HiLD 2026: 4th Workshop on High-dimensional Learning Dynamics 1–27

A "feature ODE" describing the learning behavior of shallow MLPs on simple functions

Joseph Turnbull* JOEYTURNBULL@BERKELEY.EDU
UC Berkeley, Berkeley, United States

Berkan Ottlik* OTTLIK@SEAS.UPENN.EDU
University of Pennsylvania, Philadelphia, United States

James Simon JSI@BERKELEY.EDU
UC Berkeley, Berkeley, United States
Imbue, San Francisco, United States

Abstract

We study the gradient flow training dynamics of two-layer multi-layer perceptrons (MLPs) on anisotropic Gaussian data. Even in this simple setting, solving the dynamics in full is notoriously difficult, so we try something different. We propose a "feature ODE" that might be intuitively expected to describe the learning dynamics of a shallow MLP. We find that MLPs trained to learn simple target functions (low-order Hermite polynomials and staircase functions) actually roughly follow the loss trajectories from this feature ODE. This agreement is a surprise because of the size of the simplification and the fact that it does not come from a rigorous chain of approximations. We discuss this surprise and speculate on its reasons and uses.

1. Introduction

How do shallow MLPs learn simple functions of anistropic Gaussian data? Many researchers have studied these training dynamics, and we have learned a lot. [1] One definite conclusion has emerged: the problem is hard — surprisingly hard, for what should be a simple toy problem in deep learning theory.

What is the source of the difficulty? Taking a look at the functional form of a shallow MLP,

$$f_{\text{MLP}}(\boldsymbol{x}) = \langle \boldsymbol{w}_{\text{out}}, \sigma\left(\boldsymbol{W}_{\text{in}}\boldsymbol{x}\right)\rangle, \tag{1}$$

the main annoyance is the elementwise nonlinearity $\sigma(\cdot)$. Elementwise nonlinearities do not play well with our nicest mathematical tools: try to exactly solve the dynamics of a model parameterized as Equation (1) using e.g. regular matrix ODEs, and the elementwise nonlinearity (which is rather unnatural from a linear-algebraic point of view) gets in the way.

Rather than develop a new tool for treating the exact dynamics of a shallow MLP, we propose something different altogether. We design our system to be a solvable model of the shallow MLP that keeps two of the MLP's key traits. Firstly, there should be two trainable linear transformations with a (fixed) nonlinear operation in between, and secondly, the model class needs to represent

*Joint primary authorship.
1. See Section A for a discussion of related works.

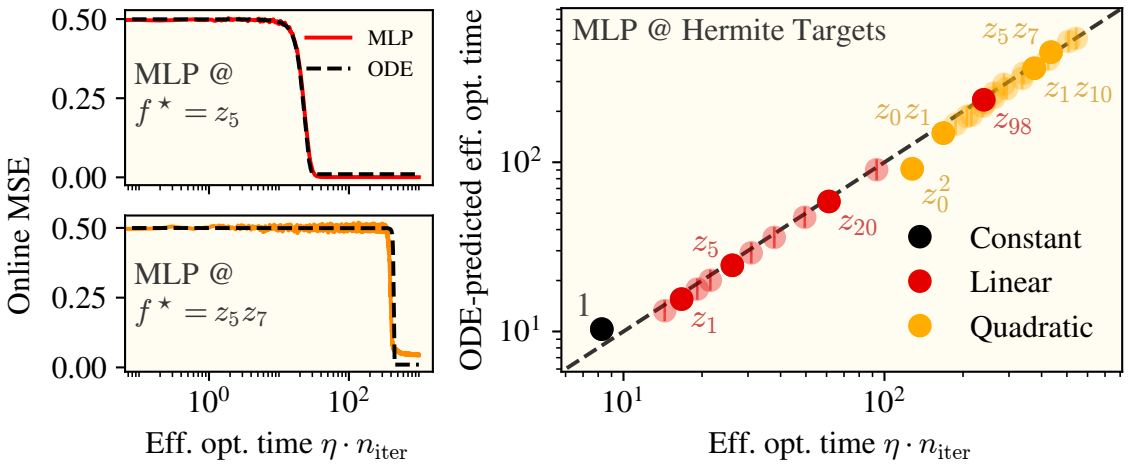

Figure 1: **We suggest a simple ODE of scalar variables whose solution predicts loss trajectories for shallow MLPs learning simple functions of anisotropic data.** These plots treat shallow MLPs with shifted SiLU activation functions ($\sigma(x) = \text{SiLU}(x + 0.5)$) learning various Hermite polynomial target functions data sampled $\boldsymbol{x} \sim \mathcal{N}(0, \boldsymbol{\Gamma})$ with $\boldsymbol{\Gamma} = \text{diag}(\gamma_i)$, $\gamma_i \propto (i+6)^{-1.8}$. We write $z_i \equiv \boldsymbol{x}_i / \sqrt{\gamma_i}$ for normalized coordinates. **Left.** MLP loss trajectories closely match the ODE for example linear and quadratic target functions. **Right.** Effective optimization times $\eta \cdot n_{\text{iter}}$ required for learning to complete are well-predicted by this ODE across many such target functions. For quadratic terms of the form $\text{He}_2(z_i)$, the leading term is indicated in the plots (z_i^2). See Section B for experimental details.

any nonlinear function. We get both of these traits in the following "deep polynomial regression" (DPR) model:

$$f_{\text{DPR}}(\boldsymbol{x}) = \langle \boldsymbol{B}, \boldsymbol{\psi}_{\boldsymbol{c}}(\boldsymbol{A}\boldsymbol{x}) \rangle, \tag{2}$$

where $\boldsymbol{A}$ is a square matrix, $\boldsymbol{B}$ is a readout vector, and $\boldsymbol{\psi}_{\boldsymbol{c}}$ is the following *polynomial featurization* operation:

$$\boldsymbol{\psi}_{\boldsymbol{c}}(\boldsymbol{h}) := \text{vec}\left(\left[\sqrt{c_0} \quad \sqrt{c_1}\,\boldsymbol{h} \quad \frac{\sqrt{c_2}}{\sqrt{2!}}\,\boldsymbol{h}\boldsymbol{h}^\top \quad \frac{\sqrt{c_3}}{\sqrt{3!}}\,\boldsymbol{h}^{\otimes 3} \quad \cdots \right] \right), \tag{3}$$

with $\boldsymbol{c} = (c_\ell)$ a sequence of constants.

Why is this a reasonable proxy for a shallow MLP? The matrix $\boldsymbol{A}$ serves the role of $\boldsymbol{W}_{\text{in}}$, preconditioning the input to the nonlinear operation. The operation $\boldsymbol{\psi}_{\boldsymbol{c}}$ generates all monomial functions of the hidden vector $\boldsymbol{h} = \boldsymbol{A}\boldsymbol{x}$, giving the model full expressivity. Finally, the readout vector $\boldsymbol{B}$ maps these features to a scalar output. See Figure 2 for a visual comparison between the two models.

Note that if one simply fixes $\boldsymbol{A} = \boldsymbol{I}$, then Equation (2) is polynomial regression: $f(\boldsymbol{x}) = \langle \boldsymbol{b}, \boldsymbol{\psi}_{\boldsymbol{c}}(\boldsymbol{x}) \rangle$. Equivalently, it is a kernel machine with kernel $K(\boldsymbol{x}, \boldsymbol{x}') = \sum_\ell \frac{c_\ell}{\ell!} \langle \boldsymbol{x}, \boldsymbol{x}' \rangle^\ell$. With $\boldsymbol{A}$ present, f_{DPR} is a kernel machine with *trainable* kernel $K_{\boldsymbol{A}}(\boldsymbol{x}, \boldsymbol{x}') = \sum_\ell \frac{c_\ell}{\ell!} \langle \boldsymbol{A}\boldsymbol{x}, \boldsymbol{A}\boldsymbol{x}' \rangle$.

The wonderful thing about DPR is that its gradient flow dynamics on squared loss can in many cases be reduced to a simple ODE of scalar variables and solved exactly. The punchline of this paper is that the resulting ODE is often predictive of the loss trajectories and learning times for MLPs learning low-order polynomials (Figures 1 and 3).

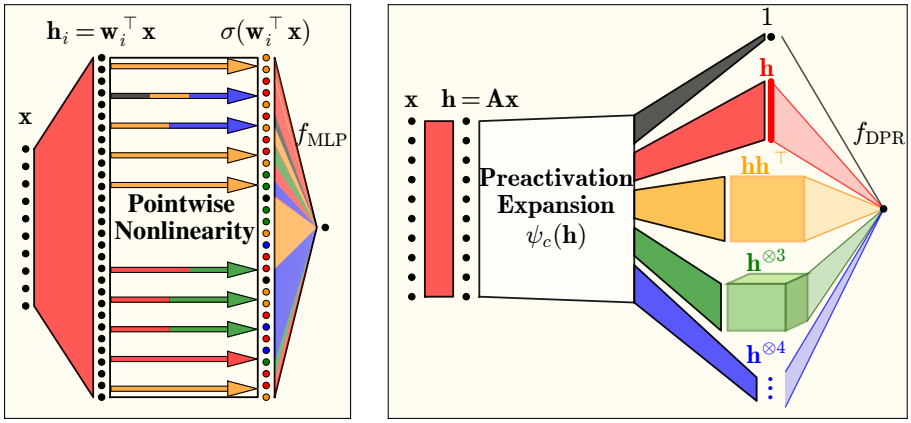

Figure 2: **Comparison between a shallow MLP and deep polynomial regression (DPR). Left.** A shallow MLP has nonlinearities applied elementwise, and each nonlinearity contains many polynomial orders. Different elements tend to different components of the preactivation. **Right.** In DPR, the nonlinear operation is a preactivation expansion which separately generates monomial interactions of the input vector.

2. The feature ODE

We draw data as $\boldsymbol{x} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Gamma})$, with anisotropic covariance $\boldsymbol{\Gamma} = \text{diag}(\gamma_1, \cdots, \gamma_d)$. Given a target function f^*, we construct the square loss $\mathcal{L} = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim\mathcal{N}(0,\boldsymbol{\Gamma})}\left[(f^*(\boldsymbol{x}) - f_{\text{DPR}}(\boldsymbol{x}))^2\right]$. We study gradient flow optimization of the DPR parameters:

$$\dot{\boldsymbol{A}} = -\nabla_{\boldsymbol{A}}\mathcal{L}, \qquad \dot{\boldsymbol{B}} = -\nabla_{\boldsymbol{B}}\mathcal{L}, \qquad \boldsymbol{A}(0) = 1/\sqrt{d}\cdot\boldsymbol{I}, \qquad \boldsymbol{B}(0) = \mathbf{0}. \tag{4}$$

Observe that the model function f_{DPR} can be written out explicitly in polynomial form:

$$f_{\text{DPR}}(\boldsymbol{x}) = \sum_{\ell\geq 0}\sum_{\boldsymbol{\beta}\in[d]^\ell} \frac{\sqrt{c_\ell}}{\sqrt{\ell!}} B_{\boldsymbol{\beta}} \prod_{m=1}^{\ell} (\boldsymbol{A}\boldsymbol{x})_{\beta_m}. \tag{5}$$

The great simplifying blessing is that, when f^* depends on only a handful of the coordinates of $\boldsymbol{x}$, then most of these terms can be neglected. In particular, if f^* depends on only a subset of coordinates $\{x_i\}$ such that $i \in \mathcal{I} \subset [d]$, then any readout parameters $b_{\boldsymbol{\beta}}$ with $\boldsymbol{\beta} \notin \mathcal{I}^d$ will remain identically zero under gradient flow. Furthermore, first-layer parameters A_{ij} with $j \notin \mathcal{I}$ will feel no gradient and remain equal to $\kappa\delta_{ij}$.

We now invoke two assumptions on the data. First, we assume that $\gamma_i \ll 1$.[2] With this assumption, we find that terms of higher orders are learned more slowly, and only the lowest-order terms of f_{DPR} required to represent the target function f^* grow substantially and are relevant to the dynamics. Second, we expand the target function as a sum of Hermite polynomials: $f^*(\boldsymbol{x}) = \sum_{\boldsymbol{\alpha}\in\mathbb{N}_0^d} v_{\boldsymbol{\alpha}} h_{\boldsymbol{\alpha}}(\boldsymbol{\Gamma}^{-1/2}\boldsymbol{x})$, and make a sparseness assumption on the coefficients $v_{\boldsymbol{\alpha}}$ which will be

2. In our typical scaling, $\text{Tr}[\boldsymbol{\Gamma}] = 1$ is the natural normalization, so this amounts to an assumption that no single coordinate gets a substantial fraction of the variance.

satisfied in every case we discuss (the per-order parity criterion, Section C.3). See Section C for full discussion of these assumptions and the consequent derivations.

After some algebra, we are left with the following ODE:

$$\mathcal{L} = \frac{1}{2} \sum_{\boldsymbol{\alpha} \in \mathcal{N}_0^d} \left(v_{\boldsymbol{\alpha}} - \tilde{c}_{\boldsymbol{\alpha}} \cdot b_{\boldsymbol{\alpha}} \prod_i a_i^{\alpha_i} \right)^2 , \tag{6}$$

$$\dot{a}_i = -\partial_{a_i} \mathcal{L}, \qquad \dot{b}_{\boldsymbol{\alpha}} = -\partial_{b_{\boldsymbol{\alpha}}} \mathcal{L} \, , \tag{7}$$

where $\tilde{c}_{\boldsymbol{\alpha}} := \sqrt{c_{|\boldsymbol{\alpha}|}} \cdot \prod_{i=1}^d \gamma_i^{\alpha_i/2}$. We refer to the above as the *feature ODE*. The point of the feature ODE is that (a) when the target function is very sparse (i.e. only a few v_α are nonzero), then we need only track a handful of scalar variables, and (b) the resulting loss dynamics resemble those of a shallow MLP trained on the same target function.

When the target function consists of a single Hermite term $f^*(\boldsymbol{x}) = h_{\boldsymbol{\alpha}}(\boldsymbol{x})$, the loss in the feature ODE simplifies to just $\mathcal{L} = \frac{1}{2}(1 - \tilde{c}_{\boldsymbol{\alpha}} \cdot b \prod_i a_i^{\alpha_i})^2$, and the resulting dynamics of a few variables can be studied nicely. The loss and its dynamics cannot be directly solved, but a number of approximations can be made, and are done so in Section E, such that an explicit solution for the effective optimization time $\tau \equiv \eta \cdot n_{\text{iter}}$ can be found. In short, we define the effective optimization time by having b reach its final value, of $b_f = \tilde{c}_{|\boldsymbol{\alpha}|}^{-\frac{1}{\sum_i \boldsymbol{\alpha}_i + 1}}$:

$$\int_0^\tau \mathrm{d}t = \tau = \int_0^{b_f} \frac{\mathrm{d}b}{\tilde{c}_{|\boldsymbol{\alpha}|} \prod_{i=1}^d \left(\boldsymbol{\alpha}_i b^2(t) + a_i^2(0) \right)^{\frac{\boldsymbol{\alpha}_i}{2}}} \, . \tag{8}$$

We experimentally validate that MLPs learn target functions in an amount of gradient steps defined by Equation (8), finding great agreement with results summarized in the right side of Figure 1.

Example calculations. To concretize our findings, we provide an in-depth discussion on how the feature ODE can be used to analyze an MLP learning a number of targets presented in the paper. Throughout our discussion, we will use normalized coordinates $z_i = x_i / \sqrt{\gamma_i}$ such that $|f^\star| = 1$.

Linear target. On a linear target, the feature ODE and dynamics reduce to the form

$$\mathcal{L} = \frac{1}{2} \left(1 - \tilde{c}_1 a_i b \right)^2 \qquad \dot{a}_i = \left(1 - \tilde{c}_1 a_i b \right) \cdot \tilde{c}_1 b \qquad \dot{b} = \left(1 - \tilde{c}_1 a_i b \right) \cdot \tilde{c}_1 a_i \, .$$

where we would like to pause to appreciate the simplicity of our ODE: describing the learning dynamics can reliably be captured by only *two* dynamical variables! For the remainder of the examples, we leave out the dynamics in a and b as they follow from simple derivatives of the loss.

For the loss trajectory, refer to the top left of Figure 1, with the effective optimization time being

$$\tau = \int_0^{\tilde{c}^{\frac{1}{2}}} \frac{\mathrm{d}b}{\tilde{c}_1 \sqrt{b^2(t) + a_i^2(0)}} \approx \frac{1}{\tilde{c}_1} \ln \left(\frac{2}{\sqrt{\tilde{c}_1} a_i(0)} \right)$$

Quadratic targets. There are two different types of quadratic targets to consider: the first consists of a mixed-index target, $z_i z_j$, and the second is a pure-quadratic target $h_2(z_i)$. We review the pure-quadratic case's effective optimization time, with the mixed-index being developed similarly. We note that the feature ODE can be obtained by similar arguments used for the linear case,[3] We find the dynamics and effective optimization time as,

$$\tau \approx \int_0^\infty \frac{\mathrm{d}b}{\tilde{c}_2 \left(2b^2(t) + a_i^2(0)\right)} = \frac{a_0^{-1}}{\tilde{c}_2} \frac{\pi}{2^{3/2}},$$

The mixed-index case has one small but noticeable change: the effective optimization time has the $2^{3/2}$ in the denominator replaced with a 2. Surprisingly, the repeated-index quadratic is learned with exactly $\sqrt{2}$ times fewer steps than it would take to learn a mixed-index quadratic, all other properties remaining constant (validated in Figure 1)!

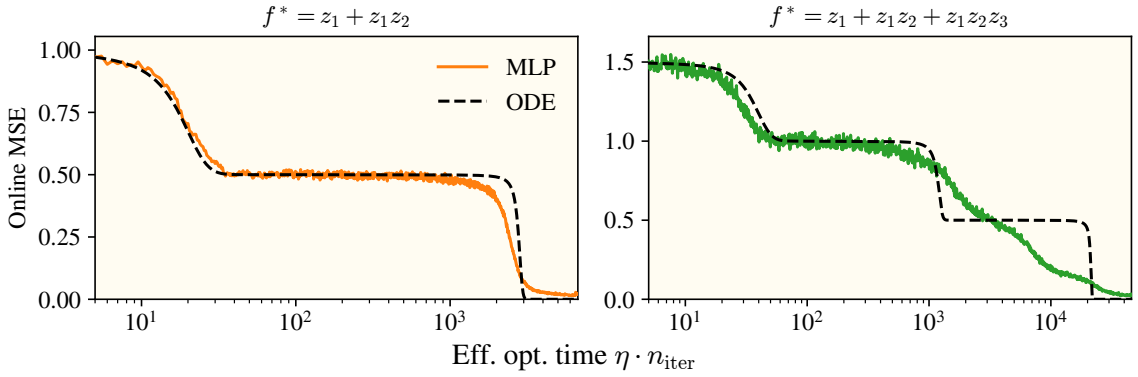

Figure 3: **The feature ODE predicts loss trajectories for shallow MLPs learning low-degree "staircase functions."** Predicted loss trajectories are staircases with a number of steps equal to the order of the target function. True loss curves for shallow MLPs roughly follow these staircases, with steps at roughly the predicted times. See Section B for experimental details.

Staircase with two terms Thus far, we have only discussed target functions with a single term. Notably, this is vastly different from what most real targets consist of. To get closer to real targets, we analyze feature ODE on a "staircase function" (defined in Abbe et al. (2023)) with two terms: $f^\star = z_i + z_i z_j$. The dynamics of the feature ODE are not discussed in detail, however they follow similar arguments to those of the linear and quadratic cases and are instead shown in Figure 3. We note that there is currently no clean expression for the optimization time of the feature ODE on staircase functions, although presume it is possible.

As more terms are added, the feature ODE will undoubtedly become more complex, however we believe the feature ODE will still be a useful tool that can be used to analyze the learning dynamics of the shallow MLP.

3. Note that, strictly speaking, the pure-quadratic ODE lacks a corresponding mode to the constant 1. So long as $\tilde{c}_2 \ll \tilde{c}_0$, the difference in dynamics when this term is excluded compared to included is negligible.

3. Outlook & Conclusion

We are excited about future work utilizing this framework to study feature learning. We believe our framework allows for a principled study of the evolution of the neural feature matrix $\boldsymbol{W}_{\text{in}}^\top \boldsymbol{W}_{\text{in}}$ (introduced in Radhakrishnan et al. (2024)), with some, albeit weaker, evidence for a principled study of the output layer weight vector $\boldsymbol{w}_{\text{out}}$ being possible. We are optimistic we can predict dynamics of MLPs learning more complicated functions in varied hyperparameter regimes by further studying DPR and studying more general simplified ODEs.

References

Emmanuel Abbe, Enric Boix-Adserà, Matthew Brennan, Guy Bresler, and Dheeraj Nagaraj. The staircase property: How hierarchical structure can guide deep learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. URL https://arxiv.org/abs/2108.10573.

Emmanuel Abbe, Enric Boix-Adserà, and Theodor Misiakiewicz. The merged-staircase property: a necessary and nearly sufficient condition for SGD learning of sparse functions on two-layer neural networks. In *Conference on Learning Theory (COLT)*, 2022. URL https://arxiv.org/abs/2202.08658.

Emmanuel Abbe, Enric Boix-Adsera, and Theodor Misiakiewicz. Sgd learning on neural networks: leap complexity and saddle-to-saddle dynamics, 2023. URL https://arxiv.org/abs/2302.11055.

Luca Arnaboldi, Ludovic Stephan, Florent Krzakala, and Bruno Loureiro. From high-dimensional & mean-field dynamics to dimensionless ODEs: A unifying approach to SGD in two-layers networks. In *Conference on Learning Theory (COLT)*, 2023.

Luca Arnaboldi, Yatin Dandi, Florent Krzakala, Bruno Loureiro, Luca Pesce, and Ludovic Stephan. Online learning and information exponents: On the importance of batch size, and time/complexity tradeoffs. *International Conference on Machine Learning (ICML)*, 2024. URL https://arxiv.org/abs/2406.02157.

Gérard Ben Arous, Reza Gheissari, and Aukosh Jagannath. Online stochastic gradient descent on non-convex losses from high-dimensional inference. *Journal of Machine Learning Research*, 22 (106):1–51, 2021. URL https://arxiv.org/abs/2003.10409.

Raphaël Berthier, Andrea Montanari, and Kangjie Zhou. Learning time-scales in two-layers neural networks. *Foundations of Computational Mathematics*, 2024. URL https://arxiv.org/abs/2303.00055.

Alberto Bietti, Joan Bruna, Clayton Sanford, and Min Jae Song. Learning single-index models with shallow neural networks. *Advances in neural information processing systems*, 35:9768–9783, 2022.

Blake Bordelon and Cengiz Pehlevan. Self-consistent dynamical field theory of kernel evolution in wide neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://arxiv.org/abs/2205.09653.

Blake Bordelon and Cengiz Pehlevan. Dynamics of finite width kernel and prediction fluctuations in mean field neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2304.03408.

Etienne Boursier, Loucas Pillaud-Vivien, and Nicolas Flammarion. Gradient flow dynamics of shallow ReLU networks for square loss and orthogonal inputs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Lénaïc Chizat and Francis Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. URL https://arxiv.org/abs/1805.09545.

Elizabeth Collins-Woodfin, Courtney Paquette, Elliot Paquette, and Loucas Pillaud-Vivien. Hitting the high-dimensional notes: An ODE for SGD learning dynamics on GLMs and multi-index models. *Information and Inference: A Journal of the IMA*, 13(4), 2024. URL https://arxiv.org/abs/2308.08977.

Alex Damian, Jason D. Lee, and Mahdi Soltanolkotabi. Neural networks can learn representations with gradient descent. In *Conference on Learning Theory (COLT)*, 2022. URL https://arxiv.org/abs/2206.15144.

Alex Damian, Eshaan Nichani, Loucas Pillaud-Vivien, and Jason D. Lee. Smoothing the landscape boosts the signal for SGD: Optimal sample complexity for learning single index models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2305.10633.

Yatin Dandi, Florent Krzakala, Bruno Loureiro, Luca Pesce, and Ludovic Stephan. How two-layer neural networks learn, one (giant) step at a time. *Journal of Machine Learning Research*, 25(349), 2024a. URL https://arxiv.org/abs/2305.18270.

Yatin Dandi, Luca Pesce, Hugo Cui, Florent Krzakala, Yue M. Lu, and Bruno Loureiro. Repetita iuvant: Data repetition allows SGD to learn high-dimensional multi-index functions. *arXiv preprint*, 2024b. URL https://arxiv.org/abs/2405.15459.

Sebastian Goldt, Madhu Advani, Andrew M. Saxe, Florent Krzakala, and Lenka Zdeborová. Dynamics of stochastic gradient descent for two-layer neural networks in the teacher-student setup. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. URL https://arxiv.org/abs/1906.08632.

Arthur Jacot, François Ged, Berfin Şimşek, Clément Hongler, and Franck Gabriel. Saddle-to-saddle dynamics in deep linear networks: Small initialization training, symmetry, and sparsity. *arXiv preprint*, 2021. URL https://arxiv.org/abs/2106.15933.

Dhruva Karkada, Joseph Turnbull, Yuxi Liu, and James B. Simon. Predicting kernel regression learning curves from only raw data statistics, 2025. URL https://arxiv.org/abs/2510.14878.

Daniel Kunin, Giovanni Luca Marchetti, Feng Chen, Dhruva Karkada, James B. Simon, Michael R. DeWeese, Surya Ganguli, and Nina Miolane. Alternating gradient flows: A theory of feature

learning in two-layer neural networks, 2025. URL https://arxiv.org/abs/2506.06489.

Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671, 2018. URL https://arxiv.org/abs/1804.06561.

Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Generalization error of random feature and kernel methods: hypercontractivity and kernel matrix concentration. *Applied and Computational Harmonic Analysis*, 59, 2022.

Alireza Mousavi-Hosseini, Denny Wu, Taiji Suzuki, and Murat A. Erdoğdu. Gradient-based feature learning under structured data. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Alireza Mousavi-Hosseini, Denny Wu, and Murat A. Erdoğdu. Learning multi-index models with neural networks via mean-field Langevin dynamics. In *International Conference on Learning Representations (ICLR)*, 2025.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability, 2023. URL https://arxiv.org/abs/2301.05217.

Scott Pesme, Loucas Pillaud-Vivien, and Nicolas Flammarion. Implicit bias of SGD for diagonal linear networks: a provable benefit of stochasticity. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Adityanarayanan Radhakrishnan, Daniel Beaglehole, Parthe Pandit, and Mikhail Belkin. Mechanism for feature learning in neural networks and backpropagation-free machine learning models. *Science*, 383(6690):1461–1467, 2024.

Ali Rahimi and Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 3, page 5, 2007.

Maria Refinetti, Sebastian Goldt, Florent Krzakala, and Lenka Zdeborová. Classifying high-dimensional Gaussian mixtures: Where kernel methods fail and neural networks succeed. In *International Conference on Machine Learning (ICML)*, 2021. URL https://arxiv.org/abs/2102.11742.

Grant M. Rotskoff and Eric Vanden-Eijnden. Trainability and accuracy of neural networks: An interacting particle system approach. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

David Saad and Sara A. Solla. Exact solution for on-line learning in multilayer neural networks. *Physical Review Letters*, 74(21):4337–4340, 1995a.

David Saad and Sara A. Solla. On-line learning in soft committee machines. *Physical Review E*, 52 (4):4225–4243, 1995b.

Justin Sirignano and Konstantinos Spiliopoulos. Mean field analysis of neural networks: A central limit theorem. *Stochastic Processes and their Applications*, 130(3):1820–1852, 2020.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Deep kernel learning, 2015. URL https://arxiv.org/abs/1511.02222.

Arie Wortsman and Bruno Loureiro. Kernel ridge regression under power-law data: spectrum and generalization. *arXiv preprint arXiv:2510.04780*, 2025.

Greg Yang and Edward J Hu. Tensor programs iv: Feature learning in infinite-width neural networks. In *International Conference on Machine Learning*, pages 11727–11737. PMLR, 2021.

Appendix A. Related works

Our work intersects with a wide range of preexisting literature, most notably: the analysis of single- and multi-index target functions, saddle-to-saddle dynamics, staircase function learning, mean-field treatments of two layer networks, and kernel learning. We also briefly discuss potential relations to the "neural feature matrix" line of study.

Single- and multi- index models Single- and multi- index models (SIMs, MIMs respectively) have been proposed as the simplest non-trivial targets for shallow MLPs to learn. Therefore, a great deal of literature covers how MLPs are trained on them (Ben Arous et al., 2021; Bietti et al., 2022; Mousavi-Hosseini et al., 2023; Damian et al., 2023, 2022). Similar to our experiments on staircase functions, Berthier et al. (2024) studied the gradient flow trajectory of wide networks in the mean-field regime and found the same multi-timescale plateau-and-drop structure we showcase. Similar to our ODE, Arnaboldi et al. (2023, 2024) derived a dimensionless ODE detailing the batch size an MLP requires to escape a saddle point when trained on a SIM. We occasionally induce a separation of timescales argument to reduce our feature ODE, which is preceeded by Dandi et al. (2024a); Damian et al. (2023); Dandi et al. (2024b); Mousavi-Hosseini et al. (2025) making batch size arguments for when this is valid on MIMs.

Saddle-to-saddle and step-like learning dynamics The empirical observation that gradient flow on small-initialization MLPs proceeds via long plateaus punctuated by rapid drops has by now been documented from many angles: saddle-to-saddle dynamics in deep linear networks (Jacot et al., 2021); plateaus and timescale separation in two-layer SIMs (Berthier et al., 2024); the implicit bias perspective of Pesme et al. (2021); alternating gradient flows (Kunin et al., 2025); and the structured plateaus exhibited by the teacher-student ODEs of Saad and Solla (1995b). Our feature ODE reproduces this plateau structure (Figure 3) without explicitly solving for saddles.

Staircase functions The staircase functions studied in our experiments ($f^\star = z_1$, $z_1 + z_1 z_2$, $z_1 + z_1 z_2 + z_1 z_2 z_3$) come from the line initiated by Abbe et al. (2021, 2022, 2023). Our work differs by analyzing and characterizing the effects of the anisotropy on the learning dynamics. Closely related, the global convergence results of Boursier et al. (2022) for ReLU networks on orthogonal inputs also exhibit a staircase decomposition of the loss.

Mean-field analyses of two-layer networks Our DPR model and feature ODE can be compared against the more rigorous yet less intuitive mean-field literature of (Mei et al., 2018; Chizat and Bach, 2018; Sirignano and Spiliopoulos, 2020; Rotskoff and Vanden-Eijnden, 2018). At a high level, mean-field studies derive a *partial* differential equation for the distribution of parameters in the $N \to \infty$ limit and gives global-convergence results under suitable assumptions. These analyses take a slighly different approach to removing the effect of neuron specialization, taking an infinite-dimensional measure over neurons. Close historical analogs to our ODE can be found from Saad and Solla (1995b,a); Goldt et al. (2019), deriving order-parameter ODEs for teacher-student target functions (similar to our explicit Hermite target decomposition). Refinetti et al. (2021) used analogous ODE reductions for high-dimensional classification problems. Perhaps most similar to our work is the DMFT literature of Bordelon and Pehlevan (2022, 2023) that captures the full evolution of inner-product kernels during feature learning.

Effective ODEs and high-dimensional limit theorems for SGD A separate analytical thread, instead of taking a width limit, takes a high-dimensional input limit and derives rigorous ODE/SDE

descriptions of summary statistics under one-pass SGD. Collins-Woodfin et al. (2024) established that online gradient flow on MIMs with general data covariance, admits an ODE that exactly predicts the risk and other observables. Arnaboldi et al. (2023) gave a parallel result for two-layer networks, providing a dimensionless ODE that unifies mean-field and high-dimensional descriptions.

Kernels, Hermites, and random features Several of the technical ingredients we use—the Hermite decomposition of activations to read off the level coefficients c_ℓ (Section D), the dot-product kernel approximation in the "near-origin" or "on-sphere" regime, and the polynomial truncation of the random-feature kernel are similar to results found in (Karkada et al., 2025; Wortsman and Loureiro, 2025). The semi-frozen network discussed in Section C.1 can be viewed as a random feature approximation to the DPR model; we owe Rahimi and Recht (2007) for their random feature work. Mei et al. (2022) used Hermite expansions to characterize precisely when neural networks beat their associated kernels, providing the polynomial-by-polynomial picture of approximation that our DPR truncation order p inherits.

Neural feature matrix and feature-learning mechanisms We believe DPR is a sound tool to accurately describe the evolution of the neural feature matrix (NFM) $\boldsymbol{W}_{\text{in}}^\top \boldsymbol{W}_{\text{in}}$, introduced in (Radhakrishnan et al., 2024).

Preconditioned kernel learning We are not the first to suggest applying a learned preconditioner to the inputs of a kernel machine. Wilson et al. (2015) first proposed this technique, using it to improve the sample complexity of kernel regression.

Appendix B. Further experimental details

For the MLP, we use μP parameterization Yang and Hu (2021):

$$\boldsymbol{W}_{\text{in}}[n,i] \sim \mathcal{N}(0, 1/d), \quad \boldsymbol{W}_{\text{out}}[n] = \boldsymbol{0}, \quad \eta_{\boldsymbol{W}_{\text{in}}} = \eta\, N, \quad \eta_{\boldsymbol{W}_{\text{out}}} = \eta/N,$$

where $\boldsymbol{W}_{\text{in}} = \mathbb{R}^{N\times d}$, and we set $\boldsymbol{W}_{\text{out}}$ to be the all zero's vector for consistency with the feature ODE (as $N \to \infty$, $\boldsymbol{W}_{\text{out}}$ tends to 0). For the SFN and DPR, we initialize $\boldsymbol{A} = \frac{1}{\sqrt{d}}\boldsymbol{I}_d$ and zero out the readout tensors $\boldsymbol{B}^{(\ell)}$. All three models train with online SGD on streamed Gaussian data.

Figure 1 experimental details. Experiments are performed with data $\boldsymbol{x} \sim \mathcal{N}(0, \Gamma)$, with $\boldsymbol{x} \in \mathbb{R}^{100}$ and $\Gamma = \text{diag}(\gamma_i)$ where $\gamma_i \sim (i+6)^{-1.8}$ normalized such that $\sum_i \gamma_i = 1$. The learning rate is set to $\eta = 10^{-2}$ which we find to be near the gradient flow regime. Our data is streamed online with a batch size $bsz = 4000$—we note that any reasonably large choice for the batch size reliably reproduces our results. Our MLP is setup in the μP regime Yang and Hu (2021), with a width $N = 8192$. Our activation function of choice is a shifted SiLU: $\sigma(x) = \text{SiLU}(x + 0.5)$ which offers the nice properties of (1) being relatively close to the standard practice ReLU, and (2) having a full power series, with the Taylor expansion around $x = 0$ having no coefficients of 0. We make explicit note that the vast majority of biasless nonlinearities are typically not full in their power series. An exponential moving average smoother with constant 0.9 ($\mathcal{L}(t) = 0.9\mathcal{L}(t-1) + 0.1\Delta\mathcal{L}(t)$ is used to smooth over random Gaussian fluctuations in the data. **Loss curves.** Training is done with a set $100,000$ gradient steps, at which point the run terminates. **Effective optimization time predictions.** We train on a number of targets finding great agreement on all linear terms with slight disagreement on quadratic terms with high covariance γ_i. Cubic and higher order target functions are not included as the predicted optimization times were prohibitive. We believe

anisotropic effects that make real MLP optimization times differ than the ODE predicted times are more prevalent for these higher order targets.

Figure 3 experimental details. Both panels train an MLP of width $N = 8192$ online with batch size 10,000 and a single init seed, and overlay the feature ODE prediction. Panel (a) targets the 2-staircase $f^*(\boldsymbol{z}) = z_1 + z_1 z_2$ with $d = 2$, a power-law exponent $\alpha = 10$ yielding $\boldsymbol{\gamma} \approx (2.00 \times 10^{-2}, 1.95 \times 10^{-5})$, learning rate $\eta = 0.5$, and cosine activation $\sigma(z) = \sqrt{2}\cos(z + \pi/4)$. Panel (b) targets the 3-staircase $f^*(\boldsymbol{z}) = z_1 + z_1 z_2 + z_1 z_2 z_3$ with $d = 3$, a power-law exponent $\alpha = 6$ yielding $\boldsymbol{\gamma} \approx (2.95 \times 10^{-2}, 4.61 \times 10^{-4}, 4.05 \times 10^{-5})$, learning rate $\eta = 0.03$, and tanh-half activation. We were only able to see the staircase behavior with a large ratio of data eigenvalues. With larger data eigenvalues and smaller learning rate the feature ODE becomes less predictive.

Figure 4 and Figure 5 experimental details. Both figures share the same configuration: $d = 3$, a power-law exponent $\alpha = 1.5$ yielding $\boldsymbol{\gamma} \approx (3.11 \times 10^{-3}, 1.10 \times 10^{-3}, 5.98 \times 10^{-4})$, the cosine activation $\sigma(z) = \sqrt{2}\cos(z + \pi/4)$ with no biases, learning rate $\eta = 0.2$, batch size 10,000, and a single init seed. The SFN uses width $N = 8192$, and DPR uses truncation order $p = 3$ to accommodate the highest-order target. We sweep across four targets in $\boldsymbol{z}$ coordinates: z_1, $\mathrm{He}_2(z_1)$, $z_1 + z_1 z_2$, and $z_1 + z_1 z_2 + z_1 z_2 z_3$.

Figure 6, Figure 7, and Figure 8 experimental details. Each figure compares the diagonal-$\boldsymbol{A}$ dynamics of an MLP run against an SFN run on the same target. All six runs use width $N = 8192$, batch size 10,000, the cosine activation $\sigma(z) = \sqrt{2}\cos(z + \pi/4)$ with no biases, and a single init seed. The per-run hyperparameters are:

- Figure 6, target $f^*(\boldsymbol{z}) = z_1$: MLP uses $d = 2$ with uniform $\boldsymbol{\gamma} = (2.50 \times 10^{-3}, 2.50 \times 10^{-3})$ and $\eta = 0.1$; SFN uses $d = 3$ with $\alpha = 1.5$ yielding $\boldsymbol{\gamma} \approx (3.11 \times 10^{-3}, 1.10 \times 10^{-3}, 5.98 \times 10^{-4})$ and $\eta = 0.2$.
- Figure 7, target $f^*(\boldsymbol{z}) = \mathrm{He}_2(z_1) + \mathrm{He}_2(z_2)$: MLP and SFN both use $d = 2$ with uniform $\boldsymbol{\gamma} = (2.50 \times 10^{-3}, 2.50 \times 10^{-3})$ and $\eta = 0.1$.
- Figure 8, target $f^*(\boldsymbol{z}) = z_1 + z_1 z_2$: MLP uses $d = 2$ with $\alpha = 10$ yielding $\boldsymbol{\gamma} \approx (2.00 \times 10^{-2}, 1.95 \times 10^{-5})$ and $\eta = 0.5$; SFN uses $d = 3$ with $\alpha = 1.5$ yielding $\boldsymbol{\gamma} \approx (3.11 \times 10^{-3}, 1.10 \times 10^{-3}, 5.98 \times 10^{-4})$ and $\eta = 0.2$.

Appendix C. Detailed derivation of the feature ODE from MLPs

We derive the feature ODE from an MLP with MSE loss in three steps. Rather than conjecturing DPR directly from the MLP, we introduce an intermediate model—the *semi-frozen network* (SFN)—whose infinite-width limit recovers DPR, and we derive the SFN from the MLP by a conjecture on the input layer's weight matrix.

C.1. Further details on MLPs to Semi-Frozen Networks

A one-hidden-layer MLP computes

$$f_{\mathrm{MLP}}(\boldsymbol{x}) = \boldsymbol{W}_{\mathrm{out}}\, \sigma(\boldsymbol{W}_1 \boldsymbol{x}), \qquad \boldsymbol{W}_1 \in \mathbb{R}^{N \times d},$$

where σ acts pointwise. To pass from the MLP to a semi-frozen network, we assume that the hidden neurons form a homogeneous population: no single neuron encodes special information,

and adding or removing any one has a negligible effect on the output. We require this homogeneity to hold throughout training, not just at initialization.

The input layer plays two roles. It rotates the data, upweighting coordinates that matter for the output; and it routes the rotated data into the pointwise nonlinearity, where different neurons see different mixtures. The SFN disentangles these roles by freezing the routing matrix and training only the rotation.

Skinny SVD. Let $\boldsymbol{W}_1 = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^\top$ be the thin SVD, with $\boldsymbol{U} \in \mathbb{R}^{N\times d}$ having orthonormal columns, $\boldsymbol{S} \in \mathbb{R}^{d\times d}$ diagonal, and $\boldsymbol{V} \in \mathbb{R}^{d\times d}$ orthogonal. Defining $\boldsymbol{A} := \boldsymbol{S}\boldsymbol{V}^\top$ and $\boldsymbol{W}_{\text{frozen}} := \boldsymbol{U}$ gives

$$f_{\text{SFN}}(\boldsymbol{x}) = \boldsymbol{W}_{\text{out}}\,\tilde{\sigma}(\boldsymbol{W}_{\text{frozen}}\boldsymbol{A}\boldsymbol{x}),$$

where $\tilde{\sigma}(x) = \sum_\ell c_\ell \ell!^2 x^\ell$, c_ℓ's given by the Taylor series expansion of the MLP's σ (see Section D). $\tilde{\sigma}$ still acts pointwise, but now on the random features $\boldsymbol{W}_{\text{frozen}}\boldsymbol{A}\boldsymbol{x}$. We initialize $\boldsymbol{A} = \frac{1}{\sqrt{d}}\boldsymbol{I}_d$ to remain consistent with μP scaling.

The SFN *freezes* $\boldsymbol{U}$ at its initialization and trains only $\boldsymbol{A}$, reducing the first-layer parameter count from Nd to d^2. In practice, rather than taking $\boldsymbol{U}$ from the SVD of a specific $\boldsymbol{W}_1$, we initialize each entry of $\boldsymbol{W}_{\text{frozen}}$ as iid Gaussian with variance $\sigma^2_{\text{frozen}}/d$ (the standard μP / random-feature scaling) and hold it fixed throughout training. The scale σ_{frozen} is a free hyperparameter that controls the preactivation magnitude; we absorb it into the data eigenvalues γ_i throughout (see the footnote in Section C.2), so it does not appear explicitly in the kernel or the feature ODE. This shift assigns the nonlinearity's expressive role to a fixed random projection. The price is any reorganization of the frozen directions—basis rotations, ICA-style updates—which we discuss in Section F.

Width and concentration. As $N \to \infty$, the SFN's random feature kernel concentrates around its expectation over the frozen weights, which is the SFN kernel we use in Section C.2. At finite N, the SFN's Gram matrix fluctuates by $O(1/\sqrt{N})$ across seeds, so the SFN matches the kernel only up to width corrections.

Neural feature matrix. The MLP carries a "neural feature matrix" NFM $:= \boldsymbol{W}_{\text{in}}^\top \boldsymbol{W}_{\text{in}}$. For an SFN, $\boldsymbol{A}$ plays the analogous role:

$$\boldsymbol{A} \;\sim\; \sqrt{\tfrac{1}{N}\,\boldsymbol{W}_{\text{in}}^\top \boldsymbol{W}_{\text{in}}}.$$

C.2. Further details on Semi-Frozen Networks to DPR

Here, we replace the pointwise activation in the SFN with explicit polynomial features and obtain deep polynomial regression (DPR). At a high level, the SFN's feature kernel admits a polynomial decomposition, and DPR is the explicit-feature model that realizes the same decomposition.

The SFN feature kernel. Because $\boldsymbol{W}_{\text{frozen}}$ is frozen, the random feature kernel holds throughout training, not only at initialization (unlike the NNGP). Taking $N \to \infty$, the random feature average

converges to its expectation over the frozen weights[4] (Rahimi and Recht, 2007):

$$K_{\text{SFN}}(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}_{\boldsymbol{w}\sim\mathcal{N}(0,\boldsymbol{I}_d/d)}\left[\tilde{\sigma}(\boldsymbol{w}^\top \boldsymbol{A}\boldsymbol{x})\, \sigma(\boldsymbol{w}^\top \boldsymbol{A}\boldsymbol{x}')\right].$$

The $1/d$ per-component variance is the standard μP / random-feature scaling that keeps the preactivation $O(1)$ as d grows. The preactivations $\boldsymbol{w}^\top \boldsymbol{A}\boldsymbol{x}$ and $\boldsymbol{w}^\top \boldsymbol{A}\boldsymbol{x}'$ are jointly Gaussian, so the kernel reduces to a two-dimensional Gaussian expectation:

$$K_{\text{SFN}}(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}_{z_1,z_2\sim\mathcal{N}(0,\boldsymbol{\Sigma}')}[\tilde{\sigma}(z_1)\, \tilde{\sigma}(z_2)], \quad \boldsymbol{\Sigma}' = \frac{1}{d}\begin{pmatrix} \|\boldsymbol{A}\boldsymbol{x}\|^2 & \boldsymbol{x}^\top \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{x}' \\ \boldsymbol{x}'^\top \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{x} & \|\boldsymbol{A}\boldsymbol{x}'\|^2 \end{pmatrix}.$$

The kernel therefore depends on three scalars: the two norms $\|\boldsymbol{A}\boldsymbol{x}\|^2$, $\|\boldsymbol{A}\boldsymbol{x}'\|^2$ and the dot product $\boldsymbol{x}^\top \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{x}'$.

Dot-product approximation. The kernel depends on all three scalars in general, but reduces to a function of the dot product alone in two regimes. First, when $\|\boldsymbol{A}\boldsymbol{x}\|^2$ concentrates across data points (the "on-sphere" regime), the diagonal entries factor out as a data-independent prefactor. Second, when $\|\boldsymbol{A}\boldsymbol{x}\|^2/d$ stays small for all data points (the "near-origin" regime), the diagonal entries sit near zero and the kernel ignores their variation entirely. The near-origin regime corresponds to the wide-kernel-theory limit; the on-sphere regime is weaker, allowing larger norms but requiring them to be approximately equal. Small data eigenvalues place us in the near-origin regime, since $\mathbb{E}_{\boldsymbol{x}}\left[\|\boldsymbol{A}\boldsymbol{x}\|^2/d\right] = (1/d)\sum_j a_j^2\gamma_j$. In both cases, the kernel reduces to

$$K_{\text{SFN}}(\boldsymbol{x}, \boldsymbol{x}') \approx k_{\text{SFN}}\left(\tfrac{1}{d}\, \boldsymbol{x}^\top \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{x}'\right).$$

Polynomial decomposition. Taylor-expanding k_{SFN} gives

$$K_{\text{SFN}}(\boldsymbol{x}, \boldsymbol{x}') \approx \sum_{\ell=0}^{\infty} c_\ell \ell! \left(\tfrac{1}{d}\, \boldsymbol{x}^\top \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{x}'\right)^\ell \equiv K_{\text{DPR}}(\boldsymbol{x}, \boldsymbol{x}'),$$

where the $(1/d)^\ell$ factors are absorbed into c_ℓ. Applying Mercer's theorem with eigenfunctions $\phi \sim \sqrt{c_\ell \ell!}\,(\boldsymbol{A}\boldsymbol{x})^{\otimes\ell}$ yields the DPR model:

$$f_{\text{DPR}}(\boldsymbol{x}) = \sum_{\ell=0}^{p} \sqrt{c_\ell \ell!}\, \boldsymbol{B}^{(\ell)} (\boldsymbol{A}\boldsymbol{x})^{\otimes\ell},$$

with trainable symmetric order-ℓ tensors $\boldsymbol{B}^{(\ell)}$ and an order-p truncation.

Truncation error. DPR drops all terms $\ell > p$ in the expansion above. The order-ℓ kernel contribution is $(c_\ell \ell!)\,(\frac{1}{d}\, \boldsymbol{x}^\top \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{x}')^\ell$, which decays with ℓ when $\|\boldsymbol{A}\boldsymbol{x}\|^2/d$ stays small. Taking the expectation over data $\boldsymbol{x} \sim \mathcal{N}(0, \boldsymbol{\Gamma})$ gives $\mathbb{E}_{\boldsymbol{x}}\left[\|\boldsymbol{A}\boldsymbol{x}\|^2/d\right] = (1/d)\sum_j a_j^2\gamma_j$, so smaller data eigenvalues and larger p both tighten the approximation.

4. Three equivalent choices set the preactivation scale and produce identical dynamics: (a) rescale the data eigenvalues $\gamma_i \to \beta^2\gamma_i$, breaking the trace normalization $\sum_i \gamma_i = 1$; (b) shift to higher-indexed input coordinates of the same power-law spectrum, which yields smaller eigenvalues while preserving the trace; (c) multiply the preactivation by β inside the model itself, equivalent to writing a σ_{frozen} factor inside the activation. Throughout the appendix and the experimental details (Section B) we absorb any preactivation scale into the data eigenvalues γ_i, so the kernel below carries no σ_{frozen} factor.

C.3. Further details on DPR to feature ODE

We simplify DPR's gradient flow ODE to a system of scalar variables in $\boldsymbol{A}$ and the target-aligned entries of $\boldsymbol{B}$, recovering the feature ODE defined in the main text. The reduced state consists of $\boldsymbol{A}$'s diagonal entries $\boldsymbol{a} = (a_1, \ldots, a_d)$ together with one scalar $b_{\boldsymbol{\alpha}}$ per target Hermite mode—the common value of $\boldsymbol{B}^{(|\boldsymbol{\alpha}|)}$ on permutations of $\boldsymbol{\alpha}$.

The reduction rests on the same two assumptions the main text invokes (Section 2): the data eigenvalues are small, which separates the timescales of different polynomial orders; and the target's nonzero Hermite coefficients $v_{\boldsymbol{\alpha}}$ obey a per-order parity criterion, which keeps $\boldsymbol{A}$ diagonal under gradient flow.

The reduced equations. Under both assumptions, only the target-aligned $\boldsymbol{B}$ entries drive the dynamics, and we replace each permutation-symmetric block $\boldsymbol{B}^{(|\boldsymbol{\alpha}|)}$ on multi-index $\boldsymbol{\alpha}$ by a single scalar $b_{\boldsymbol{\alpha}}$. Expanding f_{DPR} around diagonal $\boldsymbol{A}$ and computing the Hermite coefficient at each target mode $\boldsymbol{\alpha}$ yields, after some algebra,

$$\mathbb{E}_{\boldsymbol{x}}\Big[f_{\mathrm{DPR}}(\boldsymbol{x})\, h_{\boldsymbol{\alpha}}(\boldsymbol{\Gamma}^{-1/2}\boldsymbol{x})\Big] = \tilde{c}_{\boldsymbol{\alpha}}\, b_{\boldsymbol{\alpha}} \prod_j a_j^{\alpha_j}, \qquad \tilde{c}_{\boldsymbol{\alpha}} := \sqrt{c_{|\boldsymbol{\alpha}|}} \prod_j \gamma_j^{\alpha_j/2} |\boldsymbol{\alpha}|!,$$

matching the main-text definition (Section 2). The residual at mode $\boldsymbol{\alpha}$ is $e_{\boldsymbol{\alpha}} := v_{\boldsymbol{\alpha}} - \tilde{c}_{\boldsymbol{\alpha}}\, b_{\boldsymbol{\alpha}} \prod_j a_j^{\alpha_j}$, and the population gradient flow produces the feature ODE:

$$\mathcal{L} = \tfrac{1}{2} \sum_{\boldsymbol{\alpha}} e_{\boldsymbol{\alpha}}^2,$$

$$\dot{b}_{\boldsymbol{\alpha}} = e_{\boldsymbol{\alpha}}\, \tilde{c}_{\boldsymbol{\alpha}} \prod_j a_j^{\alpha_j}, \qquad \dot{a}_j = \sum_{\boldsymbol{\alpha}:\, \alpha_j \geq 1} e_{\boldsymbol{\alpha}}\, \tilde{c}_{\boldsymbol{\alpha}}\, \alpha_j\, \frac{\prod_k a_k^{\alpha_k}}{a_j}\, b_{\boldsymbol{\alpha}}.$$

The sum runs over target multi-indices, since $v_{\boldsymbol{\alpha}} = 0$ kills the residual elsewhere.

Parity criterion for diagonal $\boldsymbol{A}$. The off-diagonal A_{ij} gradient takes the form $\mathbb{E}_{\boldsymbol{x}}\Big[(f - f^*)\, \boldsymbol{x}^{\boldsymbol{\alpha}'}\Big]$, with $\boldsymbol{\alpha}'$ shifted by one in coordinates i, j. Gaussian parity makes this expectation vanish unless every entry of $\boldsymbol{\alpha}'$ is even, which in turn forces two target multi-indices at the same order to differ in parity at exactly i, j. So $\boldsymbol{A}$ stays diagonal whenever no two same-order target multi-indices differ in parity at exactly two coordinates—the *per-order parity criterion*. Staircases (z_1, $z_1 z_2$, $z_1 z_2 z_3$, ...) and single Hermite modes ($\mathrm{He}_\ell(z_1)$) satisfy this criterion, and we restrict to such targets throughout the paper. Targets that fail include $z_1 + z_2$ and $\mathrm{He}_3(z_1) + \mathrm{He}_3(z_2)$, where two same-order multi-indices differ in exactly two coordinates and seed an off-diagonal gradient.

Timescale separation. The order-ℓ Hermite mode's gradient carries a factor of order $\gamma^{\boldsymbol{\alpha}}$, so higher-order modes relax much more slowly than lower-order ones whenever the data eigenvalues are small. Lower-order Hermite modes therefore settle long before higher-order modes have moved, and off-target $\boldsymbol{B}$ entries don't backreact on the target dynamics at leading order in the eigenvalues. We can therefore analyze each target multi-index independently—the property the reduced ODE above relies on.

C.4. Validation of the approximation chain

We validate the two links of the approximation chain on synthetic Gaussian data: the SFN-to-feature-ODE reduction (Figure 4) and the DPR-to-feature-ODE reduction (Figure 5). In both cases, the feature ODE is solved with the same hyperparameters as the trained network. We compare online MSE across four targets of increasing complexity: a linear monomial $f^* = z_1$, a single Hermite mode $f^* = \mathrm{He}_2(z_1)$, a two-term staircase $f^* = z_1 + z_1 z_2$, and a three-term staircase $f^* = z_1 + z_1 z_2 + z_1 z_2 z_3$. See Section B for the experimental setup.

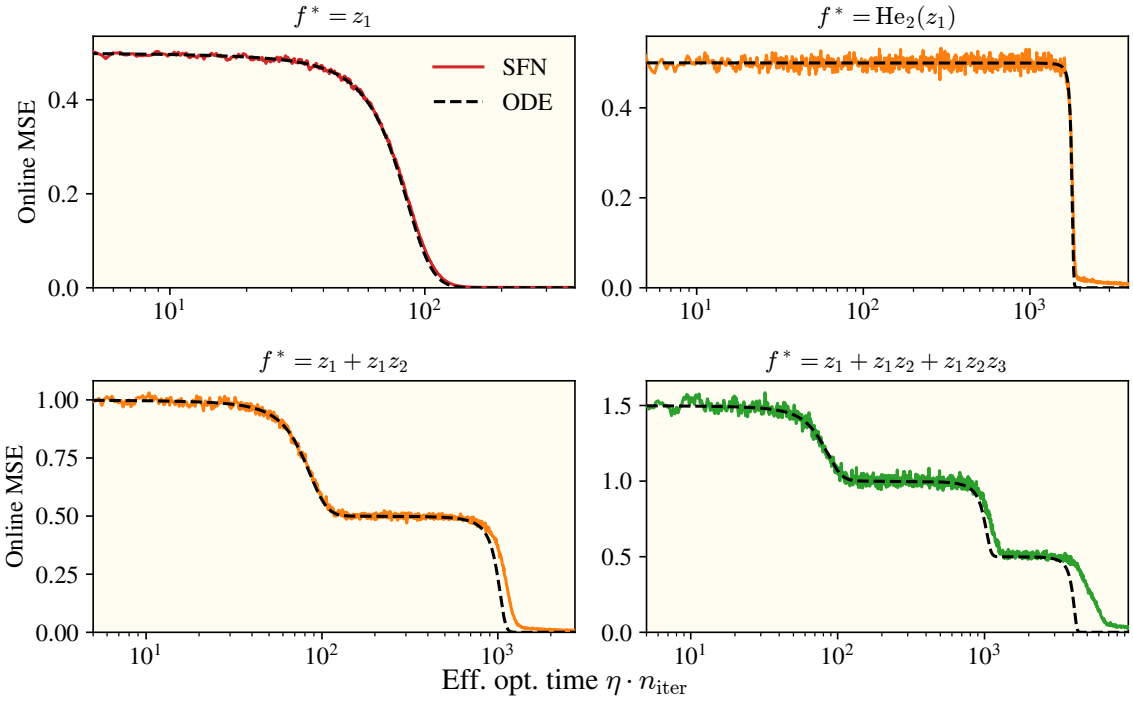

Figure 4: **SFN tracks the feature ODE across all four targets.** Online MSE for an SFN (colored) and the feature ODE (dashed black) over effective optimization time. The ODE recovers both the plateau lengths and the loss drops for each target, including the staircase structure of the two- and three-term staircase targets.

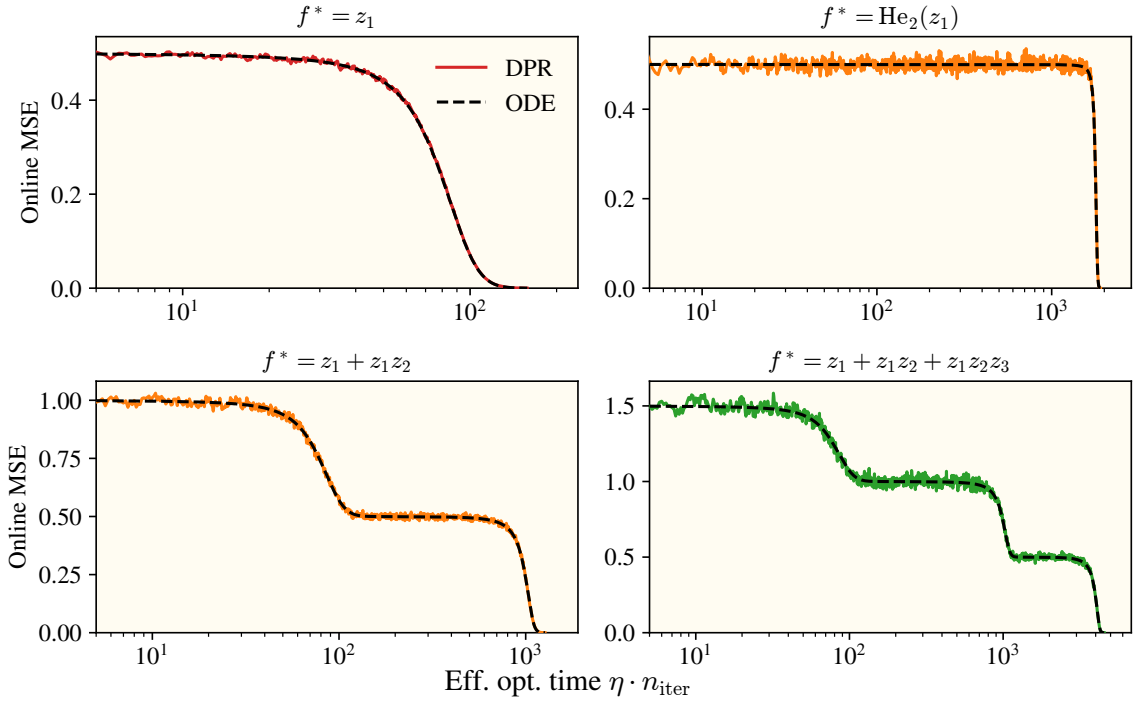

Figure 5: **DPR tracks the feature ODE across all four targets.** Same as Figure 4 but with DPR replacing the SFN. The agreement is tighter than for the SFN, consistent with DPR's explicit polynomial features removing the random-feature concentration error.

Appendix D. Level coefficients for general activations

D.1. General formula

For a general activation σ, with data on a sphere $\|\boldsymbol{A}\boldsymbol{x}\| = r$, the level coefficients c_ℓ of the dot-product kernel can be computed from the Hermite decomposition of the rescaled activation:

$$c_\ell = \frac{\ell!}{r^{2\ell}} \left(\mathbb{E}_{z\sim\mathcal{N}(0,1)} \left[h_\ell(z)\, \sigma\!\left(\tfrac{r}{\sqrt{d}} \cdot z \right) \right] \right)^2, \tag{9}$$

where $h_\ell(z) = \frac{1}{\sqrt{\ell!}} \mathrm{He}_\ell(z)$ is the ℓ-th normalized probabilist's Hermite polynomial. In words, we extract the ℓ-th Hermite coefficient of the function $z \mapsto \sigma(rz/\sqrt{d})$, where the prefactor $r/\sqrt{d}$ is the standard deviation of the preactivations.

The activation's symmetry constrains which orders are present: odd activations (e.g. tanh) have $c_{2\ell} = 0$; activations that are linear plus even (e.g. ReLU) have $c_\ell = 0$ for $\ell = 3, 5, 7, \ldots$. For the reason, we often use either a shifted activation function (e.g. $\mathrm{SiLU}(x + 0.5)$, $\tanh(x + 0.5)$) or a cosine activation function ($\sigma(z) = \sqrt{2}\cos(z + \pi/4)$), all of which are full in their Taylor series.

Approximation. While we are in the near-origin regime (discussed in Section C.2), we find that the level coefficients are found from taking it's Taylor series expansion:

$$c_\ell = \left(\sigma^{(\ell)}(0) \right)^2, \tag{10}$$

the exponent (ℓ) in parentheses denoting the ℓ-th derivative. Equation (10) gives the level coefficients used in our validation experiments.

D.2. Cosine activation

Here, we derive the dot-product kernel for the special case of a cosine activation $\sigma(z) = \sqrt{2}\cos(z + \pi/4)$ and show that it gives $c_\ell = 1$ for all ℓ.

The SFN feature kernel is

$$K(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}_{\boldsymbol{w}\sim\mathcal{N}(0,\boldsymbol{I}_d/d)}\Big[\sigma(\boldsymbol{w}^\top \boldsymbol{A}\boldsymbol{x})\,\sigma(\boldsymbol{w}^\top \boldsymbol{A}\boldsymbol{x}')\Big]\,.$$

The preactivations $z_1 = \boldsymbol{w}^\top \boldsymbol{A}\boldsymbol{x}$ and $z_2 = \boldsymbol{w}^\top \boldsymbol{A}\boldsymbol{x}'$ are jointly Gaussian with covariance

$$\Sigma' = \frac{1}{d}\begin{pmatrix} \|\boldsymbol{A}\boldsymbol{x}\|^2 & \boldsymbol{x}^\top \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{x}' \\ \boldsymbol{x}'^\top \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{x} & \|\boldsymbol{A}\boldsymbol{x}'\|^2 \end{pmatrix}.$$

Write $s_{11} = \frac{1}{d}\|\boldsymbol{A}\boldsymbol{x}\|^2$, $s_{22} = \frac{1}{d}\|\boldsymbol{A}\boldsymbol{x}'\|^2$, and $s_{12} = \frac{1}{d}\,\boldsymbol{x}^\top \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{x}'$.

Expanding the activation:

$$\begin{aligned} K &= 2\,\mathbb{E}[\cos(z_1 + \pi/4)\cos(z_2 + \pi/4)] && (11)\\ &= \mathbb{E}[\cos(z_1 - z_2)] + \mathbb{E}[\cos(z_1 + z_2 + \pi/2)]\,. && (12)\end{aligned}$$

For the first term, $z_1 - z_2 \sim \mathcal{N}(0, s_{11} - 2s_{12} + s_{22})$. Using the Gaussian identity $\mathbb{E}_{u\sim\mathcal{N}(0,\tau^2)}[\cos(u)] = e^{-\tau^2/2}$:

$$\mathbb{E}[\cos(z_1 - z_2)] = \exp\Big(-\tfrac{1}{2}(s_{11} - 2s_{12} + s_{22})\Big)\,.$$

For the second term, $z_1 + z_2 \sim \mathcal{N}(0, s_{11} + 2s_{12} + s_{22})$, and $\cos(z_1 + z_2 + \pi/2) = -\sin(z_1 + z_2)$. Since $\mathbb{E}_{u\sim\mathcal{N}(0,\tau^2)}[\sin(u)] = 0$ by symmetry, this term vanishes. Therefore:

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp\Big(-\tfrac{1}{2}(s_{11} - 2s_{12} + s_{22})\Big) = \exp\Big(-\tfrac{1}{2d}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}\boldsymbol{x}'\|^2\Big)\,.$$

This is an RBF kernel in the transformed coordinates $\boldsymbol{A}\boldsymbol{x}$. Factoring:

$$K(\boldsymbol{x}, \boldsymbol{x}') = \underbrace{\exp\Big(-\tfrac{1}{2}(s_{11} + s_{22})\Big)}_{\text{length-dependent prefactor}} \cdot \underbrace{\exp(s_{12})}_{\text{dot-product term}}\,.$$

Under the dot-product kernel approximation (length terms approximately constant, see Section C.2), the kernel is proportional to

$$\exp(s_{12}) = \exp\Big(\tfrac{1}{d}\,\boldsymbol{x}^\top \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{x}'\Big) = \sum_{\ell=0}^{\infty} \frac{1}{\ell!}\Big(\tfrac{1}{d}\,\boldsymbol{x}^\top \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{x}'\Big)^\ell\,.$$

Comparing with the general form $k(t) = \sum_\ell c_\ell\, t^\ell/\ell!$, we read off $c_\ell = 1$ for all ℓ.

Appendix E. Feature ODE Rise Time Derivations

This appendix discusses our methodology for arriving at the effective optimization time for MLPs to learn monomial target functions. Beginning with the feature ODE, we have

$$\begin{aligned}
\mathcal{L} &= \frac{1}{2}\left(1 - \tilde{c}_\ell \prod_{i=1}^{d} a_i^{\alpha_i} b\right)^2 \\
\dot{a}_i &= -\frac{\partial \mathcal{L}}{\partial a_i} = \left(1 - \tilde{c}_\ell \prod_{i=1}^{d} a_i^{\alpha_i} b\right) \frac{\alpha_i \tilde{c}_\ell \prod_{i=1}^{d} a_i^{\alpha_i - 1} b}{a_i} \\
\dot{b} &= -\frac{\partial \mathcal{L}}{\partial b} = \left(1 - \tilde{c}_\ell \prod_{i=1}^{d} a_i^{\alpha_i} b\right) \frac{\tilde{c}_\ell \prod_{i=1}^{d} a_i^{\alpha_i} b}{b} \, .
\end{aligned}$$

Noting $\frac{1}{\alpha_i} a_i \dot{a}_i = b\dot{b}$, we find $a_i(t) = \sqrt{\alpha_i b^2(t) + a_i^2(0)}$ Plugging this into the feature ODE, we get

$$\dot{b} = \left(1 - \tilde{c}_\ell \prod_{i=1}^{d} \left(\boldsymbol{\alpha}_i b^2(t) + a_i^2(0)\right)^{\frac{\boldsymbol{\alpha}_i}{2}} b\right) \tilde{c}_\ell \prod_{i=1}^{d} \left(\boldsymbol{\alpha}_i b^2(t) + a_i^2(0)\right)^{\frac{\boldsymbol{\alpha}_i}{2}} \, .$$

This equation is not generally solvable, but we can make a number of simplifications to estimate an effective optimization time. In a gradient flow regime from small initialization, the network output tends to follow two distinct regimes: near a saddle-point and growth [find ref]. We assume that the number of gradient steps to escape from a saddle-point will dominate over the number it takes to reach the next saddle-point, forcing $\left(1 - \tilde{c}_\ell \prod_{i=1}^{d} \left(\boldsymbol{\alpha}_i b^2(t) + a_i^2(0)\right)^{\frac{\boldsymbol{\alpha}_i}{2}} b\right) \approx 1$, which simplifies the ODE to

$$\int_0^{b_f} \frac{\mathrm{d}b}{\tilde{c}_\ell \prod_{i=1}^{d} \left(\boldsymbol{\alpha}_i b^2(t) + a_i^2(0)\right)^{\frac{\boldsymbol{\alpha}_i}{2}}} = \int_0^{\tau} \mathrm{d}t \, ,$$

where

$$b_f \equiv \tilde{c}^{\frac{1}{\sum_i \boldsymbol{\alpha}_i + 1}} \tag{13}$$

represents the final value of b and τ represents the learning-rate scaled number of gradient steps. We provide exact results of this integral for the cases of $\boldsymbol{\alpha}_i = 0 \; \forall \; i$ and $\boldsymbol{\alpha}_i = 1, \; \boldsymbol{\alpha}_j = 0 \; \forall \; j \neq i$ (a constant and linear target). For higher order integrals, exact results exist, although to high precision

we can replace the upper bound b_{eq} with infinity, remaining up to a $\mathcal{O}\left(c^{-\frac{2}{\sum_i \alpha_i+1}}\right)$ constant.

$$\tau_{const} = \int_0^{\frac{1}{\tilde{c}}} \frac{\mathrm{d}b}{\tilde{c}} = \frac{1}{\tilde{c}^2} \tag{14}$$

$$\tau_{lin} = \int_0^{\frac{1}{\sqrt{\tilde{c}}}} \frac{\mathrm{d}b}{\tilde{c}\sqrt{b^2 + a_i^2(0)}} = \frac{1}{\tilde{c}} \ln\left(\frac{\frac{1}{\sqrt{\tilde{c}}} + \sqrt{\frac{1}{\tilde{c}} + a_i^2(0)}}{a_i(0)}\right) \approx \frac{1}{\tilde{c}} \ln\left(\frac{2}{\sqrt{\tilde{c}} a_i(0)}\right) \tag{15}$$

$$\tau_{quad+} = \int_0^{\tilde{c}^{-\frac{1}{(\sum_i \alpha_i+1)}}} \frac{\mathrm{d}b}{\tilde{c}_\ell \prod_{i=1}^d \left(\alpha_i b^2(t) + a_i^2(0)\right)^{\frac{\alpha_i}{2}}} \tag{16}$$

$$= \int_0^{\infty} \frac{\mathrm{d}b}{\tilde{c}_\ell \prod_{i=1}^d \left(\alpha_i b^2(t) + a_i^2(0)\right)^{\frac{\alpha_i}{2}}} - \mathcal{O}\left(\tilde{c}^{-\frac{2}{\sum_i \alpha_i+1}}\right) \tag{17}$$

Below, we provide a table of the rise times found with the ∞ upper bound. For shorthand, any α_i that is unspecified takes a value of 0, and $\alpha_{i,j,k,\ldots} = a$ denotes that $\alpha_i = \alpha_j = \alpha_k = \cdots = a$. Some integrals (ie $\alpha_i = 3$, $\alpha_j = 1$) are incomplete elliptics—in such cases, we provide a numerical estimate. We drop the subscript of $\tilde{c}$ as it should be clear from the context.

- **Quadratics** $\begin{cases} \alpha_i = 2 & \tau = \frac{a_0^{-1}}{\tilde{c}} \frac{\pi}{2^{3/2}} \\ \alpha_{i,j} = 1 & \tau = \frac{a_0^{-1}}{\tilde{c}} \frac{\pi}{2} \end{cases}$
- **Cubics** $\begin{cases} \alpha_i = 3 & \tau = \frac{a_0^{-2}}{\tilde{c}} \frac{1}{\sqrt{3}} \\ \alpha_i = 2, \alpha_j = 1 & \tau = \frac{a_0^{-2}}{\tilde{c}} \frac{\pi}{4} \\ \alpha_{i,j,k} = 1 & \tau = \frac{a_0^{-2}}{\tilde{c}} (1) \end{cases}$
- **Quartics** $\begin{cases} \alpha_i = 4 & \tau = \frac{a_0^{-3/2}}{\tilde{c}} \frac{\pi}{8} \\ \alpha_i = 3, \alpha_j = 1 & \tau = \frac{a_0^{-3/2}}{\tilde{c}} (0.4623) \\ \alpha_{i,j} = 2 & \tau = \frac{a_0^{-3/2}}{\tilde{c}} \frac{\pi\sqrt{2}}{8} \\ \alpha_i = 2, \alpha_{j,k} = 1 & \tau = \frac{a_0^{-3/2}}{\tilde{c}} \frac{\pi}{2} (\sqrt{2} - 1) \\ \alpha_{i,j,k,l} = 1 & \tau = \frac{a_0^{-3/2}}{\tilde{c}} \frac{\pi}{4} \end{cases}$
- **Quintics** $\begin{cases} \alpha_i = 5 & \tau = \frac{a_0^{-4/2}}{\tilde{c}} \frac{2}{3\sqrt{5}} \\ \alpha_i = 4, \alpha_j = 1 & \tau = \frac{a_0^{-4/2}}{\tilde{c}} \left(\frac{\pi\sqrt{3}}{27} + \frac{1}{6}\right) \\ \alpha_i = 3, \alpha_j = 2 & \tau = \frac{a_0^{-4/2}}{\tilde{c}} (\sqrt{3} - \ln(2 + \sqrt{3})) \\ \alpha_i = 3, \alpha_{j,k} = 1 & \tau = \frac{a_0^{-4/2}}{\tilde{c}} \left(\frac{\sqrt{3}}{2} - \frac{\sqrt{2}}{8} \ln(5 + 2\sqrt{6})\right) \\ \alpha_{i,j} = 2, \alpha_k = 1 & \tau = \frac{a_0^{-4/2}}{\tilde{c}} (0.5) \\ \alpha_i = 2, \alpha_{j,k,l} = 1 & \tau = \frac{a_0^{-4/2}}{\tilde{c}} (\frac{\pi}{2} - 1) \\ \alpha_{i,j,k,l,m} = 1 & \tau = \frac{a_0^{-4/2}}{\tilde{c}} \frac{2}{3} \end{cases}$

- **Sextics**
$$\begin{cases} \alpha_i = 6 & \tau = \frac{a_0^{-5/2}}{\tilde{c}} \frac{3\pi}{16\sqrt{6}} \\ \alpha_i = 4, \alpha_j = 2 & \tau = \frac{a_0^{-5/2}}{\tilde{c}} \frac{\pi(\sqrt{2}-1)}{4} \\ \alpha_i = 4, \alpha_{j,k} = 1 & \tau = \frac{a_0^{-5/2}}{\tilde{c}} \frac{\pi}{9} \\ \alpha_{i,j} = 3 & \tau = \frac{a_0^{-5/2}}{\tilde{c}} \frac{3\pi}{16\sqrt{3}} \\ \alpha_{i,j,k} = 2 & \tau = \frac{a_0^{-5/2}}{\tilde{c}} \frac{3\pi}{16\sqrt{2}} \\ \alpha_{i,j} = 2, \alpha_{k,l} = 1 & \tau = \frac{a_0^{-5/2}}{\tilde{c}} \frac{\pi(2-\sqrt{2})}{4} \\ \alpha_i = 2, \alpha_{j,k,l,m} = 1 & \tau = \frac{a_0^{-5/2}}{\tilde{c}} \pi(\sqrt{2} - 1.25) \\ \alpha_{i,j,k,l,m,n} = 1 & \tau = \frac{a_0^{-5/2}}{\tilde{c}} \frac{3\pi}{16} \end{cases}$$

Appendix F. Neuron specialization and target functions

We diagnose neuron specialization with a three-panel comparison: training loss, per-direction $\boldsymbol{A}_{ii}$ or $\sqrt{\mathbf{NFM}_{ii}}$ trajectories, and a heatmap of per-neuron preactivation variance (all $N = 8000$ neurons ranked by variance, plotted against effective optimization time). The heatmap is the workhorse. Neurons all contribute to the dynamics when there are consistent color changes across neurons at a given time step and there are smooth color gradients. Neuron subpopulations drive the dynamics when groups of neurons change color independently of the rest. All experiments in this appendix use a biasless network with the cosine activation $\sigma(z) = \sqrt{2}\cos(z + \pi/4)$.

DPR has no neurons — its $\boldsymbol{B}^{(\ell)}$ tensors act on all features at once — so specialization cannot arise. The frozen weights in the SFN make neuron specialization difficult and we see more little specialized behavior in Figures 6 to 8. The feature ODE in turn predicts the loss and per-direction A_{ii} trajectories well.

MLP neurons can specialize. On tasks like modular arithmetic, for instance, small subpopulations align with Fourier frequencies of the input (Kunin et al., 2025; Nanda et al., 2023). On the simpler targets we study, whether the MLP specializes depends on the target itself. On a linear target $f^*(\boldsymbol{z}) = z_1$ (Figure 6), neither the MLP nor the SFN specializes much: a linear target produces a per-neuron gradient with no direction-dependent amplification, so all neurons evolve symmetrically. Their heatmaps are rather uniform, and the feature ODE matches loss and $\sqrt{\mathrm{NFM}_{ii}}$ throughout training. On a pure-Hermite target $f^*(\boldsymbol{z}) = \mathrm{He}_2(z_1) + \mathrm{He}_2(z_2)$ (Figure 7), the MLP specializes sharply: roughly 500 neurons at the top of the ranking drive their preactivation variance orders of magnitude above the bulk. We think this specialized subpopulation drives the MLP loss down earlier than the feature ODE predicts and pins $\sqrt{\mathrm{NFM}_{ii}}$ below the ODE-predicted A_{ii}. On a two-term staircase $f^*(\boldsymbol{z}) = z_1 + z_1 z_2$ (Figure 8), the MLP specializes only mildly — faint bands rather than a dominant subpopulation. The feature ODE captures the staircase structure of the loss and features, with only a small late-time drift in $\sqrt{\mathrm{NFM}_{2,2}}$.

We leave a more detailed study hyperparameter and target function's impact on neuron specialization to future work.

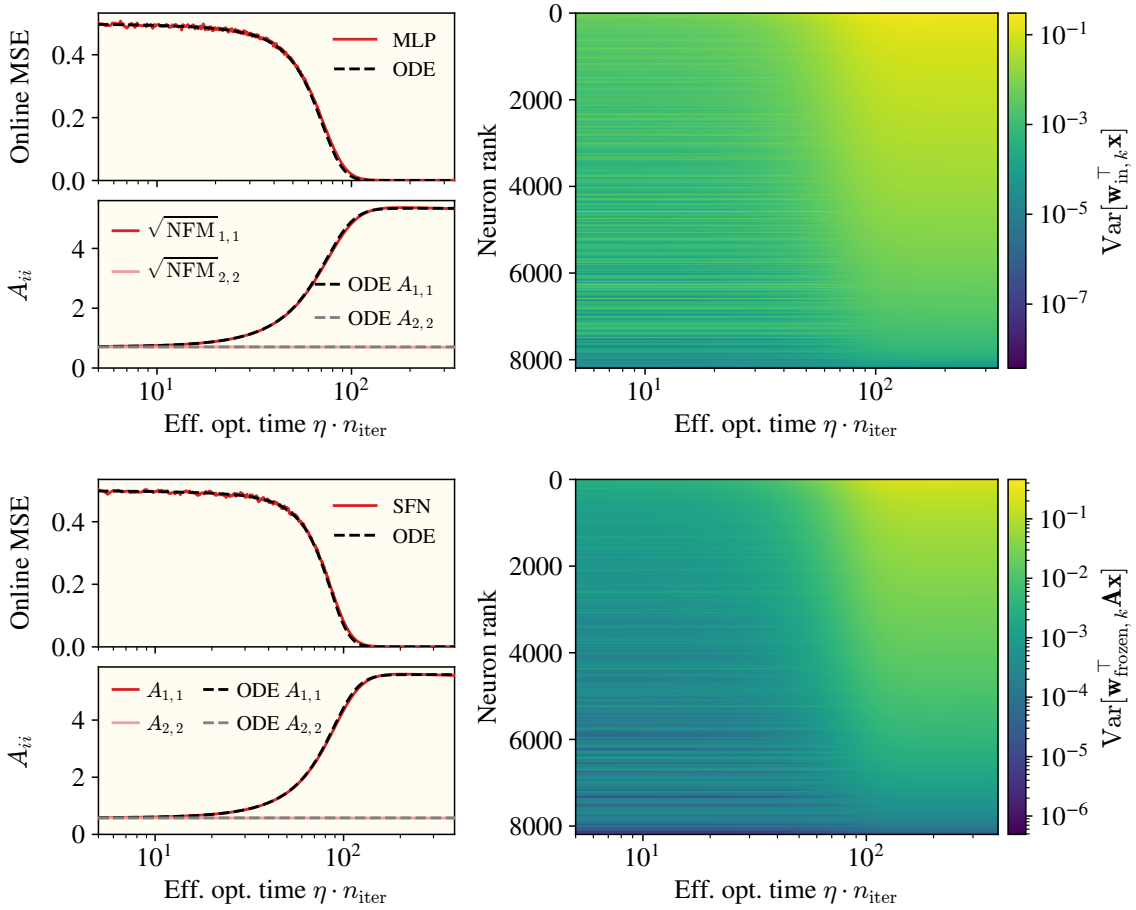

Figure 6: **Linear target $f^*(\boldsymbol{z}) = z_1$.** *Top:* MLP. *Bottom:* SFN. Neurons are sorted by preactivation variance at the end of training in the heatmaps. Loss, $\sqrt{\text{NFM}_{ii}}$ (MLP) or A_{ii} (SFN), and per-neuron preactivation variance. Both models track the feature ODE, and every neuron gains variance on the same timescale.

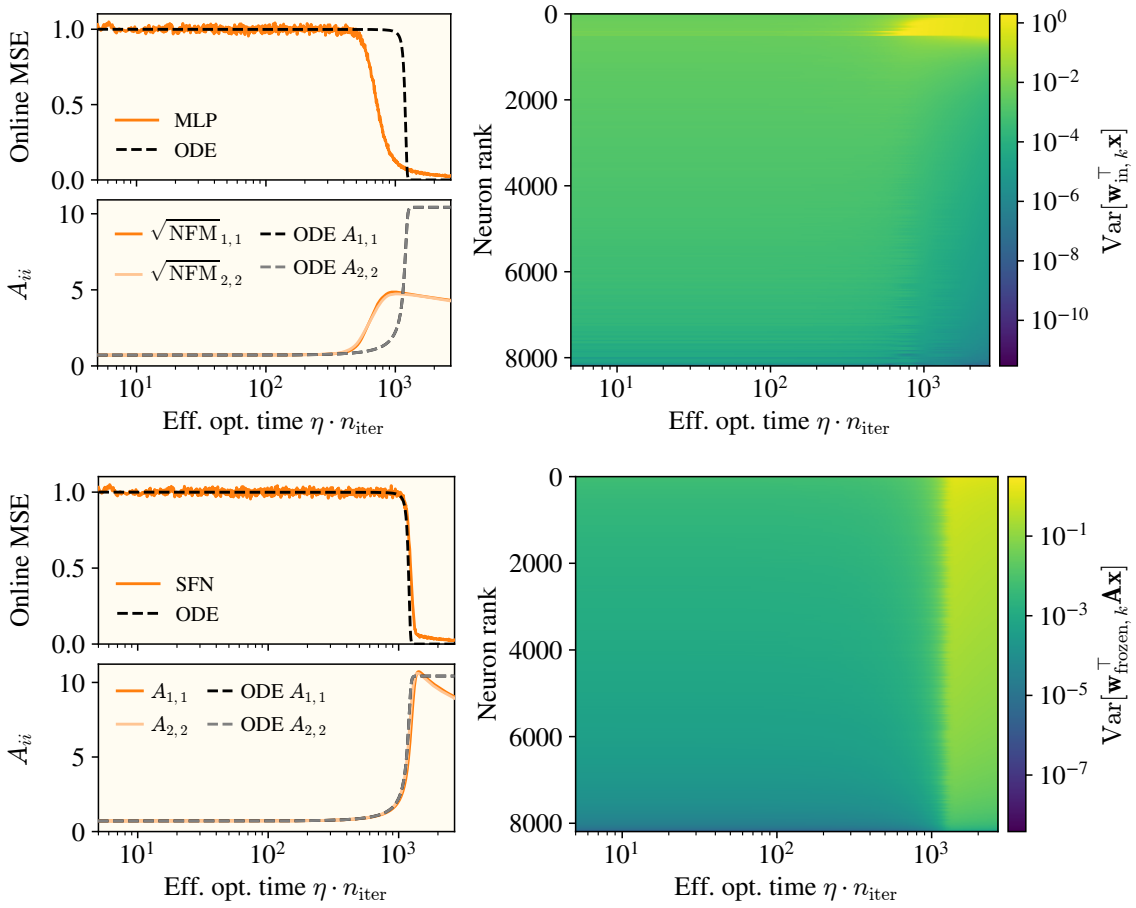

Figure 7: **Pure-Hermite target** $f^*(\boldsymbol{z}) = \text{He}_2(z_1) + \text{He}_2(z_2)$**.** *Top:* MLP. *Bottom:* SFN. Neurons are sorted by preactivation variance at the end of training in the heatmaps. A specialized subpopulation of roughly 500 MLP neurons (top, right column) gains preactivation variance orders of magnitude above the bulk, drives the loss down earlier than the feature ODE predicts, and pins $\sqrt{\text{NFM}_{ii}}$ below the ODE-predicted A_{ii}. The SFN carries no such subpopulation and tracks the ODE.

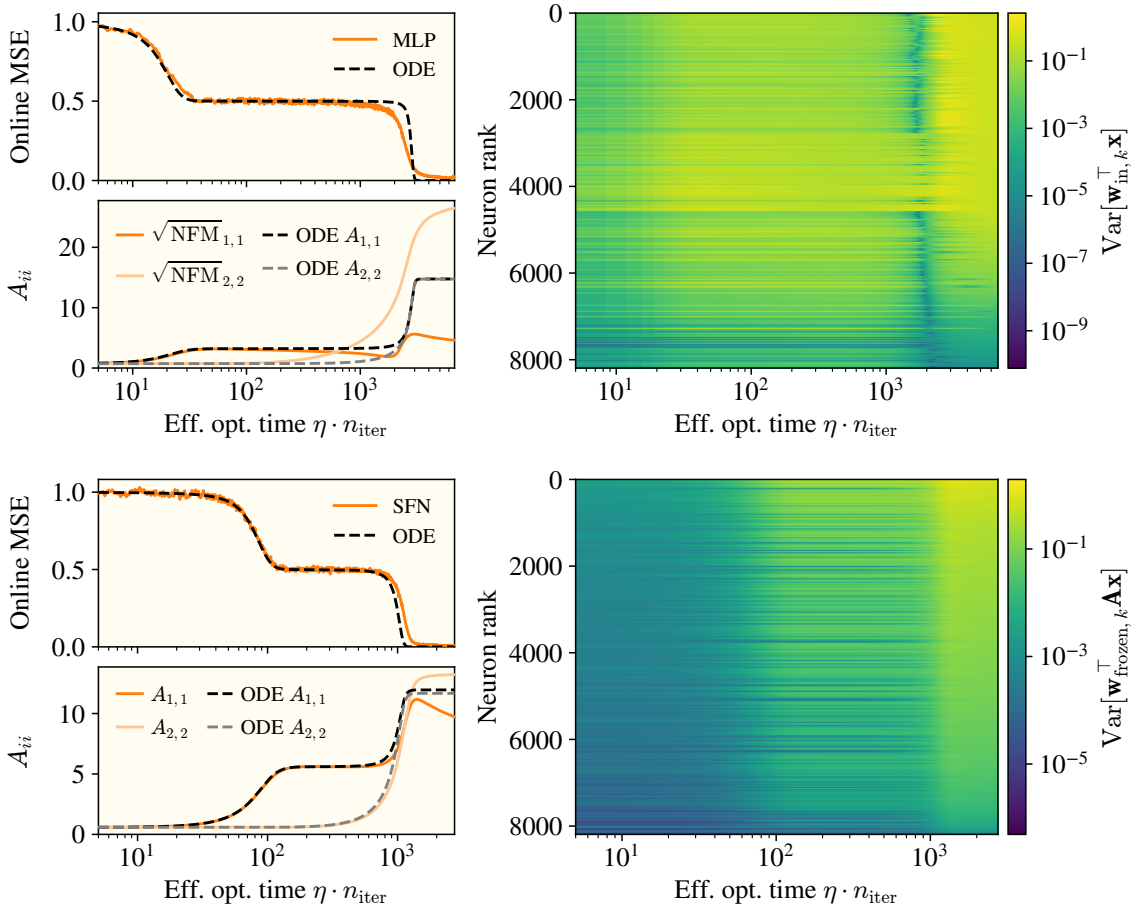

Figure 8: **Two-term staircase** $f^*(\boldsymbol{z}) = z_1 + z_1 z_2$**.** *Top:* MLP. *Bottom:* SFN. Neurons are sorted by preactivation variance at the end of training in the heatmaps. A faint specialization band in the MLP heatmap accompanies a small late-time drift of $\sqrt{\text{NFM}_{2,2}}$ above the ODE prediction. The staircase loss is otherwise well captured. The SFN matches the ODE throughout.

Appendix G. DPR population gradient

This appendix derives the DPR population gradient in closed form, using Isserlis' theorem (Wick contractions) to evaluate the Gaussian expectations. The main text (Section 2) states the resulting gradient flow for some specific target functions. Here, we provide the gradient of DPR for any polynomial target function.

Let $\boldsymbol{a}_m \in \mathbb{R}^d$ denote the m-th row of $\boldsymbol{A}$ written as a column vector, so that $(\boldsymbol{A}\boldsymbol{x})_m = \boldsymbol{a}_m^\top \boldsymbol{x}$, and let $\boldsymbol{e}_\nu \in \mathbb{R}^d$ denote the ν-th standard basis vector. Working in the normalized coordinates $\boldsymbol{z} = \boldsymbol{\Gamma}^{-1/2}\boldsymbol{x}$ (so that $\boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I}_d)$ when $\boldsymbol{x} \sim \mathcal{N}(0, \boldsymbol{\Gamma})$), we expand the target polynomially as $f^\star(\boldsymbol{z}) = \sum_{j=0}^{p} \boldsymbol{F}^{(j)}\, \boldsymbol{z}^{\otimes j}$ with $\boldsymbol{F}^{(j)}$ a symmetric order-j tensor.

The main text (Section 2) writes the target in Hermite basis, $f^\star(\boldsymbol{z}) = \sum_{\boldsymbol{\alpha}} v_{\boldsymbol{\alpha}} h_{\boldsymbol{\alpha}}(\boldsymbol{z})$, with coefficients $v_{\boldsymbol{\alpha}}$ indexed by multi-index. The monomial form used here is equivalent: standard monomial-to-

Hermite expansion identifies $\boldsymbol{F}^{(j)}$ with a linear combination of $v_{\boldsymbol{\alpha}}$ at $|\boldsymbol{\alpha}| \leq j$. We use the monomial representation because Isserlis' theorem evaluates Gaussian moments of monomial products directly. All other symbols ($\boldsymbol{A}$, $\boldsymbol{B}^{(\ell)}$, $\boldsymbol{a}_m$, $\boldsymbol{\Gamma}$, γ_i, c_ℓ) follow the main-text conventions.

G.1. Derivatives of the DPR loss

We take the DPR model

$$f_{\text{DPR}}(\boldsymbol{x}) = \sum_{\ell=0}^{p} \sqrt{\frac{c_\ell}{\ell!}}\, \boldsymbol{B}^{(\ell)} (\boldsymbol{A}\boldsymbol{x})^{\otimes \ell}, \tag{18}$$

and expand its squared loss

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{N}(0,\boldsymbol{\Gamma})}\Big[\big(f_{\text{DPR}}(\boldsymbol{x}) - f^\star(\boldsymbol{z})\big)^2\Big] = \mathbb{E}\Big[f_{\text{DPR}}^2\Big] - 2\,\mathbb{E}[f_{\text{DPR}}\, f^\star] + \mathbb{E}\Big[(f^\star)^2\Big].$$

The last term is parameter-free and drops out of the gradient. The first two split into a model-model piece (which is quadratic in $\boldsymbol{B}$) and a model-target cross piece (which is linear in $\boldsymbol{B}$); we compute the parameter derivatives of f_{DPR} before taking expectations.

$\boldsymbol{B}$ gradient. At entry $(k_1, \ldots, k_\ell)$ of $\boldsymbol{B}^{(\ell)}$, the derivative of $f_{\text{DPR}}(\boldsymbol{x})$ is the matching monomial in $\boldsymbol{A}\boldsymbol{x}$:

$$\frac{\partial f_{\text{DPR}}(\boldsymbol{x})}{\partial \boldsymbol{B}^{(\ell)}_{k_1,\ldots,k_\ell}} = \sqrt{\tfrac{c_\ell}{\ell!}}\, (\boldsymbol{A}\boldsymbol{x})_{k_1} \cdots (\boldsymbol{A}\boldsymbol{x})_{k_\ell}. \tag{19}$$

For the constant term ($\ell = 0$), $\boldsymbol{B}^{(0)}$ is a scalar and $\partial_{\boldsymbol{B}^{(0)}} f_{\text{DPR}} = \sqrt{c_0}$.

$\boldsymbol{A}$ gradient. An increment in $\boldsymbol{A}_{\mu\nu}$ shifts $(\boldsymbol{A}\boldsymbol{x})_\mu$ by x_ν, so each order-ℓ term contributes ℓ identical copies, one per tensor slot:

$$\frac{\partial f_{\text{DPR}}(\boldsymbol{x})}{\partial \boldsymbol{A}_{\mu\nu}} = x_\nu \sum_{\ell=1}^{p} \ell \sqrt{\tfrac{c_\ell}{\ell!}}\, \boldsymbol{B}^{(\ell)}_{\mu,\, b_2,\ldots,b_\ell}\, (\boldsymbol{A}\boldsymbol{x})_{b_2} \cdots (\boldsymbol{A}\boldsymbol{x})_{b_\ell}, \tag{20}$$

where repeated b-indices are summed. The constant term ($\ell = 0$) does not depend on $\boldsymbol{A}$ and contributes nothing.

G.2. Expectations via Isserlis' theorem

Each piece of the loss gradient is the expectation of a polynomial in the jointly Gaussian vectors $\boldsymbol{x} \sim \mathcal{N}(0, \boldsymbol{\Gamma})$ and $\boldsymbol{z} = \boldsymbol{\Gamma}^{-1/2}\boldsymbol{x}$. Isserlis' theorem states that for jointly Gaussian zero-mean random variables $X_1, \ldots, X_{2n}$,

$$\mathbb{E}[X_1 X_2 \cdots X_{2n}] = \sum_{\text{pairings } \pi} \prod_{(m,n)\in\pi} \mathbb{E}[X_m X_n]\,, \tag{21}$$

where the sum runs over the $(2n-1)!!$ partitions of $\{1, \ldots, 2n\}$ into unordered pairs; odd-order moments vanish. Three types of slots appear in our expressions — transformed slots $(\boldsymbol{A}\boldsymbol{x})_k$, raw slots x_ν (from the $\boldsymbol{A}$ derivative), and target slots z_f. Assigning each slot a vector $\boldsymbol{u} \in \mathbb{R}^d$,

$$\boldsymbol{u}_k = \boldsymbol{a}_k \;\; \text{(transformed)}, \qquad \boldsymbol{u}_\nu = \boldsymbol{e}_\nu \;\; \text{(raw)}, \qquad \boldsymbol{u}_f = \boldsymbol{\Gamma}^{-1/2}\boldsymbol{e}_f \;\; (\boldsymbol{z}\text{-target}), \tag{22}$$

every pairwise covariance takes the unified form $\mathrm{Cov}[\,\cdot\,,\,\cdot\,] = \boldsymbol{u}_m^\top \boldsymbol{\Gamma}\boldsymbol{u}_n$. This evaluates to $\boldsymbol{a}_m^\top\boldsymbol{\Gamma}\boldsymbol{a}_n$ between two transformed slots, $\sqrt{\gamma_f}(\boldsymbol{a}_m)_f$ between a transformed slot and a $\boldsymbol{z}$-slot, $\delta_{ff'}$ between two $\boldsymbol{z}$-slots, and analogously for the raw slot.

$\boldsymbol{B}$ expectations. For the model-model piece, the order-ℓ contribution at index $(k_1, \dots, k_\ell)$ is

$$2\sqrt{\tfrac{c_\ell}{\ell!}}\,\mathbb{E}[(\boldsymbol{A}\boldsymbol{x})_{k_1}\cdots(\boldsymbol{A}\boldsymbol{x})_{k_\ell}\, f_{\mathrm{DPR}}(\boldsymbol{x})] = 2\sqrt{\tfrac{c_\ell}{\ell!}}\sum_{i=0}^{p}\sqrt{\tfrac{c_i}{i!}}\,\boldsymbol{B}^{(i)}_{b_1,\dots,b_i}\sum_{k\text{-}b\text{ pairings}}\prod_{(m,n)}\boldsymbol{a}_m^\top\boldsymbol{\Gamma}\boldsymbol{a}_n, \tag{23}$$

where "k-b pairings" enumerates all unordered pair partitions of the $\ell+i$ indices $\{k_1,\dots,k_\ell,b_1,\dots,b_i\}$. The pairing sum vanishes whenever $\ell+i$ is odd. The model-target cross piece is analogous,

$$-2\sqrt{\tfrac{c_\ell}{\ell!}}\sum_{j=0}^{p}\boldsymbol{F}^{(j)}_{f_1,\dots,f_j}\sum_{k\text{-}f\text{ pairings}}\prod_{(m,n)}\boldsymbol{u}_m^\top\boldsymbol{\Gamma}\boldsymbol{u}_n, \tag{24}$$

with $\boldsymbol{u}_m=\boldsymbol{a}_m$ on k-slots and $\boldsymbol{u}_m=\boldsymbol{\Gamma}^{-1/2}\boldsymbol{e}_m$ on f-slots ($\boldsymbol{z}$-coordinates of the target).

$\boldsymbol{A}$ expectations. The same Wick reduction applies, except one slot is now "frozen" to the raw coordinate $x_\nu=\boldsymbol{e}_\nu^\top\boldsymbol{x}$ from the $\boldsymbol{A}$ derivative:

$$\mathbb{E}[\nabla_{\boldsymbol{A}}\mathcal{L}]_{\mu,\nu} = 2\sum_{i=1}^{p} i\sqrt{\tfrac{c_i}{i!}}\,\boldsymbol{B}^{(i)}_{\mu,b_2,\dots,b_i}\Bigg[\sum_{j=1}^{p}\sqrt{\tfrac{c_j}{j!}}\,\boldsymbol{B}^{(j)}_{c_1,\dots,c_j}\sum_{\nu\text{-}b\text{-}c\text{ pairings}}\prod_{(m,n)}\boldsymbol{u}_m^\top\boldsymbol{\Gamma}\boldsymbol{u}_n - \sum_{j=1}^{p}\boldsymbol{F}^{(j)}_{f_1,\dots,f_j}\sum_{\nu\text{-}b\text{-}f\text{ pairings}}\prod_{(m,n)}\boldsymbol{u}_m^\top\boldsymbol{\Gamma}\boldsymbol{u}_n\Bigg], \tag{25}$$

where $\boldsymbol{u}$ takes value $\boldsymbol{e}_\nu$ on the ν-slot, $\boldsymbol{a}_b$ on b- and c-slots, and $\boldsymbol{\Gamma}^{-1/2}\boldsymbol{e}_f$ on f-slots.

G.3. Full population gradient

The pieces combine into two equivalent expressions for the full population gradient.

Pairing form. Collecting the model-model and model-target reductions,

$$\nabla_{\boldsymbol{B}^{(\ell)}_{k_1,\dots,k_\ell}}\mathcal{L} = 2\sqrt{\tfrac{c_\ell}{\ell!}}\sum_{i=0}^{p}\sqrt{\tfrac{c_i}{i!}}\,\boldsymbol{B}^{(i)}_{b_1,\dots,b_i}\sum_{k\text{-}b\text{ pairings}}\prod_{(m,n)}\boldsymbol{a}_m^\top\boldsymbol{\Gamma}\boldsymbol{a}_n - 2\sqrt{\tfrac{c_\ell}{\ell!}}\sum_{j=0}^{p}\boldsymbol{F}^{(j)}_{f_1,\dots,f_j}\sum_{k\text{-}f\text{ pairings}}\prod_{(m,n)}\boldsymbol{u}_m^\top\boldsymbol{\Gamma}\boldsymbol{u}_n, \tag{26}$$

and the $\boldsymbol{A}$ gradient is given by the expression above.

Moment-tensor form. Define $\boldsymbol{S}=\boldsymbol{A}\boldsymbol{\Gamma}\boldsymbol{A}^\top$ and two Gaussian moment tensors:

$$\boldsymbol{M}^{(\ell)}_{k_1,\dots,k_\ell} = \mathbb{E}[(\boldsymbol{A}\boldsymbol{x})_{k_1}\cdots(\boldsymbol{A}\boldsymbol{x})_{k_\ell}] = \sum_{\text{pairings}}\prod_{(m,n)}\boldsymbol{S}_{m,n}, \tag{27}$$

$$\boldsymbol{M}^{(r;t)}_{s_1,\dots,s_r;\,q_1,\dots,q_t} = \mathbb{E}[z_{s_1}\cdots z_{s_r}\,(\boldsymbol{A}\boldsymbol{x})_{q_1}\cdots(\boldsymbol{A}\boldsymbol{x})_{q_t}], \tag{28}$$

where $\boldsymbol{M}^{(r;t)}$ contracts r $\boldsymbol{z}$-target slots against t transformed slots through Gaussian pairings. Using $x_\nu=\sqrt{\gamma_\nu}\,z_\nu$, the raw-slot factor in the $\boldsymbol{A}$ derivative pulls a $\sqrt{\gamma_\nu}$ outside the moment, leaving

pure $\boldsymbol{z}$-and-transformed moments. The gradient then takes the compact contraction form

$$\nabla_{\boldsymbol{B}^{(\ell)}_{k_1,\ldots,k_\ell}} \mathcal{L} = 2\sqrt{\tfrac{c_\ell}{\ell!}} \Bigg[\sum_{i=0}^{p} \sqrt{\tfrac{c_i}{i!}}\, \boldsymbol{B}^{(i)}_{b_1,\ldots,b_i}\, \boldsymbol{M}^{(\ell+i)}_{k_1,\ldots,k_\ell,\, b_1,\ldots,b_i} - \sum_{j=0}^{p} \boldsymbol{F}^{(j)}_{f_1,\ldots,f_j}\, \boldsymbol{M}^{(j;\,\ell)}_{f_1,\ldots,f_j;\, k_1,\ldots,k_\ell} \Bigg], \tag{29}$$

$$\nabla_{\boldsymbol{A}_{\mu,\nu}} \mathcal{L} = 2\sqrt{\gamma_\nu} \sum_{i=1}^{p} i\sqrt{\tfrac{c_i}{i!}}\, \boldsymbol{B}^{(i)}_{\mu,b_2,\ldots,b_i} \Bigg[\sum_{j=1}^{p} \sqrt{\tfrac{c_j}{j!}}\, \boldsymbol{B}^{(j)}_{c_1,\ldots,c_j}\, \boldsymbol{M}^{(1;\,i+j-1)}_{\nu;\, b_2,\ldots,b_i,\, c_1,\ldots,c_j} - \sum_{j=1}^{p} \boldsymbol{F}^{(j)}_{f_1,\ldots,f_j}\, \boldsymbol{M}^{(j+1;\,i-1)}_{\nu,f_1,\ldots,f_j;\, b_2,\ldots,b_i} \Bigg]. \tag{30}$$

The pairing form is more explicit and easier to truncate to a single order; the moment-tensor form makes the contraction structure transparent. Both feed directly into the diagonal-$\boldsymbol{A}$ and timescale-separation reductions used in Section C.3.