Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

The Dynamics of Spectral Bias in Operator Learning

Conrad Ainslie
University of California, Santa Cruz
Ching-Yao Lai
Stanford University
Ashesh Chattopadhyay
aschatto@ucsc.edu

University of California, Santa Cruz https://orcid.org/0000-0002-2590-1230

Research Article

Keywords: Spectral bias, PDEs, neural operators, neural tangent

Posted Date: November 13th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11713020/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

The Dynamics of Spectral Bias in Operator Learning

Conrad Ainslie
Department of Applied Mathematics,
University of California, Santa Cruz, USA

Ching-Yao Lai*
Department of Geophysics and the Institute for Computational
and Mathematical Engineering, Stanford University, USA

Ashesh Chattopadhyay†
Department of Applied Mathematics,
University of California, Santa Cruz, USA

Abstract

Spectral bias is the phenomenon where neural networks learn the low-wavenumber content of a target before its high-wavenumber content. We analyze the training dynamics of neural operators on the discrete Poisson problem and find that the bias is a property of the eigenbasis of the kernel that governs the training update, not of the wavenumber basis in which it is usually reported. A model learning a discretized operator retains its error in the eigendirections of this kernel with small eigenvalues, which are the rates at which training contracts the error along those directions. Under smooth forcing, low variance is placed at high wavenumbers, and any model, linear or nonlinear, has small eigenvalues along the eigenvectors with high-wavenumber content. Under white forcing, the linear models learn the operator, while the nonlinear models retain error in slow directions that depend on the coupling between training samples in the kernel; for the GELU dense network they disappear entirely when that coupling is removed. Modes that a model cannot represent at all contribute zero eigenvalues. The slow eigenvectors decide which wavenumbers carry this error, through their projection onto the sine modes. For the spectral operator and the linear Fourier neural operator these eigenvectors are close to sine modes, so each wavenumber has its own rate and the error collects at high wavenumbers. Delocalized models, such as the dense networks under either forcing, have error of nearly the same magnitude at every wavenumber. The spectrum of the true solution decays with k and falls below this floor at some wavenumber, past which the relative error grows. No mode is learned more slowly than another in this case; the growth comes from the decay of the target spectrum. Spectral bias is usually reported by looking at this retained error in the sine basis, and we use its projection onto the kernel's eigendirections to explain the shape it has there.

1 Introduction

A neural network trained on a multiscale target fits some scales before others. In most reported cases, the low-wavenumber content of the target is learned before the high-wavenumber content [1, 2, 3, 4], and this behavior is called spectral bias. In autoregressive emulators of turbulent flows, the small scales that spectral bias leaves unlearned are amplified during time integration and make

*Corresponding author: cyaolai@stanford.edu
†Corresponding author: aschatto@ucsc.edu

long-term emulation unstable [5]. Classical iterative solvers for elliptic equations are the inverse. Jacobi and Gauss–Seidel iterations remove oscillatory error in a few sweeps, while smooth error is reduced far more slowly [6].

The current explanations for spectral bias assume that each mode is learned at a rate that decreases with wavenumber. In the neural tangent kernel picture, training converges fastest along the kernel's eigenfunctions with the largest eigenvalues [7]. In operator learning [8, 9, 10] we show that the per-mode picture only works when the eigenvectors of the kernel driving the training update are close to sine modes. For generic dense networks they are not. The trained error of these networks is nearly flat across wavenumber, at a level determined by the eigenvalues of that operator, and the relative error grows with wavenumber because the spectrum of the true solution decays through this flat floor, not because the network learns high wavenumbers more slowly.

Several kernel descriptions of spectral bias in scientific machine learning precede this work. Wang et al. [11, 12] derive the neural tangent kernel (NTK) of physics-informed neural networks [13], show that it is constant during training in the infinite-width limit under gradient descent, and trace training failure to different convergence rates of the loss components, determined by the eigenvalues of that kernel. Wang, Wang and Perdikaris [14] identify spectral bias with a bias toward the leading eigenvectors of the limiting NTK, and show that Fourier feature embeddings shift these eigenvectors to higher frequencies. Seroussi et al. [15] analyze over-parameterized networks through an equivalence with Gaussian process regression and describe spectral bias as a statement about kernel–task alignment. Khodakarami et al. [16] report wavenumber-resolved error metrics across physics-informed and operator-learning settings and find that spectral bias depends on the optimizer as well as on the architecture. For neural operators, Khodakarami et al. [17] reduce it by scaling the high-frequency components of the latent features. Xu et al. [18] review mitigation strategies, mainly frequency shifting and scaling, that assume a low-wavenumber-first learning order. Wang and Lai [19] remove the error plateau of regression and physics-informed networks, near 10^{-5}, by training successive networks on the residue of the previous one, whose content moves to higher frequencies at each stage.

Our setup differs from this work in a few ways. The kernel we study is that of the Adam update, with its diagonal preconditioner included (Section 4.1); with the preconditioner set to the identity it reduces to the gradient-flow NTK [7, 11]. Due to minibatching, the kernel is averaged over multiple samples. This is the cause of the cross terms in K, which capture the batch average in the weight updates. In the dense networks, the trained error is held in slow eigendirections of K that exist only because of these cross terms (Section 4.2); if each sample is treated separately, these directions disappear. There is no infinite-width or fixed-kernel assumption, since outside the lazy regime the kernel moves during training [20, 21, 22]. We measure the kernel over the last 100 steps and compare it with the error left at that point.

We work with the discrete Poisson problem and five operator models, from an ideal spectral operator to unstructured dense networks (Section 2). In the linear case (Section 3) the update is governed by the covariance of the training data, and its spectrum fixes the learning order exactly. Section 4 covers the nonlinear models, where we use the stacked kernel of the Adam iteration. Any model that does not learn the operator is left with error in the slow eigendirections of this kernel, and those slow rates can come from the data, from the nonlinearity (through coupling between training samples), or from the representation itself. Projecting the slow eigendirections onto the sine modes then gives the profile of the retained error in wavenumber. When they are sine-aligned this is a per-mode learning rate; for the delocalized eigendirections of the dense networks it is a flat floor. We discuss these results in Section 5, and Appendix A works through scalar regression, where the same statements can be made in closed form.

2 Operator learning for the Poisson problem

The operator of focus in this paper is the inverse of the one-dimensional discrete Poisson operator. Using the forcing $f \in \mathbb{R}^N$ as input, we define the target u as

$$u = A^{-1} f \in \mathbb{R}^N, \tag{1}$$

where $A \in \mathbb{R}^{N \times N}$ is the matrix form of the discrete Poisson operator. In scalar regression, spectral bias is often studied with the Fourier coefficients of a single target function. As this is an operator, we learn a linear map instead.

2.1 Discrete Poisson problem and the target operator

The setting is the Poisson equation on $(0, 1)$ with homogeneous Dirichlet boundary conditions,

$$-u_{xx}(x) = f(x), \qquad u(0) = u(1) = 0. \tag{2}$$

Using a second-order finite-difference discretization on N interior grid points gives the linear system $Au = f$, with

$$A = \frac{1}{h^2} \begin{pmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix}, \text{where} \quad h = \frac{1}{N+1}. \tag{3}$$

Let $S \in \mathbb{R}^{N \times N}$ be the standard orthonormal discrete sine basis, with entries

$$S_{jk} = \sqrt{\frac{2}{N+1}} \sin\left(\frac{\pi jk}{N+1}\right), \qquad j, k = 1, \ldots, N, \tag{4}$$

and columns v_k, $k = 1, \ldots, N$. Then $A = S \Lambda_A S^\top$ with

$$\Lambda_A = \operatorname{diag}(\kappa_1, \ldots, \kappa_N), \qquad \kappa_k = \frac{4}{h^2} \sin^2\left(\frac{\pi k}{2(N+1)}\right). \tag{5}$$

In the modal coordinates $\hat{f} = S^\top f$ and $\hat{u} = S^\top u$ the solution is

$$\hat{u}_k = \frac{1}{\kappa_k} \hat{f}_k, \tag{6}$$

so the target is diagonal in the sine basis. Its eigenvalues are $g_k^* = 1/\kappa_k$ for all k, with $g_k^* \approx 1/(\pi^2 k^2)$ for $k \ll N$. This is why the inverse of the discrete Poisson operator is low-pass, as it damps the high wavenumbers. This damping is an aspect of the target operator and is not studied here. The focus of our analysis is on how this operator is learned.

Forcing ensembles. Fresh training data is created for each batch by sampling forcings with independent mean-zero modal coefficients, $f = \sum_k \hat{f}_k v_k$ with $\mathbb{E}[\hat{f}_k \hat{f}_j] = \sigma_k^2 \delta_{kj}$. The forcing covariance $\Sigma_f := \mathbb{E}[f f^\top]$ is diagonal in the sine basis with eigenvalues σ_k^2. Two forcing ensembles are used. *Smooth forcing* has $\sigma_k^2 = k^{-4}$, so variance is concentrated at low k. *White forcing* has $\sigma_k^2 = 1$ and spreads variance across all modes equally.

Architecture hierarchy. We train five operator models. The ideal diagonal spectral operator has only one trainable eigenvalue for each of its $K_{\mathrm{ret}} = 32$ retained sine modes. The two Fourier neural operators [9, 10] combine per-mode spectral weights with pointwise channel mixing in each layer; the nonlinear FNO applies a GELU [23] after each layer, except the last, and the linear FNO does not. The GELU dense network is a multilayer perceptron on the grid values, and the deep linear dense network is the same perceptron with the nonlinearities and biases removed. We refer to the last four as the generic models and to the last two as the dense networks. All models are trained with Adam [24] on the mean-squared error over freshly sampled forcing batches. Architecture and training details are given in Appendix B.

2.2 A flat error floor

For a model $u_\theta = F_\theta(f)$ we define the RMS modal amplitude of the output over a validation set of forcings indexed by μ, $\langle (S^\top u_\theta^\mu)_k^2 \rangle_\mu^{1/2}$, and the spectrum of the true solutions,

$$U_k = \left\langle \left(S^\top u^\mu\right)_k^2 \right\rangle_\mu^{1/2}. \tag{7}$$

Figure 1 shows the trained hierarchy in physical space and in the sine basis. In physical space, the trained outputs overlay the true solution up to a fine-scale residual in Fig. 1(c,d). In the sine basis, each output tracks U_k up to a model-dependent mode. Beyond it, the generic models stay near an error floor, flat for the dense networks and almost flat for the nonlinear FNO, while U_k continues to decay, and the ideal diagonal spectral operator drops to zero at its truncation in Fig. 1(e,f). Under white forcing the linear FNO and the deep linear dense network track U_k over the whole band. The mode number where the gap starts to grow depends on the architecture and dataset. Training lowers the flat part of the error without changing its shape, so beyond the gap the error relative to the target grows with k. *This is the phenomenon usually reported as spectral bias.*

We can also state this as an error spectrum. For a trained model we define the RMS modal error over the validation set,

$$E_k = \left\langle \left(S^\top (u_\theta^\mu - u^\mu)\right)_k^2 \right\rangle_\mu^{1/2}, \tag{8}$$

and the relative RMS modal error,

$$\varepsilon_k = \sqrt{\frac{\left\langle \left(S^\top (u_\theta^\mu - u^\mu)\right)_k^2 \right\rangle_\mu}{\left\langle \left(S^\top u^\mu\right)_k^2 \right\rangle_\mu}}, \tag{9}$$

so that $\varepsilon_k = E_k / U_k$.

Figure 2 shows both error spectra at the end of training for all five models. Beyond a low-k regime, the modal error E_k of the dense networks and of the nonlinear FNO is nearly flat in k for both forcings in Fig. 2(a,b), at a level determined by the model and the forcing. The linear FNO is flat at a low level under white forcing but rises with k under smooth forcing, and the ideal diagonal spectral operator learns its retained modes exactly and increases after the truncation. Section 4.3 explains this range of shapes by the projection of the slow eigendirections of the training kernel onto the sine basis. The relative modal error ε_k in Fig. 2(c,d) grows with k for every generic model under smooth forcing, which is the well-known signature of spectral bias. The FNO models, however, level off at high k. Under smooth forcing their eigenvectors are localized on one or a few sine modes in Fig. 4(a,c), more cleanly than under white forcing in Fig. A4(a,c), so each mode approximately keeps its own error, σ_k times the error in the model's response at that mode, which falls with k in Fig. 2(a). The apparent severity of spectral bias decreases as the forcing places more

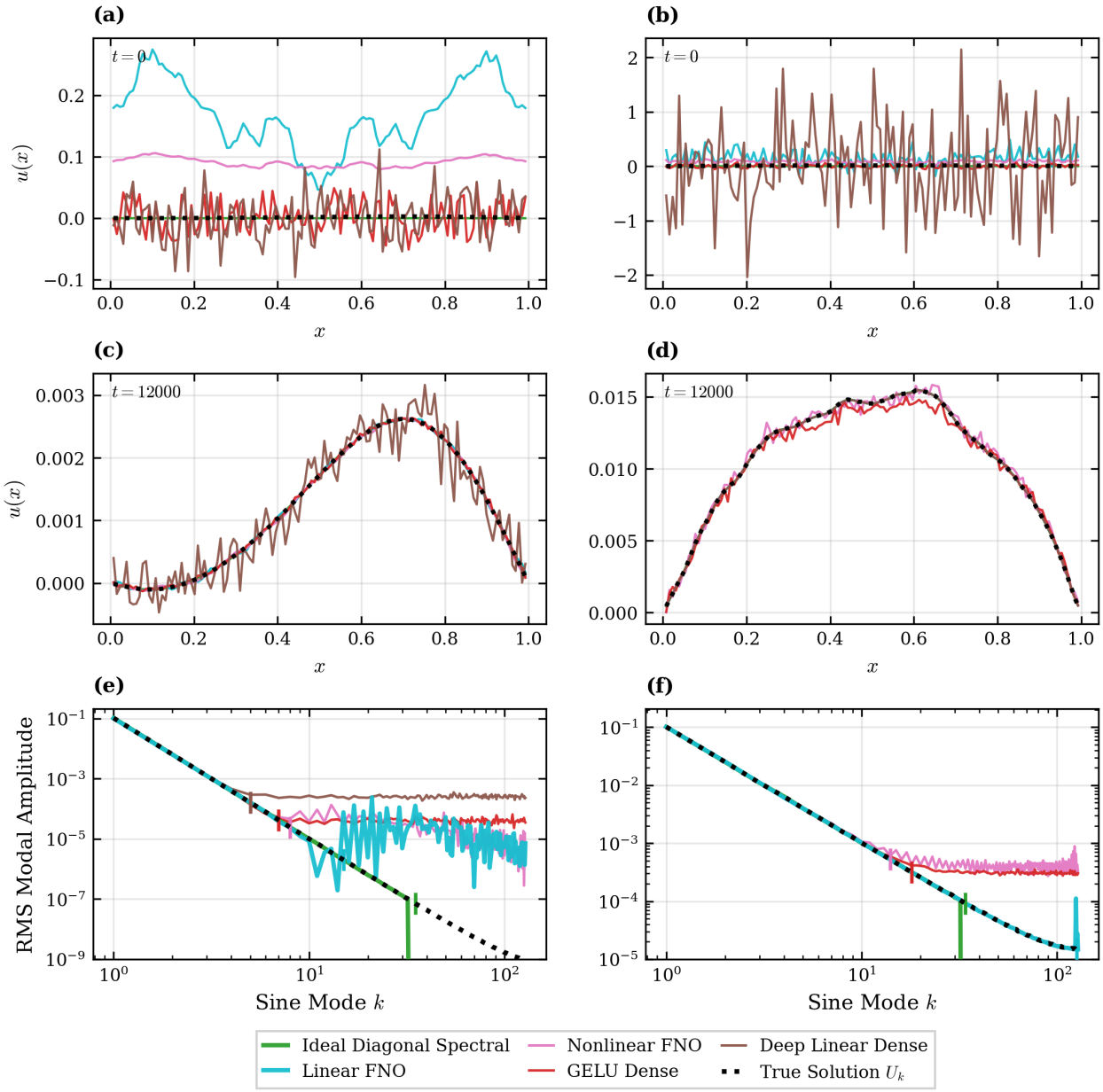


Figure 1: **Spectral bias in the trained hierarchy** (smooth forcing left, white forcing right). (a,b) Model outputs for one validation forcing at the first training step ($t = 0$) against the true solution (black). (c,d) The same at the final training step ($t = 12000$). (e,f) RMS modal amplitude of the trained outputs against the true solution spectrum U_k of Eq. (7). Vertical ticks mark the mode at which each model departs from U_k, and the crossing, computed after a three-point median filter on E_k, defines the gap mode k_{gap} of Eq. (10). The spike of the linear FNO at the end of the band in (f) is an input-independent offset at the highest odd modes (Appendix D).

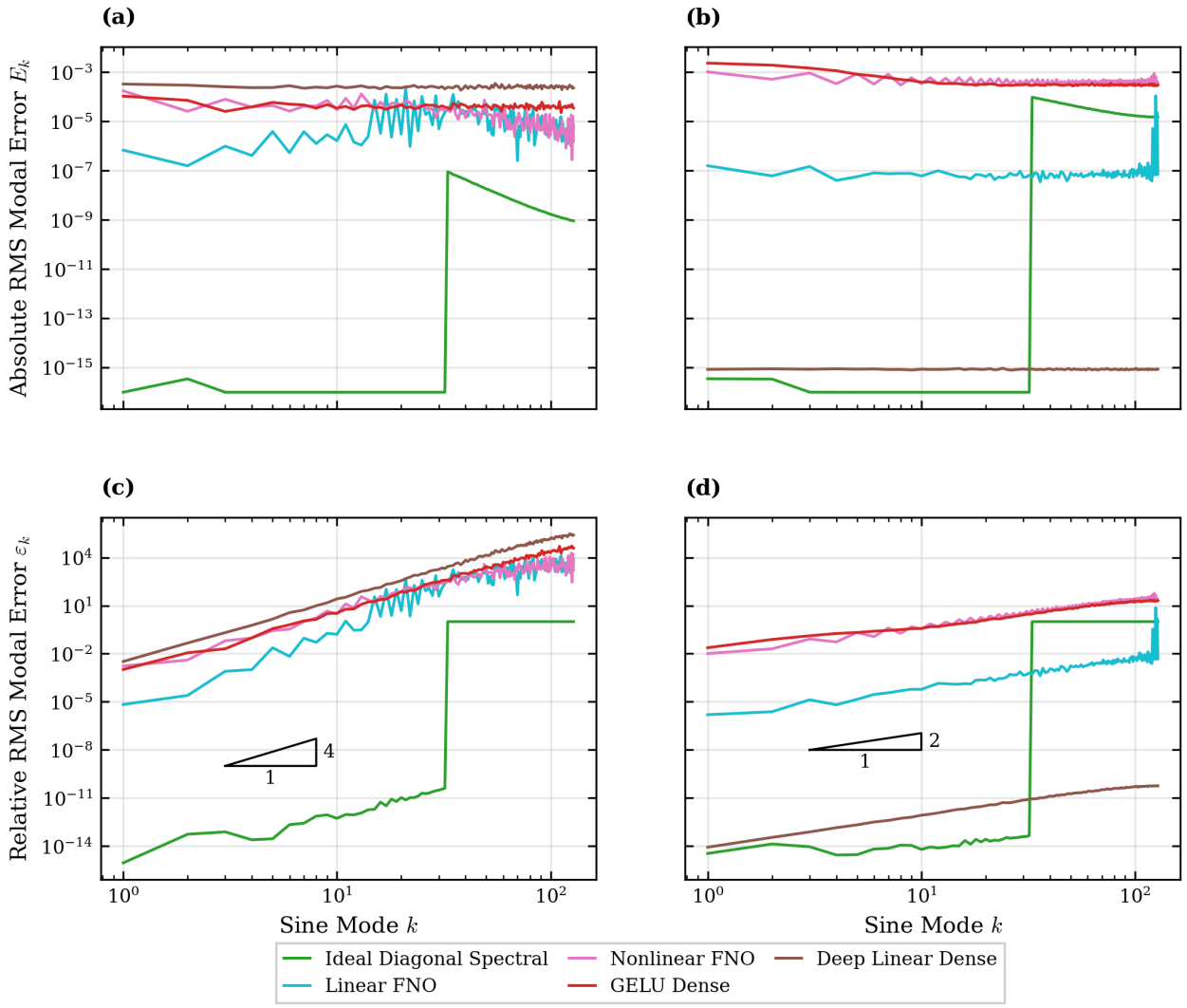


Figure 2: **Final modal error spectra of the hierarchy** (smooth forcing left, white forcing right). (a,b) Modal error E_k of Eq. (8). (c,d) Relative modal error ε_k of Eq. (9). Slope triangles mark $\varepsilon_k \propto k^4$ (smooth) and k^2 (white), the growth predicted for a flat E_k and $k \ll N$.

energy in the high modes: ε_k grows much more slowly under white forcing in Fig. 2(d) than under smooth forcing in Fig. 2(c), consistent with the linear analysis of Section 3.

The growth of ε_k does not label a mode at which learning dynamics change. For the dense networks and the nonlinear FNO, E_k is nearly constant in k beyond a model-dependent mode in Fig. 2(a,b) (see also Section 4.3), so the growth of $\varepsilon_k = E_k/U_k$ is the decay of its denominator. There is a single error level and a mode beyond which the target falls below it. We call this crossing the gap mode,

$$k_{\text{gap}} := \min\{k : E_k \geq U_k\}\,. \tag{10}$$

For $k \geq k_{\text{gap}}$ the output at mode k is dominated by error and stays at the floor regardless of the true solution at that mode. When $E_k < U_k$ at every mode, as for the deep linear dense network under white forcing, there is no gap. This is why the output spectra of Fig. 1(e,f) are flat in the gap region while the target continues to decay. Section 4 traces the level of the floor to the operator

that governs the training update.

3 The data covariance governs the learning order in linear operator learning

Under a linear learning rule, the error e about the minimizer evolves as $\dot{e} = -\mathcal{T}e$, where $\mathcal{T}$ is the symmetric operator representing the update. Expanding e in the eigenpairs $(\phi_\alpha, \lambda_\alpha)$ of $\mathcal{T}$ gives

$$e(t) = \sum_\alpha e^{-\lambda_\alpha t}\, a_\alpha(0)\, \phi_\alpha, \qquad a_\alpha(0) = \phi_\alpha^\top e(0). \tag{11}$$

A discrete iteration with step size η contracts each coefficient by $1 - \eta\lambda_\alpha$ per step, as in the scalar models of Appendix A. The eigenvalues λ_α set the learning rates, and they are themselves functions of the update operator $\mathcal{T}$. Analysis in the sine basis then becomes a projection of the eigenvectors ϕ_α onto the sine modes.

Let the model be a single matrix, $u_\theta = Mf$ with $M \in \mathbb{R}^{N\times N}$, trained on the population mean-squared error. With the operator error $\Delta := M - A^{-1}$, the loss is

$$\mathcal{L}(M) = \frac{1}{2}\,\mathbb{E}\|Mf - A^{-1}f\|^2 = \frac{1}{2}\,\mathbb{E}\Big[f^\top \Delta^\top \Delta\, f\Big] = \frac{1}{2}\,\mathrm{tr}\Big(\Delta\, \Sigma_f\, \Delta^\top\Big), \tag{12}$$

where the last equality uses the cyclic property of the trace. The forcing distribution enters the loss only through its covariance Σ_f. The gradient is $\nabla_M \mathcal{L} = \Delta\Sigma_f$, and under gradient flow, the operator error obeys

$$\dot{\Delta} = -\Delta\, \Sigma_f, \qquad \Delta(t) = \Delta(0)\, e^{-\Sigma_f t}. \tag{13}$$

In the eigenbasis of Σ_f, each component of the operator error decays exponentially at a rate equal to the corresponding eigenvalue. Writing

$$\Sigma_f = Q\, \Lambda_f\, Q^\top, \qquad \Lambda_f = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_N^2), \tag{14}$$

with orthonormal eigenvectors q_k, and projecting onto q_k,

$$\Delta(t)\, q_k = e^{-\sigma_k^2 t}\, \Delta(0)\, q_k. \tag{15}$$

In shallow linear operator learning, the forcing spectrum $\{\sigma_k^2\}$ sets the learning speed in each direction. The variance of the training data determines what decays quickly. High-variance directions get excited often and their error decays fast, while low-variance directions are not often seen and get penalized rarely in the loss. For this experiment, the covariances of the forcings in Section 2.1 are diagonal in the sine basis, $q_k = v_k$ with v_k the kth column of S, and Eq. (15) becomes

$$\Delta(t)\, v_k = e^{-\sigma_k^2 t}\, \Delta(0)\, v_k. \tag{16}$$

Under white forcing, the linear model learns every mode at the same rate, while under smooth forcing it learns the low modes first.

Loss decomposition in the sine basis. This covariance dependence is clearest when the learned operator is itself diagonal in the sine basis, $(S^\top u_\theta)_k = g_k \hat{f}_k$. The per-sample modal error is then $(g_k - 1/\kappa_k)\hat{f}_k$, and the population loss splits mode by mode,

$$\mathcal{L} = \frac{1}{2}\sum_{k=1}^{N} \sigma_k^2 \left(g_k - \frac{1}{\kappa_k}\right)^2. \tag{17}$$

The squared eigenvalue error at mode k is weighted by that mode's forcing variance σ_k^2. Under white forcing this weight is equal over all modes, whereas under smooth forcing high-mode errors are weakly penalized. As a result, a model can have large error in its high modes compared to its low modes and still have low loss.

Physical damping versus learning order. The eigenvalues $g_k^* \sim k^{-2}$ imply that, even under white forcing, the true solution must carry less energy in the high wavenumbers than the forcing. As in Section 2.1, low-pass behavior is an aspect of our chosen target operator and is separate from the learning order. Low wavenumbers are learned first here because the training data place more variance there, not because the Poisson operator smooths the optimization. Jacobi iteration behaves differently again, with damping rates that are eigenvalues of an iteration matrix built from A itself (Section 5). Figures 1 and 2 show the effect of the data directly: under white forcing the linear models have lower error and no accuracy gap, even though white forcing excites every mode. The deep linear dense network lies on the true solution in Fig. 1(d,f), and its modal error is at machine epsilon in Fig. 2(b). This is the benchmark against which the nonlinear results of Section 4 should be compared. In the linear case the update operator is the data covariance, and spectral bias is exactly the data spectrum. The forcing spectrum is the first of three sources of slow rates identified in this paper. Section 4 adds the nonlinearity of the network and the truncation of the representation.

4 Nonlinear operator learning

4.1 The stacked kernel of the Adam update

A neural operator is a map $F_\theta : \mathbb{R}^N \to \mathbb{R}^N$, $u_\theta = F_\theta(f)$, with parameters $\theta \in \mathbb{R}^P$. Given a training batch $\{(f^\mu, u^\mu)\}_{\mu=1}^B$ with $u^\mu = A^{-1} f^\mu$ and per-sample errors $e^\mu(\theta) = F_\theta(f^\mu) - u^\mu$ (sample indices are superscripts on vectors and subscripts on matrices throughout), the batch gradient of the mean-squared error, the mean over the BN entries of the batch, is $\frac{2}{BN} \sum_\mu J_\mu^\top e^\mu$ with $J_\mu := \partial F_\theta(f^\mu)/\partial\theta$. The models are trained with Adam, which we model as gradient descent preconditioned by the diagonal matrix

$$\Pi = \mathrm{diag}\Big(\big(s_1^{1/2} + \epsilon_{\mathrm{A}}\big)^{-1}, \ldots, \big(s_P^{1/2} + \epsilon_{\mathrm{A}}\big)^{-1} \Big), \tag{18}$$

where s_i is the running second-moment estimate of gradient coordinate i and ϵ_{A} is the stabilization constant. We neglect the momentum average. Appendix B shows that this is valid for the slow eigendirections that the analysis concerns. Under the preconditioned flow, the evolution of the parameters is governed by

$$\dot\theta = -\frac{2}{BN}\, \Pi \sum_{\mu=1}^{B} J_\mu^\top e^\mu, \tag{19}$$

and by the chain rule each sample's error evolves as [7, 25]

$$\dot e^\nu = J_\nu \dot\theta = -\frac{2}{BN} \sum_\mu J_\nu\, \Pi\, J_\mu^\top e^\mu = -(Ke)^\nu, \qquad K_{\nu\mu} = \tfrac{2}{BN}\, J_\nu\, \Pi\, J_\mu^\top, \tag{20}$$

where the B per-sample errors are stacked into one vector $e = (e^1; \ldots; e^B) \in \mathbb{R}^{BN}$ and the per-sample Jacobians into $J := (J_1; \ldots; J_B) \in \mathbb{R}^{BN \times P}$. We call $K = \frac{2}{BN}\, J \Pi J^\top$ the "stacked kernel".

Setting $\Pi = I$ recovers the gradient-flow NTK. Written out over the batch,

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} e^1 \\ e^2 \\ \vdots \\ e^B \end{pmatrix} = -\frac{2}{BN}\begin{pmatrix} J_1\Pi J_1^\top & J_1\Pi J_2^\top & \cdots & J_1\Pi J_B^\top \\ J_2\Pi J_1^\top & J_2\Pi J_2^\top & \cdots & J_2\Pi J_B^\top \\ \vdots & \vdots & \ddots & \vdots \\ J_B\Pi J_1^\top & J_B\Pi J_2^\top & \cdots & J_B\Pi J_B^\top \end{pmatrix}\begin{pmatrix} e^1 \\ e^2 \\ \vdots \\ e^B \end{pmatrix}, \tag{21}$$

and the ν-th row is

$$\dot{e}^\nu = -\frac{2}{BN}\underbrace{J_\nu\Pi J_\nu^\top e^\nu}_{\text{per-sample block}} - \frac{2}{BN}\underbrace{\sum_{\mu\neq\nu} J_\nu\Pi J_\mu^\top e^\mu}_{\text{cross terms}}. \tag{22}$$

The off-diagonal blocks $J_\nu\Pi J_\mu^\top$ with $\nu \neq \mu$ are the cross terms. Through the shared parameters, these cross terms link each sample's error to the errors on the other samples. The block-diagonal kernel $\mathrm{diag}_\mu(J_\mu\Pi J_\mu^\top)$, obtained by giving each sample its own copy of the parameters, does not contain them, so any diagnostic built on the per-sample kernel alone misses them. Section 4.2 shows that dense networks have training error along the eigendirections through these cross terms.

The stacked kernel is symmetric and positive semidefinite, since Π is diagonal with positive entries and $K = \frac{2}{BN}(J\Pi^{1/2})(J\Pi^{1/2})^\top$. It therefore has an orthonormal eigenbasis $\{\phi_\alpha\}$ with eigenvalues $\lambda_\alpha \geq 0$,

$$K = \sum_\alpha \lambda_\alpha\, \phi_\alpha\phi_\alpha^\top, \tag{23}$$

and while the kernel is frozen the stacked error evolves diagonally in this basis,

$$e(t) = \sum_\alpha e^{-\lambda_\alpha t}\left(\phi_\alpha^\top e(0)\right)\phi_\alpha. \tag{24}$$

As in Section 3, each λ_α is the rate at which training contracts the component along ϕ_α, with a per-step factor $1-\eta\lambda_\alpha$ in the discrete iteration. We plot this per-step rate $\eta\lambda_\alpha$ at the learning rate η of the sampled step. The eigenpairs $(\lambda_\alpha, \phi_\alpha)$ are the modal coordinates of the learning problem. For two-layer networks in the kernel regime, Arora et al. [26] decompose the training error in the same way, along the eigenvectors of the Gram matrix of the network at initialization; here the kernel is that of the late-training Adam update.

Equation (24) holds when K is fixed. K is a function of θ and the tracked Adam states. Initialization tells us almost nothing due to the changing nature of the kernel. Thus, we evaluate K late in training at the loss floor and where Π is nearly stationary over the sampled window (Appendix B). The decomposition of the trained error over the eigendirections of K in Section 4.2 uses K averaged over the final 100 steps, and the modal-kernel diagnostics of Section 4.3 use K at the final step, with Π taken from the optimizer state at each sampled step.

4.2 Error accumulates in the slow eigendirections of the kernel

Because $\{\phi_\alpha\}$ is orthonormal, the trained error at a given step decomposes exactly as

$$e = \sum_\alpha a_\alpha\phi_\alpha, \qquad a_\alpha = \phi_\alpha^\top e, \qquad \|e\|^2 = \sum_\alpha a_\alpha^2. \tag{25}$$

The coefficient a_α is the loading of the error on eigendirection α, and a_α^2 is that direction's share of the squared error. We call the set of pairs $\{(\lambda_\alpha, a_\alpha^2)\}$ the error-loaded spectral measure. It

represents where on the rate axis the error resides. We summarize it by the cumulative error mass below a rate λ, in RMS units,

$$\Gamma(\lambda) = \left(\frac{1}{BN} \sum_{\alpha:\, \lambda_\alpha < \lambda} a_\alpha^2 \right)^{1/2}, \tag{26}$$

which increases from 0 to the full RMS error $E := \|e\|/\sqrt{BN}$ as λ increases, where B is the batch size and N is the number of grid points. The measure is computed by error-loaded Lanczos applied matrix-free to the late-training kernel, with the cross terms never dropped (Appendix B); the pairs $(\lambda_\alpha, a_\alpha^2)$ it returns are Ritz pairs of the kernel, which serve in place of the exact eigenpairs throughout. We locate the bulk of the measure by the rate at which the cumulative error mass reaches 95% of the RMS error,

$$\lambda^* := \min\big\{\lambda : \Gamma(\lambda) \geq 0.95\, E\big\}, \tag{27}$$

so that $0.95^2 \approx 90\%$ of the squared error resides below λ^*. The threshold of 95% is a convenience for analyzing the figures, and the conclusions do not depend on its value.

The central result of this section is in Fig. 3, which shows the measure for the GELU dense network under smooth and white forcing. For the full stacked kernel, the cumulative error mass accumulates over multiple small eigenvalues and reaches the independently measured error E, the dotted line in Fig. 3(a,b). When the cross terms are removed and K is replaced by its block-diagonal part, the cumulative mass collapses, leaving less than 0.1% of the squared error below $\eta\lambda \approx 8 \times 10^{-5}$ for either forcing. For this model, the slow eigendirections that retain the error exist only through the coupling between samples. Panels (c,d) resolve the same eigensystem direction by direction. Under both forcings, most of the trained error resides in the eigendirections of the stacked kernel with small eigenvalues.

The same statement holds for every model that does not learn the operator, whatever the source of the slow rates. Table 1 collects the ten cases, with the training loss of each from Fig. A5. The deep linear dense network and the linear FNO under white forcing are the two cases whose training loss reaches machine precision or is still descending at the end of training. In neither case does the trained error reside in eigendirections with small eigenvalues. The residual of the linear FNO occupies two eigendirections near λ^*, and the residual of the deep linear dense network is at machine precision and lies mostly in the null space of the kernel, with the rest at the fast end of its spectrum (Appendix C). The other eight cases plateau, and in every one of them the dominant error shares lie below λ^*.

The table identifies three sources of slow rates. First, the forcing spectrum provides them in the smooth forcing case, for linear and nonlinear models alike. This is the result of Section 3, and it is why the deep linear dense network retains its error under smooth forcing and is at machine precision under white forcing. Second, the nonlinearity of the network provides them even when the data excite every mode. This is why the nonlinear models plateau under white forcing while the linear models learn the operator. For the GELU dense network these slow directions exist only through the cross terms: the block-diagonal kernel retains none of its error below $\eta\lambda = 10^{-5}$, against 60% for the full kernel in Fig. 3(b), so an analysis that drops the cross terms cannot locate its error. For the nonlinear FNO the cross terms also shape the slow tail: 90% of the squared error lies below λ^* for the full kernel and 52% without the cross terms, and its slowest directions, $\eta\lambda < 10^{-7}$, exist only through them; see Table 1 and Fig. A2(a). For every generic model, an analysis that drops the cross terms misplaces the error on the rate axis. Third, the representation provides them as the null space of the kernel, which is the truncation of the ideal diagonal spectral operator at $k = 32$. The forcing spectrum affects the retained error in two ways. It decides which eigenvalues

Model	Forcing	Loss at the end	Error-carrying directions	Error below λ^* without cross terms	Fig.
GELU dense	smooth	plateau	slow tail	absent (0)	3(a,c)
GELU dense	white	plateau	slow tail	absent (0.12)	3(b,d)
Nonlinear FNO	smooth	plateau	slow tail	present (0.49)	A3(a,b)
Nonlinear FNO	white	plateau	slow tail	present (0.53)	A2(a,b)
Linear FNO	smooth	plateau	slow tail	present (0.13)	A3(c,d)
Linear FNO	white	descending	two directions near λ^*	0.22	A2(c,d)
Deep linear dense	smooth	plateau	slow tail	absent (0.004)	A3(e,f)
Deep linear dense	white	machine epsilon	null space and fast directions	1.0	A2(e,f)
Ideal diagonal spectral	smooth	truncation floor	null space	1 (null space)	A3(g,h)
Ideal diagonal spectral	white	truncation floor	null space	1 (null space)	A2(g,h)

Table 1: The hierarchy under both forcings. Training loss is the end of training from Fig. A5. The last two columns are from the error-loaded spectral measure in the figure listed: where the error resides on the rate axis, and the share of the squared error that remains below λ^* when the cross terms are removed.

are small and therefore how much error remains. It also decides where the gap mode of Eq. (10) falls, because the spectrum U_k of the true solution decreases faster with k under smooth forcing than under white forcing, so the true solution drops below the error level at a lower wavenumber. Appendix C discusses each model.

4.3 Misalignment with the sine basis and the accuracy gap

Because spectral bias is read off in the sine basis, the projection of the slow eigendirections onto the sine modes fixes its profile. Slow eigendirections that are individual sine modes give a retained error localized at particular wavenumbers, with a separate learning rate for each mode; slow eigendirections delocalized over the sine modes give an error spread evenly over k. The ideal diagonal spectral operator is sine-aligned by construction, the FNOs are aligned to the extent that the forcing separates their rates, and the dense networks are indistinguishable from random directions in the sine basis under both forcings. For the dense networks the retained error is therefore flat in k, and measured against the decaying target spectrum this flat error reproduces the accuracy gap of Fig. 1.

Alignment statistics. The alignment is measured on the late-training batch-averaged kernel $\bar{K} := \frac{2}{N}\bar{J}\,\Pi\,\bar{J}^\top \in \mathbb{R}^{N\times N}$, where $\bar{J}$ is the Jacobian of the batch-averaged network output on a fixed diagnostic batch (Appendix B), normalized so that $\eta\lambda_\alpha$ is the per-step rate at which an error shared by every sample of the batch contracts. $\bar{K}$ is the $N \times N$ analogue of the per-sample block $J_\mu \Pi J_\mu^\top$ with the batch average taken inside the Jacobian, and $S^\top \bar{K} S$ is its modal kernel. For a truly diagonal kernel in the sine basis, the error in each mode would reduce independently of all others. This would be determined by each mode's eigenvalue and would give a perfect per-mode learning speed. For a unit eigenvector ϕ_α of $\bar{K}$, let $\psi_{k\alpha} := v_k^\top \phi_\alpha$ be its component on sine mode k, so that $\sum_k \psi_{k\alpha}^2 = 1$. We call $\psi_{k\alpha}^2$ the squared overlaps of eigenvector α with the sine modes and summarize them by two statistics,

$$\bar{k}_\alpha := \sum_k k\,\psi_{k\alpha}^2, \qquad \mathrm{PR}_\alpha := \Big(\sum_k \psi_{k\alpha}^4\Big)^{-1}, \tag{28}$$

the frequency centroid and the participation ratio of eigendirection α. $\mathrm{PR}_\alpha = 1$ means the direction is a single sine mode. The sum $\sum_k \psi_{k\alpha}^4$ is the inverse participation ratio: it equals 1 for a single

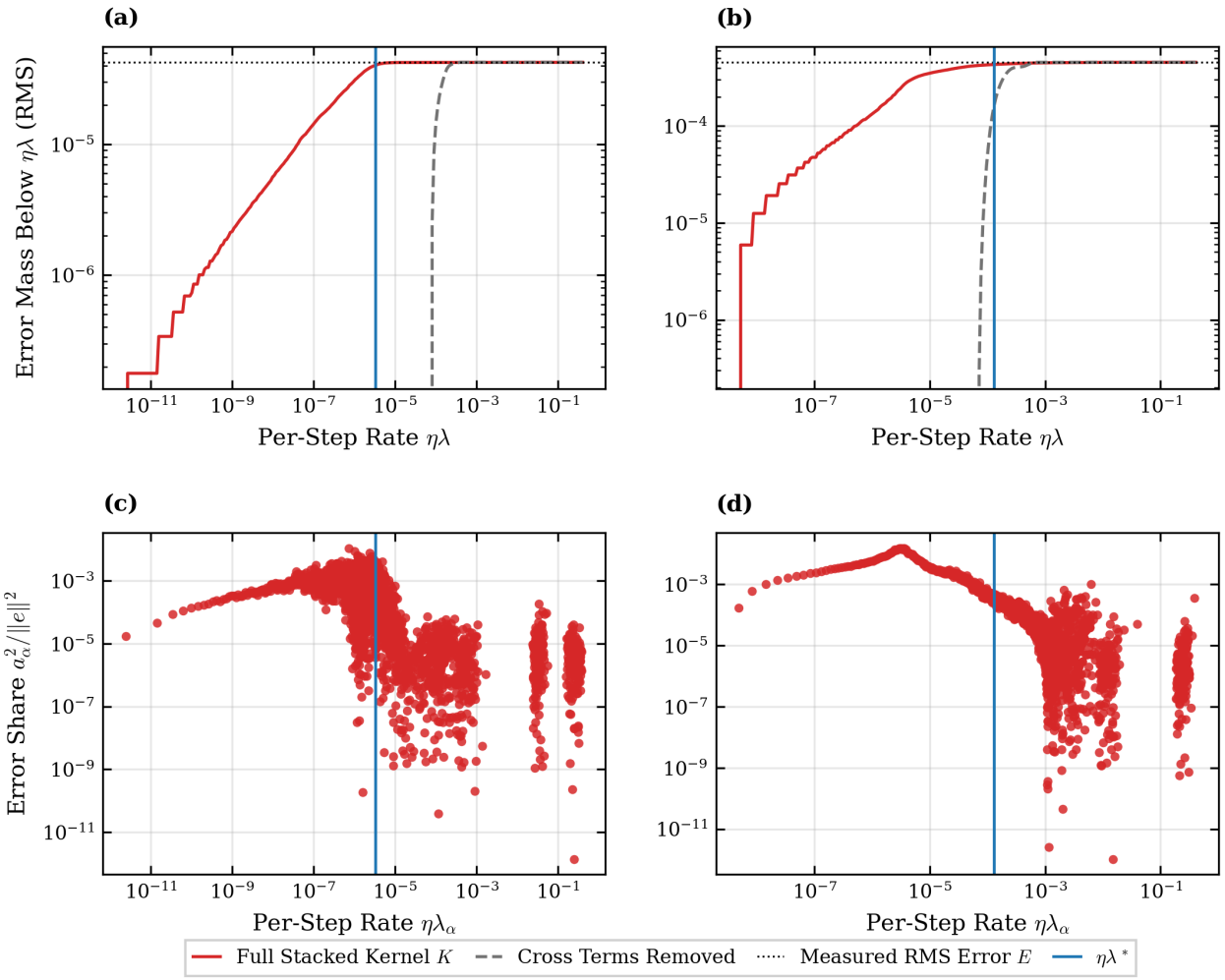


Figure 3: **The error is retained in the slow eigendirections of the stacked kernel, which for this model exist only through the cross terms** (GELU dense network; smooth forcing left, white forcing right). (a,b) Cumulative error mass $\Gamma(\lambda)$ of Eq. (26) for the full stacked kernel K (red) and for the block-diagonal kernel with the cross terms removed (grey dashed). The dotted line marks the measured RMS error E and the vertical line marks $\eta\lambda^*$, with λ^* of Eq. (27) the rate at which Γ reaches $0.95\,E$, below which about 90% of the squared error resides. (c,d) Each Ritz pair plotted at its per-step rate $\eta\lambda_\alpha$ against its error share $a_\alpha^2/\|e\|^2$.

sine mode and $1/n$ for a direction spread evenly over n modes. $\mathrm{PR}_\alpha \gg 1$ means the single rate λ_α is shared across many sine modes at once, and PR_α counts the effective number of modes over which the direction is spread.

We plot in Fig. 4(a) the squared overlaps $\psi_{k\alpha}^2$ of Eq. (28), the fraction of each eigenvector ϕ_α of $\bar{K}$ that lies along sine mode v_k, under smooth forcing. For the ideal diagonal spectral operator every eigenvector is a single sine mode, by construction. Each of its parameters controls one output mode, its Jacobian columns are the scaled sine modes $\hat{f}_k^\mu\, v_k$, and its modal kernel is diagonal at every forcing. The diagonal entries of $S^\top \bar{K} S$ scale with $\langle \hat{f}_k \rangle^2$, the batch average of the forcing coefficients, so under smooth forcing they follow $\sigma_k^2 = k^{-4}$ and the rate falls with k in Fig. 4(b). The non-retained modes span the null space of the kernel, so representational truncation is absorbed by

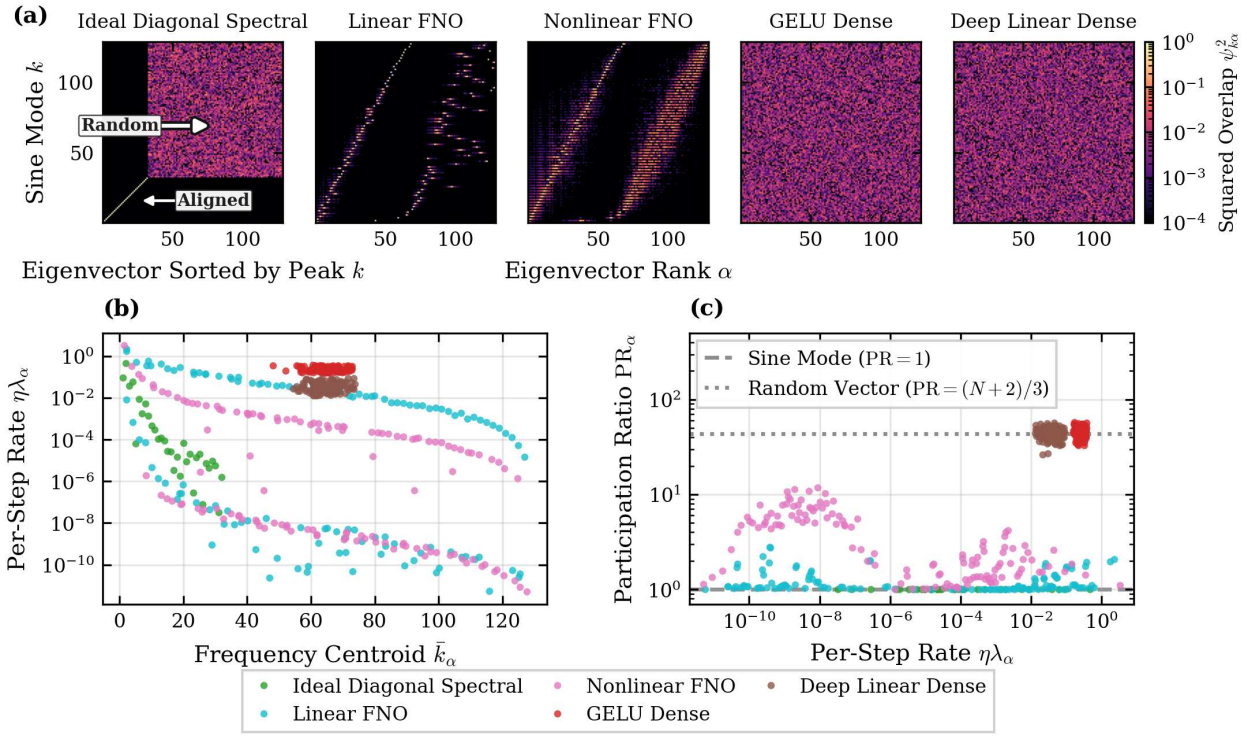


Figure 4: **Alignment of the kernel eigensystem with the sine basis** (smooth forcing; late-training batch-averaged kernel $\bar{K}$ of Section 4.3). (a) Squared overlaps $\psi^2_{k\alpha}$ of Eq. (28) between the eigenvectors of $\bar{K}$ and the sine modes, for the five models. Eigenvectors are sorted by eigenvalue, except for the ideal diagonal spectral operator, whose retained eigenvectors are sorted by their peak sine mode k; its non-retained modes span the null space, whose basis is arbitrary. (b) Per-step rate $\eta\lambda_\alpha$ of each eigenvector against its frequency centroid $\bar{k}_\alpha$. (c) Participation ratio PR_α of each eigenvector against its per-step rate. The dashed line marks a single sine mode, PR = 1. The dotted line marks the expected participation ratio of a unit vector drawn uniformly at random from the sphere in $\mathbb{R}^N$, $(N+2)/3$.

the same law as the $\lambda_\alpha \to 0$ limit. The linear FNO is sine-diagonal, with a single stripe of squared overlaps equal to one ordered by rank. The nonlinear FNO is banded around a stripe, with each eigenvector localized on a few sine modes within one parity sector by the selection rule discussed in Appendix D. The GELU dense and deep linear dense networks show no sine structure at all. Each of their eigenvectors spreads over many sine modes, so each rate is shared across many modes at once.

Figure 4(b) plots each eigenvalue against the frequency centroid of its eigenvector. For the ideal diagonal spectral operator and the FNOs, the rate falls as the centroid increases, by about five orders of magnitude over the 32 retained modes of the ideal operator and by ten orders of magnitude from 0 to N for the FNOs. Error in eigendirections with more high-wavenumber content therefore decays more slowly, which is spectral bias represented in the eigenbasis of the kernel.

The decrease of λ_α with $\bar{k}_\alpha$ follows from the forcing spectrum $\sigma_k^2 = k^{-4}$, which appears in the modal kernel through the Jacobian columns. Adam divides gradient coordinate i by $s_i^{1/2} + \epsilon_{\mathrm{A}}$. Where $s_i^{1/2} \gg \epsilon_{\mathrm{A}}$ this division cancels the scale of the gradient on parameter i and would remove the forcing spectrum from the kernel. By the end of training, the gradients on most parameters

have decayed below ϵ_{A}: $s_i^{1/2} < \epsilon_{\mathrm{A}}$ for between 93.5% and 99.9% of the parameters of the GELU dense network and the nonlinear FNO under both forcings, and for all of the FNO spectral weights on modes $k > 64$ under smooth forcing. On those parameters, $\Pi_{ii} \approx \epsilon_{\mathrm{A}}^{-1}$ and the kernel reduces to $\frac{2}{BN}\,\epsilon_{\mathrm{A}}^{-1} J J^\top$, in which the forcing spectrum appears through the Jacobian columns (Appendix C). The ordering of the eigenvalues with $\bar{k}_\alpha$ is therefore not removed by the preconditioner. The kernel makes high-wavenumber directions slow to the extent that the training data fail to excite them.

For the linear FNO, the eigenvectors are the sine modes and the rates follow the forcing spectrum, so its learning order is the data covariance through the kernel, as for the linear model of Section 3. For the nonlinear FNO, the nonlinearity mixes neighboring modes, and the learning order depends on the data through the kernel but is no longer the covariance alone. The dense networks instead collapse to clusters near $\bar{k}_\alpha \approx N/2$ with little spread in rate. The centroid of a delocalized eigenvector is close to the middle of the band regardless of its rate, so the centroid carries no information for these models. Figure 4(c) gives the participation ratios. Eigenvectors on the random-vector line are statistically indistinguishable from random directions in the sine basis. The dense networks lie on this line, the linear FNO at PR = 1 at every rate, and the nonlinear FNO between 1 and about 10, with its largest values at its slowest rates.

Note that a Frobenius measure of diagonality, i.e., the fraction of the squared entries of the modal kernel $S^\top \bar{K} S$ that lie on the diagonal, does not detect this misalignment. The eigenvectors of a matrix are determined by the size of its off-diagonal entries relative to the gaps between its diagonal entries, so when many diagonal entries are nearly equal, off-diagonal entries that are negligible in Frobenius norm are still large enough to mix the corresponding modes, and the eigenvectors become arbitrary combinations of the near-degenerate sine modes. This is the situation for the GELU dense network, whose kernel is close to a multiple of the identity matrix and is therefore nearly diagonal in the sine basis by the Frobenius measure, even though its eigenvectors are indistinguishable from random directions in that basis, as Fig. 4(c) shows. A per-mode learning rate exists only if the eigenvectors are single sine modes (PR = 1) and the corresponding rates differ from mode to mode. This holds for the ideal diagonal spectral operator and for the smooth-forcing linear FNO but not for the other models. This is why the analysis of Section 4.2 uses the eigenbasis of the kernel rather than the sine basis.

The flat floor is the projection of the slow eigendirections. We now return to the sine basis, in which spectral bias is reported. Figure 5(a,b) shows the modal error E_k of the GELU dense network at logarithmically spaced training steps under smooth and white forcing. For both forcings, beyond a low-k regime, the trained spectrum is flat in k and descends uniformly as training proceeds, i.e., training lowers the error at all wavenumbers at the same rate. This is explained by the kernel eigensystem. A slow, delocalized eigendirection contributes the same small residual to every mode onto which it projects. Let $\mathcal{A} := \{\alpha : \lambda_\alpha < \lambda^*\}$ be the set of slow eigendirections of Eq. (27) and $e_{\mathrm{slow}} := \sum_{\alpha\in\mathcal{A}} a_\alpha \phi_\alpha$ the part of the error they retain. For a stacked eigendirection $\phi_\alpha \in \mathbb{R}^{BN}$ with per-sample blocks ϕ_α^μ, write $\psi_{k\alpha}^2 := \sum_\mu (v_k^\top \phi_\alpha^\mu)^2$ for its squared overlap with sine mode k, which again sums to one, over all k. The trained modal error is then

$$E_k^2 \;\approx\; \frac{1}{B}\sum_{\alpha\in\mathcal{A}} \psi_{k\alpha}^2\, a_\alpha^2 \;\approx\; \frac{\|e_{\mathrm{slow}}\|^2}{BN}, \tag{29}$$

where the factor $1/B$ is the average over the B samples in the RMS of Eq. (8), and the second approximation uses $\psi_{k\alpha}^2 \sim 1/N$ for directions with $\mathrm{PR}_\alpha = \mathcal{O}(N)$, as in Fig. 4(c), since a direction spread over $\mathcal{O}(N)$ modes places weight of order $1/N$ on each of them. The result is independent of k and, in particular, independent of the true solution at mode k. Against the decaying target

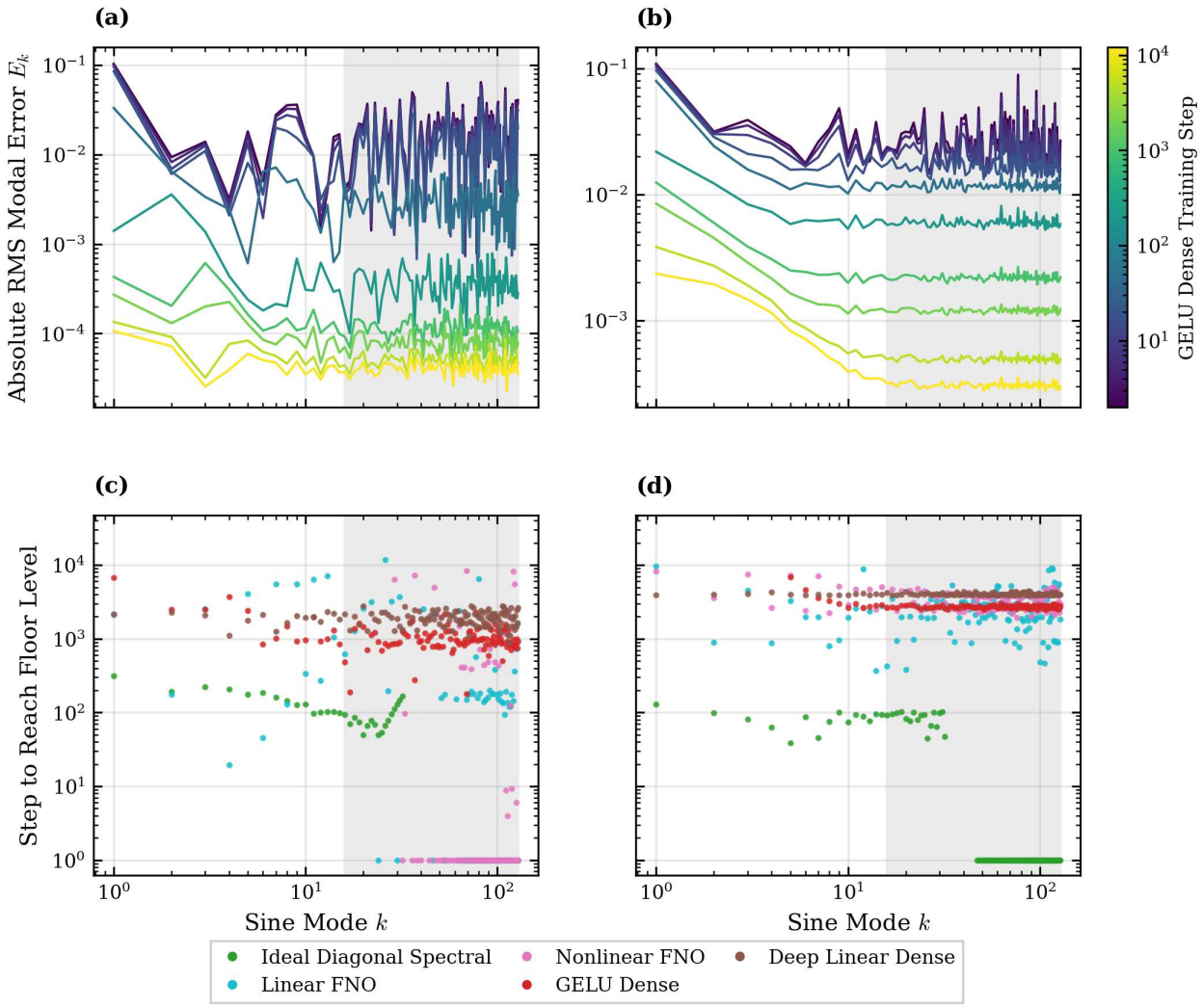


Figure 5: **The modal error of a delocalized neural operator is flat in k and every mode reaches its floor at the same time** (smooth forcing left, white forcing right). (a,b) Modal error E_k of the GELU dense network at logarithmically spaced training steps. The shaded band marks the gap region ($k \geq k_{\text{gap}}$). (c,d) Training step at which the error of each mode first reaches the floor level, 3× the final gap-median error, for every model. Modes whose error lies below this level from the first step are plotted at step 1.

spectrum $U_k \approx \sigma_k/\kappa_k$, this flat floor produces the gap mode of Eq. (10) and the growing relative error of Fig. 2, as shown in Section 2.2, with $\varepsilon_k = E_k/U_k \propto \kappa_k/\sigma_k$, which grows as k^4 under smooth forcing and as k^2 under white forcing for $k \ll N$, as the slope triangles in Fig. 2(c,d) show. The error in k_{gap} is not evidence that the network fails to learn high wavenumbers. It is an error that is the same at every wavenumber, measured against a target whose spectrum decays through it.

The smooth-forcing linear FNO has error that aligns with sine modes. Because of this, the error on each sine mode decays independently at its own rate, as in Eq. (16), and the high modes, which smooth forcing rarely excites, are learned last. Its error is jagged in Fig. 2(a) because each eigendirection retains its error at its own wavenumber rather than spreading it by Eq. (29). The deep linear dense network is delocalized as its eigenvectors are random in the sine basis, which

causes the flat floor. This is shown in Figs. 4(c) and A4(c). The nonlinear FNO lies between them. Its slowest eigendirections have PR of order ten and there are many of them across the band, so their sum is almost flat (Fig. 2, pink).

Over a window in which the kernel is close to fixed, such as the closing window in which K is sampled, the time evolution of the error follows from the same decomposition; it does not describe the whole run of Fig. 5, over which the kernel changes. While the kernel is fixed, $a_\alpha(t) = a_\alpha(0)\, e^{-\lambda_\alpha t}$ by Eq. (24), and substituting into Eq. (8) gives

$$E_k^2(t) = \frac{1}{B} \sum_{\alpha,\beta} a_\alpha(0)\, a_\beta(0)\, e^{-(\lambda_\alpha+\lambda_\beta)t} \sum_\mu (v_k^\top \phi_\alpha^\mu)(v_k^\top \phi_\beta^\mu). \tag{30}$$

The terms with $\alpha = \beta$ are

$$\sum_\alpha \chi_{k\alpha}\, e^{-2\lambda_\alpha t}, \qquad \chi_{k\alpha} = \psi_{k\alpha}^2\, a_\alpha^2(0), \tag{31}$$

and the terms with $\alpha \neq \beta$ do not vanish, since the eigendirections are orthogonal in $\mathbb{R}^{BN}$ but their projections onto a single sine mode are not orthogonal over the samples. At late times both sums are dominated by the eigendirections with the smallest λ_α that have nonzero weight on mode k. Note that every mode couples weakly to some fast directions through the weights $\chi_{k\alpha}$, so an arithmetic average of the eigenvalues acting on a mode is dominated by the fast end of the spectrum and does not predict the final error. The eigenvalues must be exponentiated before they are averaged, and the final error of a mode is governed by the slowest eigendirections to which it couples.

Synchronous floor times. Figure 5(c,d) restates the uniform descent of Fig. 5(a,b) mode by mode. It shows the training step at which the error of each mode first reaches its floor level, under smooth and white forcing. The dense networks, whose slow eigendirections are delocalized, reach the floor at essentially the same step across all modes under both forcings. The FNOs are spread over an order of magnitude in training step. All of this shows why mode-indifferent learning is a property of the delocalized architectures, not of the problem or of the forcing.

Error level and error shape are determined separately. The eigenvalues and eigenvectors of the stacked kernel appear in Eq. (31) through different factors. The eigenvalues govern how much error is retained during training, the error mass loaded below λ^* (Fig. 3; Figs. A2 and A3). The squared overlaps $\psi_{k\alpha}^2$ of Eq. (29) show where the error is projected over wavenumber k. Under white forcing, the deep linear dense network and the linear FNO, one random in the sine basis and one localized in Fig. A4(c), both learn the operator in Fig. A5(b) and have no accuracy gap in Fig. 1(f). Under smooth forcing, the same deep linear dense network, with the same random projection in Fig. 4(c), retains error in the slow tail, and a gap in accuracy is observed in Figs. A3(e,f) and 1(e). The GELU dense network and the nonlinear FNO, random and localized in the same way, both retain error in the slow tail under white forcing, and a gap in accuracy is observed. Under smooth forcing, all four generic models retain error at comparable levels, with $E = 3.6 \times 10^{-5}$ for the linear FNO, 3.3×10^{-5} for the nonlinear FNO, 4.2×10^{-5} for the GELU dense network and 2.6×10^{-4} for the deep linear dense network, and their spectra range from rising and jagged for the linear FNO to flat for the dense networks according to their projection in Fig. 2(a). The projection onto the sine basis decides its shape, either flat for delocalized directions, or dependent on eigenvalue for sine-aligned error.

5 Discussion

Under the linearized model for learning dynamics, Eq. (24), we can show that modal learning depends on the spectrum of the unfrozen kernel. Any statement about error in a functional basis (sine, in this paper) becomes a question about how that basis projects onto the kernel eigenvectors. For shallow linear models, the kernel dynamics are entirely determined by the dataset, and thus the forcing covariance determines the dynamics (Section 3). In nonlinear operator learning with Adam, this stacked kernel depends on many factors, such as the architecture and the optimizer, and how it projects onto the error is not clear until the network is trained (Section 4).

This theory resolves the apparent conflict with classical numerical analysis by providing a framework that captures the learning dynamics. Weighted Jacobi iteration for this Poisson problem damps high-frequency error fastest as a result of its update operator $I - \omega \operatorname{diag}(A)^{-1}A$, with $\omega = 2/3$ [6]. Training a network, however, results in a more complex process that needs further analysis. For the ideal diagonal spectral operator and the Fourier neural operators, the kernels are somewhat aligned with the sine basis and have a spectrum that decreases with mode number. These models demonstrate spectral bias in that they learn each mode at a rate set by the eigenspectrum of the stacked kernel of Eq. (22). The theory also explains why the MLP-based models do not show true spectral bias—their error is flat in the sine basis—since their kernels have no sine-aligned structure. This structure depends on many factors, and there does not seem to be a way to determine the presence of spectral bias before training, because the structure of the stacked kernel is hard to predict.

The analysis restricts what spectral bias can mean for generic architectures. There is now a link between the residual vector and any chosen analysis basis. Spectral bias has the structure it does for two reasons. In some architectures, the network actually learns some modes faster, as its stacked kernel eigenvectors align with the chosen basis and, more importantly, this alignment corresponds to a drop in eigenvalue as the frequency increases. In contrast, the MLP-based models have no such alignment, so their eigenvalues carry no information about which sine modes are learned. Regardless of eigenvector alignment, the eigenvalues of the stacked kernel have a notable impact on the magnitude of the final error. Small eigenvalues trap error that is not removed over training, and so spectral bias is in practice determined by how well the small-eigenvalue eigenvectors project onto a chosen basis. Eigenvector alignment with the sine basis therefore cannot by itself predict spectral bias. It predicts the form the error takes once the eigenvalues have fixed its size, and spectral bias is a wavenumber-selective learning rule only when that projection is sine-aligned.

Acknowledgments

C.-Y.L. acknowledges the NSF via grant OAC-2608400. AC and CA acknowledge the support from the National Science Foundation (grant no. 2425667), DARPA APaQus program, the Sloan Foundation, and Schmidt Sciences, LLC. A.C. and C.-Y.L. designed the research. C.A. wrote the code and executed the research. Early versions of the code were provided by A.C. A.C. and C.A. wrote the paper, and all the authors analyzed the results and edited the manuscript.

Data and code availability

All data in this study are generated by the code: training forcings are sampled from the ensembles of Section 2.1 and the targets are computed exactly. The code for training the models, computing

the kernel diagnostics and reproducing every figure is available on GitHub at `https://github.com/cainsliewastaken/Spectral_Learning`.

References

[1] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[2] Zhi-Qin John Xu, Yaoyu Zhang, Tao Luo, Yanyang Xiao, and Zheng Ma. Frequency principle: Fourier analysis sheds light on deep neural networks. *Communications in Computational Physics*, 28(5):1746–1767, 2020.

[3] Ronen Basri, David Jacobs, Yoni Kasten, and Shira Kritchman. The convergence rate of neural networks for learned functions of different frequencies. In *Advances in Neural Information Processing Systems*, 2019.

[4] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems*, 2020.

[5] Ashesh Chattopadhyay, Y. Qiang Sun, and Pedram Hassanzadeh. Challenges of learning multi-scale dynamics with AI weather models: Implications for stability and one solution. *arXiv preprint arXiv:2304.07029*, 2023.

[6] William Briggs, Van Emden Henson, and Steve McCormick. *A Multigrid Tutorial*. SIAM, 2nd edition, 2000.

[7] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, 2018.

[8] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

[9] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

[10] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to PDEs. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

[11] Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why PINNs fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022.

[12] Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021.

[13] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[14] Sifan Wang, Hanwen Wang, and Paris Perdikaris. On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 384:113938, 2021.

[15] Inbar Seroussi, Asaf Miron, and Zohar Ringel. Spectral-bias and kernel-task alignment in physically informed neural networks. *Machine Learning: Science and Technology*, 5(3):035048, 2024.

[16] Siavash Khodakarami, Vivek Oommen, Nazanin Ahmadi Daryakenari, Maxim Beekenkamp, and George Em Karniadakis. Spectral bias in physics-informed and operator learning: Analysis and mitigation guidelines. *Computer Methods in Applied Mechanics and Engineering*, 461:119156, 2026.

[17] Siavash Khodakarami, Vivek Oommen, Aniruddha Bora, and George Em Karniadakis. Mitigating spectral bias in neural operators via high-frequency scaling for physical systems. *Neural Networks*, 193:108027, 2026.

[18] Zhi-Qin John Xu, Lulu Zhang, and Wei Cai. On understanding and overcoming spectral biases of deep neural network learning methods for solving PDEs. *Journal of Computational Physics*, 530:113905, 2025.

[19] Yongji Wang and Ching-Yao Lai. Multi-stage neural networks: Function approximator of machine precision. *Journal of Computational Physics*, 504:112865, 2024.

[20] Lénaïc Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[21] Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M. Roy, and Surya Ganguli. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the Neural Tangent Kernel. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[22] Blake Bordelon and Cengiz Pehlevan. Self-consistent dynamical field theory of kernel evolution in wide neural networks. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

[23] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

[24] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[25] Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in Neural Information Processing Systems*, 2019.

[26] Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 322–332, 2019.

[27] Gene H. Golub and Gérard Meurant. *Matrices, Moments and Quadrature with Applications*. Princeton University Press, 2010.

[28] Lin Lin, Yousef Saad, and Chao Yang. Approximating spectral densities of large matrices. *SIAM Review*, 58(1):34–65, 2016.

A Spectral bias in scalar regression

In scalar regression, the definition of spectral bias is unclear on its own. It depends on the operator being learned and how many coordinates it acts on in a target space.

A.1 Prescribed sine modes

Suppose the target is $u^*(x) = \sum_{k=1}^{n_\mathrm{m}} c_k \sin(k\pi x)$ and the model uses the same basis with trainable amplitudes, $u_\theta(x) = \sum_{k=1}^{n_\mathrm{m}} a_k \sin(k\pi x)$. With the design matrix $\Phi \in \mathbb{R}^{n_\mathrm{s} \times n_\mathrm{m}}$, $\Phi_{ik} = \sin(k\pi x_i)$, on the training points $x_1, \ldots, x_{n_\mathrm{s}}$, the empirical loss is $\mathcal{L}(\boldsymbol{a}) = \frac{1}{2}(\boldsymbol{a} - \boldsymbol{c})^\top \Phi^\top \Phi\, (\boldsymbol{a} - \boldsymbol{c})$, and gradient descent with step size η gives

$$\boldsymbol{a}^{n+1} - \boldsymbol{c} = (I - \eta\, G)\, (\boldsymbol{a}^n - \boldsymbol{c}), \qquad G := \Phi^\top \Phi. \tag{32}$$

A.2 Orthogonal sampling

If the training grid makes the sine basis orthogonal on the sample points, then $\Phi^\top \Phi = \gamma I$ for some $\gamma > 0$ and each coefficient evolves independently,

$$a_k^{n+1} - c_k = (1 - \eta\gamma)\, (a_k^n - c_k), \qquad k = 1, \ldots, n_\mathrm{m}. \tag{33}$$

All modes are learned at the same rate, which can be seen in Fig. A1(a).

A.3 Non-orthogonal sampling

If the sample points do not make the sine basis orthogonal, G is not diagonal and Eq. (32) mixes the coefficients. With $G = O\, \Lambda_G\, O^\top$, $\Lambda_G = \mathrm{diag}(\mu_1, \ldots, \mu_{n_\mathrm{m}})$ and O orthogonal, the components $\xi^n := O^\top(\boldsymbol{a}^n - \boldsymbol{c})$ evolve as

$$\xi_j^{\,n+1} = (1 - \eta\mu_j)\, \xi_j^{\,n}. \tag{34}$$

These eigenvectors are not sine modes, and as a result individual sine modes are coupled to each other and non-monotonic, as seen in Fig. A1(b). This is the same setup as Section 4, with the stacked kernel in place of G.

B Architectures and training details

All experiments use the discrete Poisson problem of Section 2.1 with $N = 128$ interior grid points, in double precision.

Ideal diagonal spectral operator. The model is

$$u_\theta = S\, \Lambda_\theta\, S^\top f, \qquad \Lambda_\theta = \mathrm{diag}(g_1, \ldots, g_{K_\mathrm{ret}}, 0, \ldots, 0), \tag{35}$$

with one trainable eigenvalue per retained sine mode and no other parameters, $K_\mathrm{ret} = 32$ retained modes, and parameters initialized at zero. Its Jacobian columns are the scaled sine modes $\hat{f}_k^{\,\mu}\, v_k$, so its modal kernel is exactly diagonal in the sine basis.

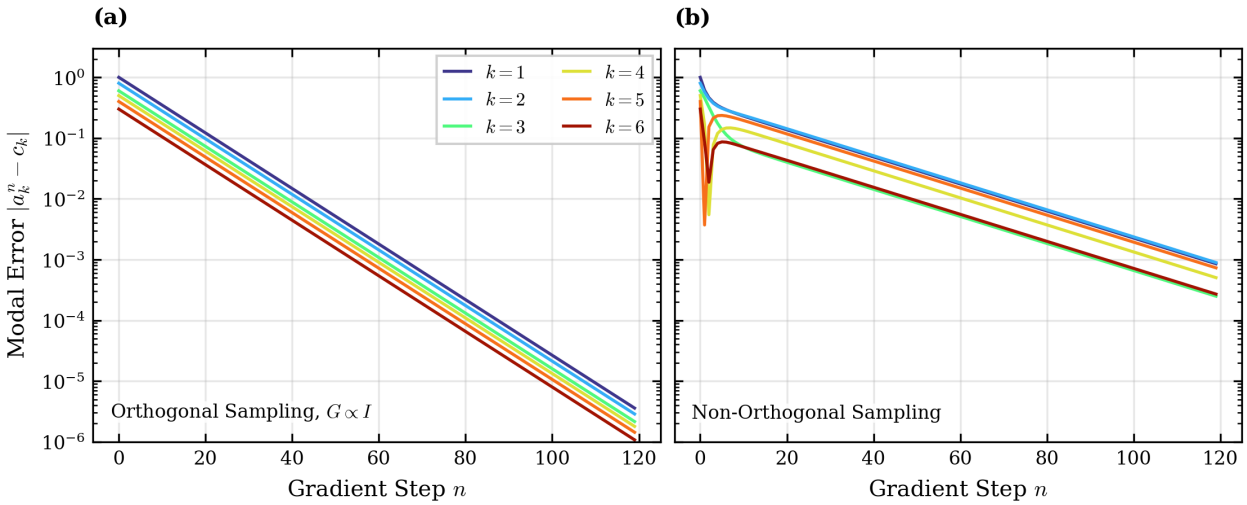


Figure A1: **Fixed sine-basis model: spectral order is governed by the data geometry through the Gram matrix**. (a) Under orthogonal sampling, G is diagonal with modes that decay at the same rate (Eq. 33). (b) Under non-orthogonal sampling, G couples the sine modes (Eq. 32), so its coefficients are non-monotonic and the rates act on the eigenvectors of G (Eq. 34).

Fourier neural operators. The FNOs lift the grid signal to $d_c = 32$ channels with a pointwise linear layer, apply $D = 3$ FNO layers, and project back to one channel with a pointwise linear layer. Each FNO layer sums a pointwise linear channel-mixing branch and a spectral branch that transforms to sine coefficients, applies an independent trainable $d_c \times d_c$ weight to each retained mode, and transforms back. The runs retain all 128 modes. The nonlinear FNO applies a GELU after each layer and the linear FNO applies none. Spectral weights are initialized from a zero-mean Gaussian with standard deviation 0.05. The lift, local, and projection layers carry biases, whose spatially constant output underlies the parity selection rule of Appendix D.

Dense networks. The GELU dense network is a multilayer perceptron acting on the 128-dimensional grid vector, of depth 4 and width 512, with GELU activations between layers. The deep linear dense network has the same depth and width, no nonlinearities, and no bias terms, with weights initialized from a zero-mean Gaussian with standard deviation 0.05. Its product of weight matrices realizes a single linear operator.

Training. All models are trained with Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon_A = 10^{-8}$) on the mean-squared error, with batch size 512, freshly sampled forcings at every step, gradient-norm clipping at 1, and 12000 optimization steps. The learning rate is 10^{-2} for the ideal diagonal spectral operator and 5×10^{-5} for all other models, annealed by a cosine schedule to 1% of its initial value over the run. Error spectra are computed on a fixed validation set of 4096 forcings per ensemble.

Kernel diagnostics. Unless stated otherwise, kernel statistics are averaged over the last 100 timesteps. The stacked kernel $K = \frac{2}{BN} J \Pi J^\top$ is applied using a Jacobian–vector product that contains the cross terms. The terms in Eq. (26) are computed via error-loaded Lanczos [27, 28], which is initialized at e with full reorthogonalization and run for $m = 2000$ steps. Where appropriate, all metrics are averaged over 4 batches of size $B = 2048$. In Section 4.3, the modal kernel metrics are computed using the batch-averaged kernel $\bar{K} = \frac{2}{N} \bar{J} \Pi \bar{J}^\top$ on a fixed batch of 32.

Validity of the diagonal Adam model. Eq. (20) makes two approximations: it drops momentum and holds Π fixed. Momentum averages the gradient over $(1-\beta_1)^{-1}$ steps, so only the fast modes with $\eta\lambda_\alpha \gtrsim 1-\beta_1$ are affected. As the error lies within only the slow modes, this is not impactful. The preconditioner Π moves at a scale of $(1-\beta_2)^{-1} = 1000$ steps, so the final 100-step average is effectively constant as well. Combined, the linearization holds provided the error-carrying rates stay below $\eta\lambda^* \leq 1.3 \times 10^{-4}$ and only the last 100 steps are used.

C The hierarchy under white and smooth forcing

Figure A2 shows the other models absent from Fig. 3. The nonlinear FNO plateaus above machine epsilon and reproduces the structure of Section 4.2, while the linear models learn the operator and so lack it.

Linear models. Unlike the nonlinear models, the linear models carry no slow tail. The deep linear dense network trains to machine epsilon, leaving a numerical residual with 79% of its error in the kernel's null space (Fig. A2e,f); the ideal diagonal spectral operator likewise cannot represent its truncated modes, so its residual sits in the null space (Fig. A2g,h). The linear FNO differs: its error is a single slow direction near λ^* (Fig. A2c,d) that the cross terms only redistribute.

Smooth forcing. Unlike white forcing, every model trained under smooth forcing retains slow-mode error (Fig. A3). The deep linear dense network is no longer at machine epsilon and has error in the high modes, as the nonlinear kernel theory predicts.

Alignment under white forcing. For white forcing (Fig. A4), the FNO eigenvectors lose the alignment they had under smooth forcing and begin to resemble the GELU and deep linear dense networks. This results in some of the expected channel mixing, though they are still more aligned with the sine basis than those models.

Eigenvalue magnitudes and the optimizer state. The Adam preconditioner of Eq. (18) causes the spectrum to grow as the loss falls, so the rate axes in Fig. A2 span different ranges. This happens because the second-moment estimates decay, which pushes up the diagonal values of Π. The eigenvalues of $K = \frac{2}{BN} J\Pi J^\top$ therefore reflect both the Jacobian and the optimizer state. As a result, the models' absolute rates should not be compared directly; the measure should instead be read relative to λ^*.

D A parity selection rule for the FNO kernels

The alternating banded structure of the FNO overlap matrices in Fig. 4(a) is a reflection symmetry of the problem and the architecture. Let $R \in \mathbb{R}^{N\times N}$ be the reflection about the midpoint, $(Rf)_j = f_{N+1-j}$; the sine modes have alternating parity under R,

$$R\, v_k = (-1)^{k+1} v_k, \tag{36}$$

so odd-k modes are reflection-even and even-k modes reflection-odd, and the discrete Poisson operator commutes with R. The FNOs are reflection-equivariant at every parameter value: the spectral weights, lift, projection, and pointwise nonlinearity all commute with R, and the channel mixing acts identically at every grid point, so without biases or nonlinearity the FNO is diagonal in

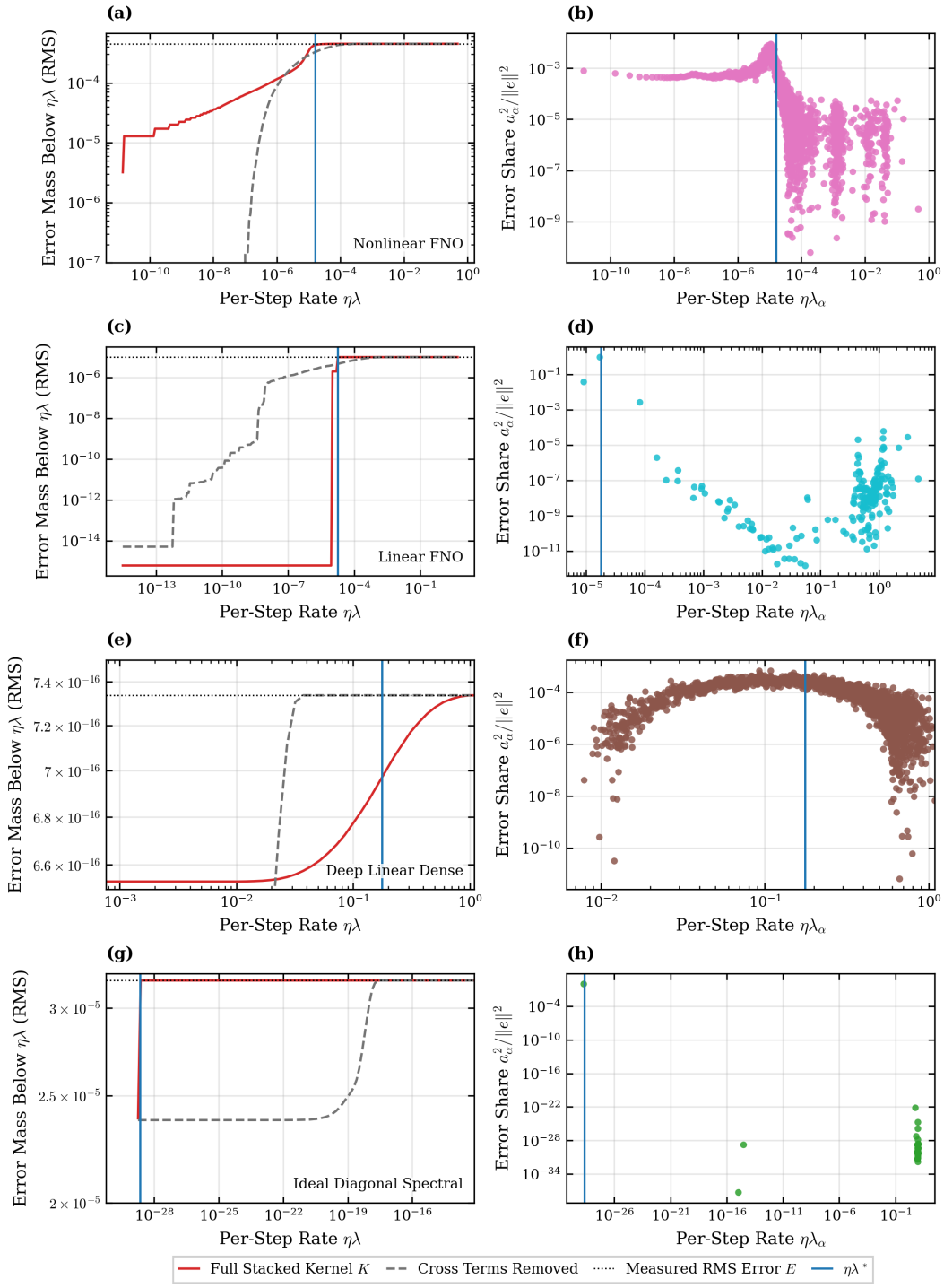


Figure A2: **The error-loaded spectral measure of Fig. 3, repeated for the other four models** (white forcing; rows are nonlinear FNO (a,b), linear FNO (c,d), deep linear dense (e,f), and ideal diagonal spectral (g,h)). Left column, cumulative error mass $\Gamma(\lambda)$ of Eq. (26) for the full stacked kernel (red) and for the block-diagonal kernel with the cross terms removed (grey dashed), with the measured RMS error E (dotted) and $\eta\lambda^*$ with λ^* of Eq. (27) (vertical line). Right column, each Ritz pair at its per-step rate $\eta\lambda_\alpha$ against its error share $a_\alpha^2/\|e\|^2$. The rate axes differ across rows (Appendix C).

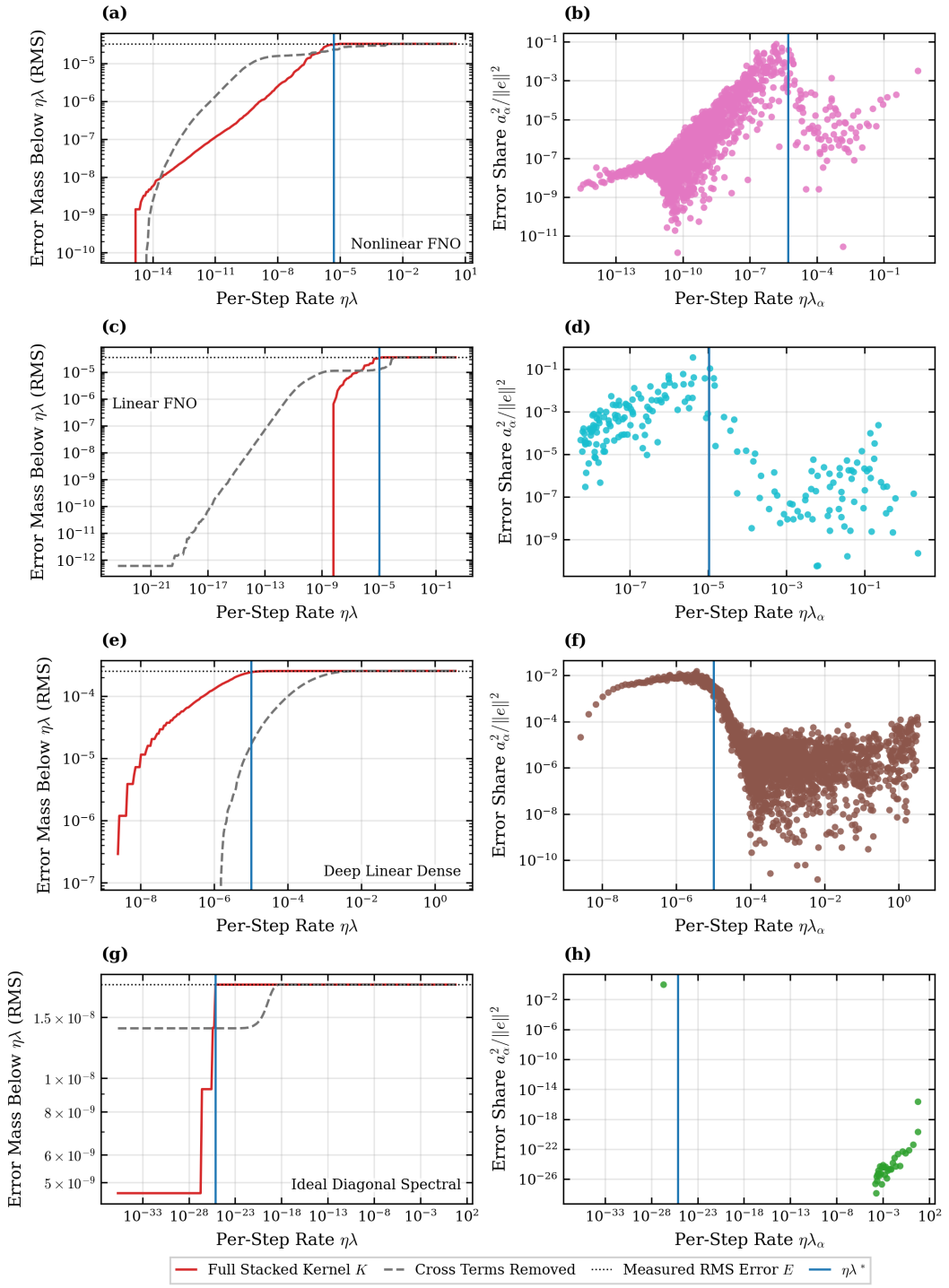


Figure A3: **The error-loaded spectral measure of Fig. A2, repeated under smooth forcing**. Rows are nonlinear FNO (a,b), linear FNO (c,d), deep linear dense (e,f), and ideal diagonal spectral (g,h); the GELU dense network is shown in Fig. 3(a,c). Lines and markers as in Fig. A2.

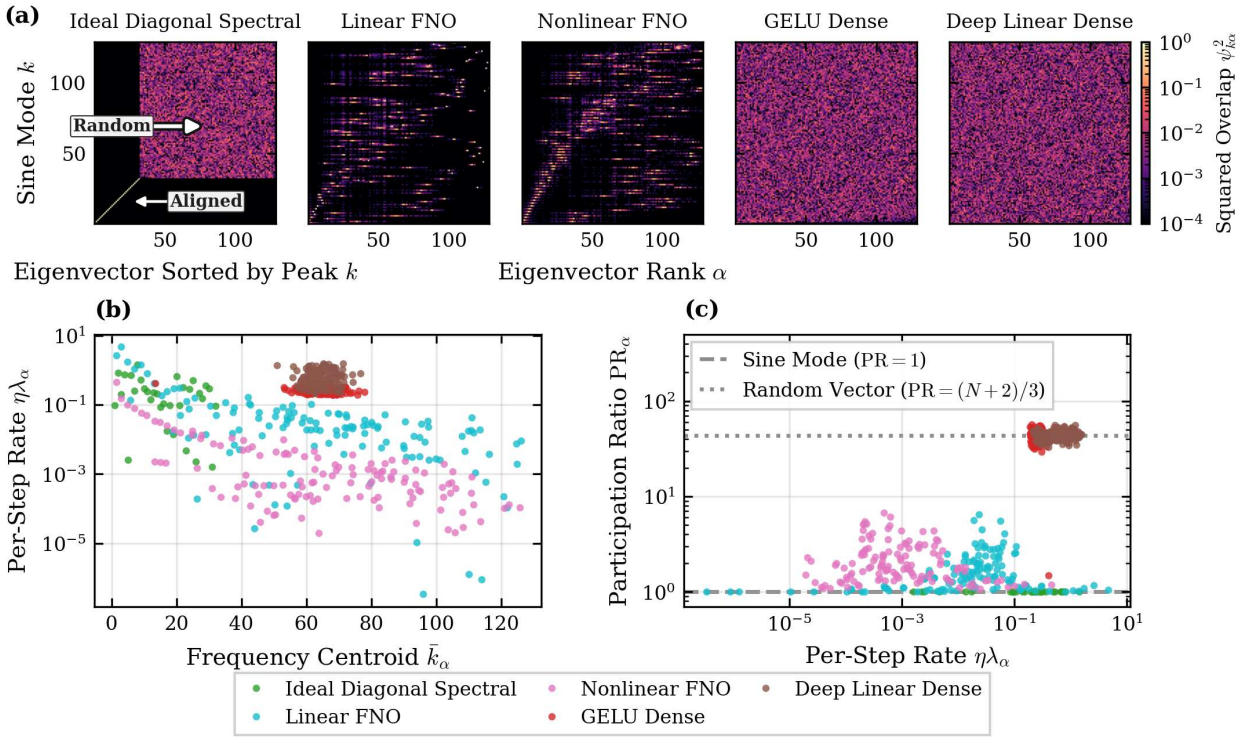


Figure A4: **Alignment with the sine basis under white forcing** (late-training batch-averaged kernel $\bar{K}$; panels as in Fig. 4). (a) Squared overlaps $\psi^2_{k\alpha}$ between the eigenvectors of $\bar{K}$ and the sine modes, sorted as in Fig. 4(a). (b) Per-step rate $\eta\lambda_\alpha$ against the frequency centroid $\bar{k}_\alpha$. (c) Participation ratio PR_α against per-step rate, with the reference lines of Fig. 4(c).

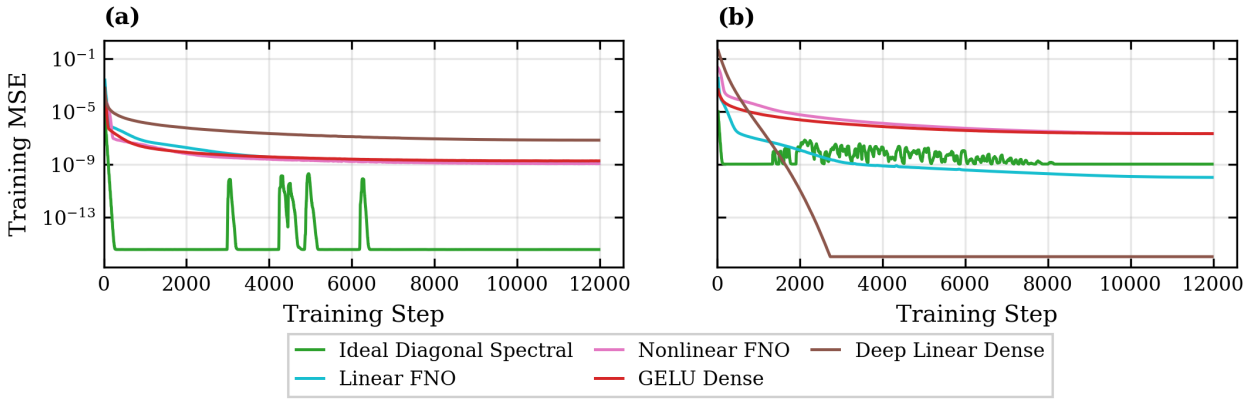


Figure A5: **Training loss of the hierarchy** under smooth (a) and white (b) forcing.

the sine basis. The only coupling in the linear FNO comes from its biases, which add a reflection-even constant with content on odd modes only, so coupling acts within one parity sector. The modal kernel is equivariant in expectation and for a reflection-symmetric batch, but per-sample fluctuations break the symmetry, so its dominant eigenvectors concentrate in one parity sector ($\psi^2_{k\alpha} \approx 0$ for k of the wrong parity) and appear as bands rather than exact zeros. The dense networks have no reflection symmetry and show no bands. The same biases account for the linear FNO's residual under white forcing. Its output at $f = 0$ lies on odd modes and carries 99.99% of

the squared error, almost all at $k = 125$ and $k = 127$, set by the spectral weights downstream of the biases. The offset is the slowest direction of the kernel (Fig. A2(d)) and the only place its error exceeds U_k, the isolated spike in Figs. 1(f) and 2(b,d); its mean square, about 10^{-10}, equals the final training loss in Fig. A5(b). The residual is therefore an input-independent offset rather than an error in the learned linear map.