical) setting

In Figure 3, we provide a simple illustrative numerical experiment visualizing the tradeoffs; details are below.

This is an instance in dimension $d = 100$ with $A = L/\lambda_{\max}(L) + 0.1I$, where L is the Laplacian matrix of the path graph on 100 vertices; this objective is 2.2-smooth and 0.2-strongly convex, b was sampled from $\mathcal{N}(0, I_{100})$, and $x_1 = 0$. Gradient descent (the non-accelerated baseline) was run with a learning rate of 0.9, determined via grid search on 0.1 increments (convergence was not significantly improved with a finer grid). The Chebyshev nodes were chosen with $m = 0.2, M = 2.2, T = 32$, resulting in the four learning rate schedules shown.

This experiment was run with 80-bit (long double) precision, for illustrative purposes. At 32 or even 64 bits, or with larger T, the increasing schedule exhibits exponential blowup of numerical noise, even in this small setting. In the plot to the right, i.i.d. spherical Gaussian noise $\sim \mathcal{N}(0, 0.0005I)$ was added.

F.2. One-dimensional counterexample

In Section 5.1, we noted $\log\cosh(x) + 0.01x^2$ as a "counterexample by numerical simulation" to the hypothesis that gradient descent with the fractal Chebyshev schedule converges on general convex functions. This function is 1.02-smooth and 0.02-strongly convex.

To refute the possibility that Theorem 3 holds, it simply suffices to show that there is some setting $m \leq 0.02, M \geq 1.02$ and T such that the theoretical bound does not hold. We chose $m = 0.01, M = 5, T = 32$ (noting that it was quite easy to generate counterexamples). The initial iterate was set to $x_1 = 2$. The trajectory compared to the theoretical bound is shown in Figure 11.

Results are shown in Figure 11. Gradient descent (constant step size $1/M$) and Nesterov's accelerated gradient (constant step size $1/M$; momentum parameter $\gamma = 1 - \sqrt{1/0.02}$) are shown for comparison. Of course, none of these parameters are optimized; in the one-dimensional case, it is possible to reach the exact minimizer in one iteration.

We conjecture that stronger negative results (say, an infinite family of counterexamples for all T) can be constructed.

F.3. Convex experiments

To examine the empirical behavior of gradient descent with the fractal Chebyshev learning rate schedule on a deterministic higher-dimensional convex (but not quadratic) loss, we used the benchmark of logistic regression (with trainable biases, thus totaling $d = 7850$ trainable parameters) on the MNIST dataset (LeCun et al., 1989) with normalized raw pixel features, with an ℓ_2 regularization coefficient of 10^{-3}. The initial iterate was set to zero during all runs, for a completely deterministic
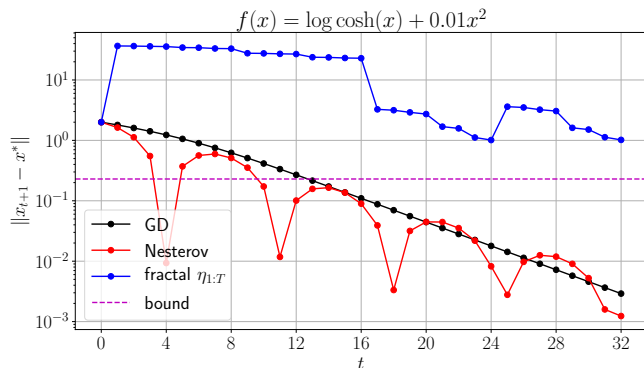
Figure 11: Non-convergent behavior of the fractal Chebyshev schedule on $f(x) = \log\cosh(x) + 0.01x^2$. The final iterate fails to follow the convergence bound from Theorem 3.
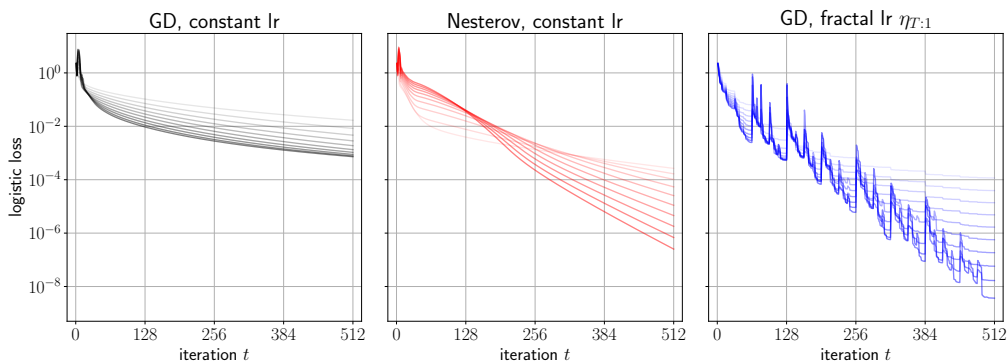

Figure 12: Convex deterministic MNIST experiments: comparison of classic non-accelerated and accelerated algorithms, and the fractal Chebyshev schedule. Most opaque curves correspond to the optimal tuned constant multiplier; lower opacity corresponds to shrinking the steps in equally spaced increments, down from 0.5 to 0.1. Gradient descent with the fractal schedule makes non-local progress and converges at the accelerated rate in practice.

setting. To measure the global minimum, we ran L-BFGS (Liu & Nocedal, 1989) until convergence to the 64-bit numerical precision floor.

We compared three iterative algorithms: gradient descent with a constant learning rate (known in theory to get the slow rate), Nesterov's accelerated gradient descent with a constant learning rate and momentum parameter 0.9 (known in theory to get the accelerated rate), and gradient descent with the reversed fractal schedule $\eta_{T:1}$ (no known theoretical guarantees in this setting). We tuned the constant learning rates in increments of 0.1 until divergence (arriving at 0.5 as the largest stable learning rate). For the fractal schedule, we chose $m = 0.0006, M = 5$, and tuned the global learning rate multiplier in increments of 0.1 (arriving also at 0.5). The step sizes in the schedule were in the range $[0.05, 408.65]$, with a mean of 4.56 (*much* larger than the maximum stable constant learning rate).

Figure 12 shows our results: in this setting, gradient descent can achieve accelerated convergence by overstepping the threshold of guaranteed local progress. We used the reverse schedule here (largest step last), as suggested for parameter stability in the noiseless setting. The forward schedule converged with accelerated rates for some choices of hyperparameters, but convergence on this non-quadratic objective was sensitive to initial large steps.

It is not the purpose of Figure 12 to demonstrate a comparison between Nesterov's acceleration and the fractal schedule; this is a somewhat brittle comparison and is sensitive to hyperparameter choice and floating-point precision. The quantitative comparison from this experiment is between gradient descent with the optimal constant learning rate and any fractal Chebyshev schedule which outperforms it. The Nesterov training loss curves are provided as an illustration only.
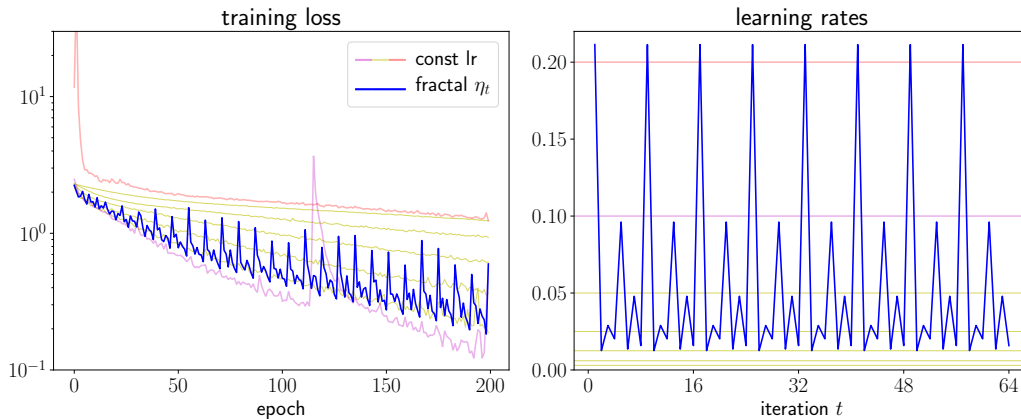
Figure 13: ResNet-18/CIFAR-10 training with batch size 8192 and a repeated $T = 8$ fractal Chebyshev schedule. *Left:* Training loss curves. *Right:* Learning rates; the schedule pokes through the edge of stability (magenta and red) without destabilizing training.

F.4. Deep learning experiments

We present some simple experiments for the fractal Chebyshev schedule on deep neural networks. The purpose of this preliminary study is to demonstrate that the constant learning rate "edge of stability" can be overstepped without causing training to diverge, using a carefully designed schedule. We do not make claims about end-to-end performance improvements that are robust under ablation and tuning other hyperparameters. A more systematic examination of the behavior of "spiky" learning rate schedules in deep learning is left for future work.

In the deep learning experiments, it is most convenient to think of a learning rate schedule as a time-varying multiplier on top of an existing baseline. Thus, it is most helpful to set the scaling hyperparameters to let the fractal schedule act as a "learning rate bonus": set $M = 1$, so that the smallest multiplier is approximately 1, the largest $\approx 1/m$, and the mean $\approx 1/\sqrt{m}$.

CIFAR-10/ResNet experiments. The experiments were conducted on the CIFAR-10 dataset on a pre-activation ResNet-18 model (He et al., 2016) with $d \approx 11M$ parameters. As a baseline, we trained the network with vanilla minibatch SGD with batch size 8192; the choice of a large batch was made to reduce confounding factors arising from stochasticity, and we omitted the usual practice of momentum in order to remove temporal correlations between step sizes. To find the edge of stability for training with a constant learning rate, we searched over the fixed learning rate parameter on an exponential grid of powers of 2, as depicted in Figure 13 (right): the learning rate of 0.05 leads to stable and the best results; at 0.1, training is subject to destabilizing outliers, and at 0.2, the model does not train at all; this is summarized in the fainter training loss curves in Figure 13 (left).

We applied a cyclic fractal schedule with $m = 0.05$, $M = 1$, and $T = 8$, as a periodic multiplier on top of the constant learning rate 0.125; this is pictured in Figure 13 (right) as the blue curve. Although this schedule uses large learning rates that would cause unstable training, the fractal Chebyshev schedule periodically surpasses these learning rates while maintaining stable training.

We did not evaluate the model based on generalization performance (indeed, we have removed the usual practices of momentum, random cropping image augmentation, and a decaying learning rate schedule), but in this set of experiments we found the test accuracy to be slightly higher (83%) than the best constant learning rate baseline (81%). The stability results were consistent over 5 trials. These experiments were run in PyTorch with an $8\times$ NVIDIA Tesla V100 GPU machine, and each run took less than 30 minutes for 200 epochs.

MNIST experiments with a small neural network. We chose a simpler and cheaper-to-train model to present a few more empirical insights on the behavior of fractal Chebyshev schedules beyond known theory. Namely, we use the model
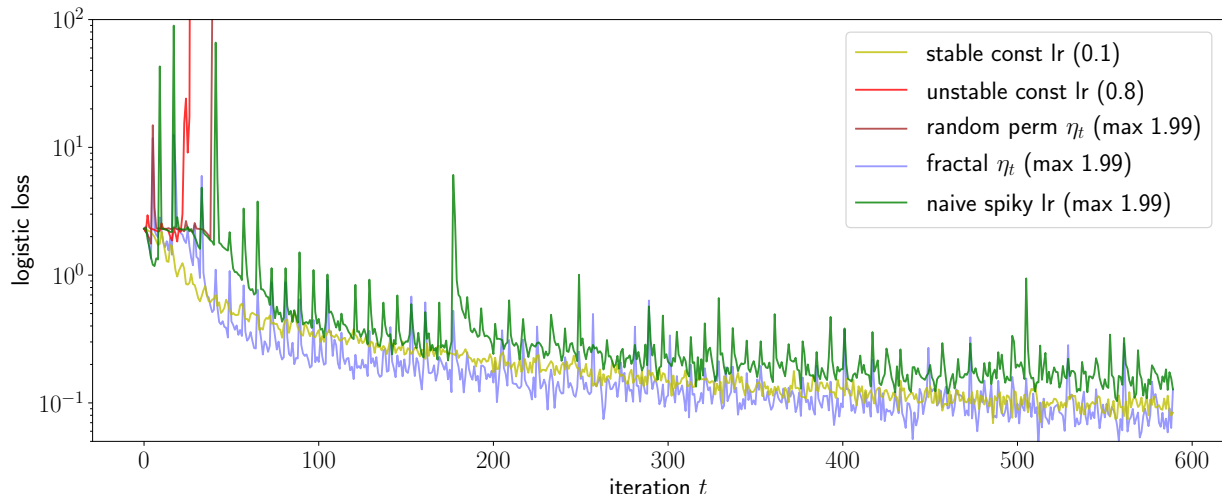
Figure 14: MNIST experiments, to show that baselines fail to stabilize training as successfully as the fractal Chebyshev schedule. The constant learning rate edge of stability is surpassed at a constant learning rate of 0.8, while the fractal schedule can take steps of up to 1.99. Random permutations of the same schedule cause divergent training, as does a simple "spiky" schedule which only oversteps once per cycle.

for MNIST classification from the PyTorch tutorial[8]: two convolutional layers, followed by two fully-connected layers, with a total of ~ 1.2M parameters. The model was trained with SGD with batch size 1024.

In this setup, the same methodology as the larger-scale experiments was used on a finer grid (linearly spaced between 0.1 and 0.8) to determine a stable constant learning rate (0.1) and an unstable one (0.8). A fractal Chebyshev schedule with $m = 1/20, M = 1, T = 64$ accelerated convergence when applied to the stable constant learning rate baseline. However, randomly permuting this schedule caused divergent training. Furthermore, applying a periodic multiplier of $(20, 1, 1, 1, 1, 1, 1, 1)$ resulted in *worse* convergence. This exploratory study suggests that some of the self-stabilizing behavior of the fractal schedule in the theoretical setting (where large steps are stabilized by internal copies of Chebyshev iteration, which also consist of large steps) may hold, even for deep networks. Results were consistent over 10 trials.

These experiments were run in PyTorch on a $1\times$ NVIDIA Tesla P100 GPU machine, and each run took around less than 1 minute for 10 epochs.

G. Additional discussion on related work

G.1. Fractal cyclic Chebyshev iterative methods

We provide a review of the line of work that serves as the origin of these fractal permutations. These were motivated by the setting of *cyclic iteration methods* for solving linear equations by least-squares in Banach spaces, a primitive in finite element methods for solving partial differential equations. All citations we could find for this line of work have been in the context of numerical methods for least-squares; much of it is in the Russian language, untranslated. We have not encountered prior work linking these methods to machine learning.

Lebedev & Finogenov (1971) construct the fractal permutation seen in this paper, and proved the prefix and suffix bounds, as well a series bound for all prefixes (as opposed to infixes). This remarkable paper is the starting point for us (as well as the authors, evidently). Appendix C is an attempt to make that paper more accessible (it is far longer than the original paper).

Lebedev & Finogenov (1973) consider generalized versions of the construction, where T is any positive integer, and the polynomial splitting is performed with the prime factors. They describe general conditions under which stability of a cyclic method (thus only prefix, suffix, and series bounds) can be achieved, and prove stability theorems like the previous work about constructions where the only prime factors of T and 2 and 3.

---

[8]https://pytorch.org/tutorials/recipes/recipes/defining_a_neural_network.html

Lebedev & Finogenov (1976), working in this generalized setting, also analyze the stability of a single cycle of a fractal Chebyshev schedule. They consider series sums of infixes where the series terminate at indices d_n which form a divisor chain of T, rather than general indices.

Lebedev & Finogenov (2002; 2004) provide English-language overviews of selected methods and theorems from this line of work.

In all of these works, the authors were motivated by high-precision settings in computational physics, rather than statistical or model error. This is perhaps why they were motivated to generalize the results of (Lebedev & Finogenov, 1971) in their follow-up papers in a different way than ours. Thus, even though the ideas and motivations overlap with the ones considered in our work, especially in (Lebedev & Finogenov, 1976), the authors might not have found it important to bound the noise stability of every intermediate iteration of the algorithm. However, thinking of the perturbations as arising from statistical error or model misspecification, this is a natural notion for our setting. We could not find a way to immediately derive our estimates from any theorem or intermediate lemma in (Lebedev & Finogenov, 1976).

It remains an interesting direction for future work to find efficient algorithms to compute stable schedules for general T, and analyze their every-iterate stability like in our work. We could not see immediate ways to extend Theorems 9 and 10 to their more general classes of schedules.

G.2. Learning rate schedules and tradeoffs in practice

State-of-the-art models do not show any signs of consensus towards principled or fully-understood learning rate schedules, adaptive or otherwise. A common practice has been to use a cosine learning rate schedule, originally proposed for cyclic warm restarts (Loshchilov & Hutter, 2016) but widely adopted in its one-cycle form. For example, GPT-3 (Brown et al., 2020) was trained with a cosine schedule. Large-scale empirical studies (Shallue et al., 2019) indicate that the optimal choice of learning rate schedule is sensitive to the batch size. See the discussion on learning rate schedules in (You et al., 2019) for a discussion of recent empirical observations in pretraining large-scale language models.

Several papers study the theoretical tradeoffs between stability and acceleration in large-scale stochastic optimization: (Bottou & Bousquet, 2007; Devolder et al., 2014; Chen et al., 2018; Agarwal et al., 2020b). A common message throughout these papers is that the best choice of iterative optimization algorithm depends in general on the data, model, and computational resources.

(Cohen et al., 2021) provide an empirical account of the insufficiency of second-order Taylor approximations of the loss function in deciding the correct learning rate. (Agarwal et al., 2020a) point out that learning rate schedules are entangled with adaptive gradient methods.

G.3. Learning rate schedules in theory

While learning rate schedules while ubiquitously used in practice, the diversity of existing practical learning rate schedules has received little theoretical treatment. In convex optimization, learning rate schedules have primarily been employed in stochastic settings, in particular to correctly average the zero-mean noise. It is well known in the stochastic and online optimization literature that a step schedule akin to $t^{-1/2}$ is necessary for the convergence of stochastic gradient descent. In the case of zero-mean bounded variance stochastic noise, the AC-SA algorithm proposed by (Lan, 2012) which achieves optimal convergence rates employs an effective step decay schedule of $t^{-3/2}$. In a complementary line of work, (Ge et al., 2019) show that for the streaming least squares regression problems, no polynomial decay of learning rates achieves the minimax rate; on the other hand the rate is achieved by the geometric decay learning rate schedule which is very popular in practice.

An alternative point of view towards the power of learning rate schedules arises from the Polyak step size (Polyak, 1987; Hazan & Kakade, 2019), which is a single learning rate per step which generalizes the classical gradient descent oblivious to the smoothness/strong convexity properties of the function. The Polyak step size requires the knowledge of the optimality gap at any point. The vanilla version of the Polyak step size is unable to provide accelerated rates; an extension of these ideas to momentum has been carried out by Barré et al. (2020).

Practical deep learning models due to the presence of normalization layers lead to homogeneous models. For such models, Li & Arora (2019) perhaps surprisingly show that the standard training algorithm which includes weight decay and momentum is equivalent to performing an exponentially increasing learning rate schedule. Li et al. (2020) further explore the intricate

interaction of weight decay and learning rates in such models proposing the notion of an intrinsic learning rate. The practice of using a large initial learning rate in optimization from the point of view of better generalization has been theoretically investigated in (Li et al., 2019) (see references herein for a detailed treatment of the topic).

A line of work (Orabona & Tommasi, 2017; Cutkosky & Orabona, 2018) derives parameter-free algorithms for selecting learning rates which are optimal in the noise-dominated (as opposed to curvature-dominated) regime. These algorithms are shown to be practical for training deep neural networks with small batch size (e.g. a convolutional network for CIFAR-10 with batch size 128). The theory presented in this paper is only applicable to large batch/curvature-dominated settings which is the regime, where one might hope to isolate the benefits of acceleration. In small-batch/noise-dominated settings, the precise role of acceleration/learning rate schedules is muddled with confounding factors (e.g. variance reduction); see the next section for a discussion of this point. Designing adaptive algorithms which interpolate between these results and ours, like the analysis of Nesterov's acceleration under additive noise (Lan, 2012), is an interesting direction for future work; we hope that this will lead to new practical algorithms for large-scale settings.

G.4. Acceleration methods and momentum

The phenomenon of acceleration in numerical analysis and optimization is a classical concept which has manifested through a large variety of viewpoints , algorithms, and analyses over the years. We provide a very short and limited summary of these manifestations, focusing on more modern machine learning focused developments. For an in-depth treatment, we strongly recommend the reader to refer the recent monograph (d'Aspremont et al., 2021). Possibly the earliest works on non-linear acceleration in numerical analyses date back to Aitken's Δ^2 (Aitken, 1927), Wynn's epsilon algorithm (Wynn, 1956), and Anderson acceleration (Anderson, 1965) (see (Sidi et al., 1986) for an in-depth survey, or the blogpost (Bach, 2020) for a condensed description). The recent work of (Li & Li, 2020) establishes an optimum rate for an Anderson acceleration method based on Chebyshev polynomials. The more standard suite of acceleration algorithms applied in machine learning arise from the direct acceleration algorithms like Polyak's momentum (also known as the heavy ball algorithm) (Polyak, 1964a) and Nesterov's breakthrough result (Nesterov, 1983) which established the optimal rates for general smooth convex optimization.

More recently, various acceleration algorithms (Allen-Zhu & Orecchia, 2014; Bubeck et al., 2015) have been proposed, with more intuitive analyses than Nesterov's. Another line of work stemming from the work of (Su et al., 2014; Wibisono & Wilson, 2015) derives Nesterov-like methods via discretizations of appropriate continuous-time differential equations. A lesser known (but relevant to our work) version of direct acceleration is Nemirovski's acceleration based on a line search (*not* a search over η_t like the greedy steepest descent method); see (Bubeck, 2019) for a concise exposition. An alternative methodology for acceleration (Monteiro & Svaiter, 2013; Lin et al., 2018) comes about via iteratively solving appropriate (strongly convex) proximal point problems using classical iterative methods. This latter line of work has been influential in deriving optimal accelerated versions of higher-order methods (Nesterov, 2008; Bubeck et al., 2019).

In stochastic and/or non-convex settings (including deep learning), the role of acceleration is not fully clear. In the general convex case, worst case theory (Lan, 2012) suggests that acceleration leads to benefits only in curvature-dominated regime (as opposed to the noise dominated regime). Nevertheless, heavy-ball momentum and Nesterov acceleration are part of the core toolkit in state-of-the-art optimization for optimization in various batch size regimes (Sutskever et al., 2013; Kingma & Ba, 2014; Dozat, 2016). Recent theoretical work (Cutkosky & Orabona, 2019) suggests that momentum can implicitly perform variance reduction (akin to a low-pass filter), leading to improved convergence rates for stochastic optimization in non-convex problems. Understanding the variance-reducing mechanisms of the fractal schedules (or any learning rate schedule in general) is an interesting direction for future work.

In a recent orthogonal line of inquiry into momentum methods, Pedregosa & Scieur (2020); Scieur & Pedregosa (2020) analyze an average case setting of the quadratic model, and establish the universality of Polyak momentum as an optimal algorithm. The analysis of globally-optimized learning rate schedules in average-case settings is an interesting direction for future work.

G.5. Optimization as a dynamical system

Our approach to analyzing stability is most similar to the view of optimization algorithms as dynamical systems (Lessard et al., 2016; Li et al., 2017). Of course, beyond the simplest objectives and noise models, optimization algorithms are nonlinear dynamical systems; thus, theory under this very general abstraction is very limited. Bousquet & Elisseeff (2002) define related but stronger notions of stability, which can lead to generalization properties (Hardt et al., 2016; Chen et al.,

2018; Agarwal et al., 2020b).

In the dynamical systems view, our work shows that stable acceleration is obtained by treating the learning rate schedule as a long-horizon planning problem, accounting for the interactions between the choices of η_t at different times and the global curvature of the loss. Even an open-loop control sequence (i.e. non-adaptive schedule) designed with global objectives has a provable benefit over a naive closed-loop controller (i.e. adaptive line search) which only uses instantaneous feedback (i.e. x_t, g_t). Thus, it may be beneficial for any closed-loop controller for the learning rate schedule to depend on global context or curvature, and possibly make negative local progress. In light of this, neural and reinforcement learning-based optimizer search (Bello et al., 2017) may be an enticing solution to the empirical problem of scheduling the learning rate with awareness of global curvature.