LEARNING SPECTRALLY OPTIMISED MESH-FREE DISCRETISATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Numerical methods for partial differential equations (PDEs) discretise differential operators, ideally reproducing the action of the continuous operators across all wavenumbers permitted by a given discretisation. Spectral-type methods approach this ideal, but rely on structured grids or high-order meshes that are hard to generate for complex geometries. In contrast, mesh-free methods are geometrically flexible, yet how faithfully they reproduce the operator across resolved scales is strongly influenced by a heuristically chosen kernel, which selects one of many weight sets satisfying the same consistency conditions without regard to the resulting spectral response. To address this, we introduce Spectrally optimised Neural Discretisations (SpeND), a framework that learns the map from local stencil geometry to discretisation weights on unstructured point clouds. A projection layer constrains every predicted stencil to the affine set defined by the discrete moment conditions, so polynomial consistency, and hence formal order of accuracy, holds exactly. Since consistency is enforced by the architecture, the weights can additionally be optimised for accuracy on a prescribed function space. Herein, we use Fourier modes and target the exact differentiation response over a chosen wavenumber band, so training is unsupervised. The loss function is a design interface: changing how it weights wavenumbers yields operators with distinct accuracy profiles. The trained discrete differential operators are PDE-agnostic and are applied without retraining to the Poisson, Burgers and Navier–Stokes equations, to near-boundary stencils, and across resolutions. At the same order and resolution, SpeND matches or improves on established mesh-free discretisations, and has been shown to reduce the wall-clock time between $3 - 20\times$ to achieve equivalent error as the baselines.

1 INTRODUCTION

Partial Differential Equations (PDEs) play an essential role in science and engineering (Stocker et al., 2013; Martínez et al., 2017; Johnston et al., 2006), however, analytical solutions are rarely available for problems of practical interest. Consequently, numerical methods are the standard tools for high-fidelity simulation, data generation, and the study of new physical regimes. Their reliability rests on a guarantee structure that is independent of the PDE being solved: a discretisation that is consistent, meaning it approximates the continuous operator, and stable, meaning errors remain bounded, is convergent. For well-posed linear problems this relationship is formalised by the Lax equivalence theorem (Lax & Richtmyer, 1956).

Global spectral methods (Orszag, 1971; Gottlieb & Orszag, 1977) sit at the accuracy ceiling of numerical methods: the symbol of the discrete operator (i.e. its action on each Fourier mode) matches that of the continuous one across the wavelengths permitted by the discretisation, so a target accuracy is reached with far fewer degrees of freedom than any fixed-order scheme. This accuracy is conditional on structured collocation points, which restricts global spectral methods to simple geometries (Boyd, 2001). Domain decomposition approaches such as spectral element methods relax this constraint by partitioning the domain into elements, at the cost of the global symbol match. Geometric flexibility is then limited by mesh generation, where high-order meshes remain difficult to produce for complex geometries (Karniadakis & Sherwin, 2005).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Unstructured strong-form methods such as Smoothed Particle Hydrodynamics (SPH) (Lucy, 1977), radial basis function-finite difference methods (RBF-FD) (Tolstykh, 2000; Bayona & Kindelan, 2013; Bayona et al., 2015), and the local anisotropic basis function method (LABFM) (King et al., 2020) approximate discrete differential operators solely based on local neighbour information and through a compactly supported kernel. The kernel, however, is chosen heuristically, so how faithfully the discrete operator reproduces the continuous one across resolvable scales is fixed by a modelling choice made that offers little control of that objective.

Machine learning (ML) has been applied at several levels of the PDE solution pipeline. Neural operators and learned surrogates replace the solver with a map between function spaces (Lu et al., 2021; Li et al., 2021; Anandkumar et al., 2019), achieving large speedups but yielding a model tied to the family of PDEs it was trained on, and without the convergence guarantees of classical numerical methods. Closer to the present work, learned discretisations retain the numerical solver and learn only the stencil coefficients (Bar-Sinai et al., 2019; Kochkov et al., 2021), recovering coarse-grid accuracy while preserving the solver structure. These methods, however, are restricted to structured grids and are trained with supervision against high-resolution reference solutions, which ties the learnt discretisation to flows with similar local structures.

In this context, we introduce Spectrally optimised Neural Discretisations (SpeND), a framework in which a neural network predicts mesh-free stencil weights on unstructured point clouds. The framework has three key properties. (i) **Guaranteed consistency order.** A projection layer constrains the network output to the affine set satisfying the discrete moment conditions, so the target consistency order holds exactly for every predicted stencil. (ii) **Numerically grounded training.** Training minimises the deviation between the analytically known modal response of the continuous and the discretised operator over a prescribed band, yielding an operator that is accurate across the desired spectrum rather than for a particular flow or PDE. (iii) **Generalisation.** A single trained operator transfers without retuning across resolutions and PDEs.

2 Background & Related Work

2.1 Numerical Approximation of Differential Operators

Let the spatial domain $\Omega \subset \mathbb{R}^d$ be represented by a set of collocation points $\mathcal{P} := \{\mathbf{x}_i\}_{i=1}^{P} \subset \Omega$, where $\mathbf{x}_i$ denotes the position of node i. For a continuous differential operator D acting on a field ϕ, a local discrete approximation at node i takes the form[1]

$$L^D[\phi(\mathbf{x}_i)] = \sum_{j \in \mathcal{N}_i} \phi_{ji}\, w_{j,i}^D, \tag{1}$$

in which L^D is the discrete counterpart of D, the index set $\mathcal{N}_i$ collects the neighbours forming the computational stencil of node i, each $w_{j,i}^D$ is the weight assigned to neighbour j within that stencil, and differences are written $(\cdot)_{ji} := (\cdot)_j - (\cdot)_i$. With suitable choices of stencil and indexing, equation 1 recovers a wide range of numerical methods, among them finite difference (FD) (Courant et al., 1967), finite elements (FE) (Clough, 1990) and SPH. What distinguishes one discretisation from another is the construction of the neighbourhoods $\mathcal{N}_i$ and of the weights $w_{j,i}^D$.

Consistency characterises how the discrete operator approaches the continuous one as the node spacing s tends to zero. A discrete operator L^D is consistent to order p if it reproduces every Taylor coefficient of D up to polynomial degree p, so that, for an operator D of order m, the truncation error is $\mathcal{O}(s^{p+1-m})$. This can be written as a linear system for the stencil weights. Collecting the stencil weights of node i into the vector $\mathbf{w}_i^D := (w_{j,i}^D)_{j \in \mathcal{N}_i}$, substituting the multivariate Taylor expansion of ϕ_{ji} about $\mathbf{x}_i$ into equation 1 and matching the discrete and continuous operators term by term yields the linear moment system

$$\mathbf{V}_i\, \mathbf{w}_i^D = \mathbf{d}^D, \qquad [\mathbf{V}_i]_{\boldsymbol{\alpha},j} = \frac{\mathbf{x}_{ji}^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!}, \qquad [\mathbf{d}^D]_{\boldsymbol{\alpha}} = \frac{D\,\mathbf{x}^{\boldsymbol{\alpha}}\big|_{\mathbf{x}=\mathbf{0}}}{\boldsymbol{\alpha}!}. \tag{2}$$

Rows are indexed by the $N_p = \binom{p+d}{d} - 1$ multi-indices $\boldsymbol{\alpha}$ with $1 \le |\boldsymbol{\alpha}| \le p$, one per Taylor monomial, and columns by the neighbours $j \in \mathcal{N}_i$, so that $\mathbf{V}_i \in \mathbb{R}^{N_p \times |\mathcal{N}_i|}$ and $\mathbf{w}_i^D \in \mathbb{R}^{|\mathcal{N}_i|}$. The

[1] A full list of symbols is given in the Nomenclature.

entries of the stencil moment vector $\mathbf{d}^D$ record the action of D on each monomial, so the stencil is required to reproduce those coefficients exactly and to eliminate all remaining ones up to degree p. For instance, taking $d = 2$, $p = 2$ and ordering $\boldsymbol{\alpha}$ as $(1,0),(0,1),(2,0),(1,1),(0,2)$ gives $\mathbf{d}^{\partial_x} = (1,0,0,0,0)^\top$ for the first derivative and $\mathbf{d}^{\nabla^2} = (0,0,1,0,1)^\top$ for the Laplacian.

When $N_p = |\mathcal{N}_i|$ and $\mathbf{V}_i$ is non-singular, the moment system admits a unique solution. On a structured grid this recovers the finite difference method. Mesh-free discretisations on unstructured point clouds instead operate with $N_p < |\mathcal{N}_i|$ to improve the conditioning of $\mathbf{V}_i$ under irregular node spacing, so the system is underdetermined and consistency alone does not determine the weights.

2.2 Classical Solvers

Spectral, spectral-like, and spectrally-optimised frameworks. Global spectral methods make differentiation exact for the resolvable band allowed by the grid and converge exponentially, but they rely on a structured tensor-product grid and on boundary conditions compatible with the basis (Boyd, 2001). Spectral element methods recover geometric flexibility by partitioning the domain into elements carrying local high-order bases (Karniadakis & Sherwin, 2005), yet the resolved band is no longer represented exactly, and the required high-order mesh is often the bottleneck for such schemes. Global radial basis function collocation removes the mesh entirely and retains near-spectral accuracy on scattered nodes, at the cost of dense operators, ill-conditioning that worsens as accuracy improves, and $\mathcal{O}(P^2)$ operations per time step (Fornberg & Lehto, 2011).

Compact schemes abandon the exact symbol and instead match the modified wavenumber of a narrow stencil to the exact one over a prescribed band, achieving spectral-like resolution at low bandwidth (Lele, 1992), an idea since extended to scattered nodes by compact moving least squares (Trask et al., 2016) and compact LABFM (Broadley et al., 2026a). In all cases the derivative is defined implicitly, requiring a sparse global solve at every evaluation, which couples the whole domain and removes the locality that makes explicit stencils cheap to refine and parallelise.

Closest to our objective are explicit schemes whose coefficients are chosen to minimise modal error rather than to maximise formal order (Tam & Webb, 1993), though the construction is restricted to structured stencils. The multi-kernel approach (Broadley et al., 2026b) also targets the modal response, but operates at a different level; it combines several instances of a given mesh-free discretisation. SpeND acts on the discretisation itself and could equally serve as the base scheme in such a combination.

Local mesh-free discretisation methods construct each differential operator by summation over a compact stencil of scattered neighbours, requiring no connectivity. This makes them geometrically flexible, tolerant of large deformation, and highly parallelisable, since every stencil is built independently. SPH (Lucy, 1977) was among the first such methods, but its weights derive from a fixed radial smoothing kernel and it is at best zeroth order consistent. A broader class of local methods instead imposes the moment conditions of equation 2 directly when constructing the weights. Notable examples are the generalised finite differences method (GFDM) (Jensen, 1972), generalised moving least squares (GMLS) (Trask et al., 2017), RBF-FD (Tolstykh, 2000) and LABFM (King et al., 2020).

In all the mesh-free methods mentioned above, the weights are built from a heuristic radial function whose properties can significantly affect the solution quality (and consequently how well the discrete operator approximates the resolvable band) (Dehnen & Aly, 2012; Le Borne & Leinen, 2023). Because the response cannot be prescribed, accuracy over the resolvable band can only be improved indirectly. Raising the order of consistency improves agreement between the discrete and continuous operators in the low-wavenumber limit, at the cost of N_p growing combinatorially with p and d, while refining the node spacing lowers the normalised wavenumbers at which the solution must be resolved, at a cost of $\mathcal{O}(s^{-d})$ nodes. Neither acts on the response across the band, leaving the stencil freedom admitted by equation 2 unexploited.

2.3 Neural solvers

Many approaches have been proposed for solving PDEs with machine learning. Physics-informed neural networks (Raissi et al., 2019) fit the solution of a single PDE instance by minimising its residual, and in their standard form must be retrained for each new instance; they also inherit the

spectral bias of the network, which limits them on multiscale problems (Wang et al., 2022; Rahaman et al., 2019). Neural operators (Lu et al., 2021; Li et al., 2021) instead learn a mapping between function spaces from example solutions, and are correspondingly tied to the equation family and boundary conditions on which they were trained. Both approaches have been applied successfully across a wide range of PDEs, but physical consistency and stability are properties of a trained model on a particular problem, assessed empirically rather than guaranteed by construction.

Hybrid solvers. A second line preserves the classical solver and learns components within it (Greenfeld et al., 2019; Satorras et al., 2019; Hsieh et al., 2019; Um et al., 2021; Brandstetter et al., 2023), targeting problems from the acceleration of iterative solvers to the stability of autoregressive rollouts. Closest to this work are **learned discretisations**. Bar-Sinai et al. (2019) learn finite volume stencil weights on coarse Cartesian grids, predicting them from the local field values with a convolutional network and training against the coarse-grained time derivative of a high-resolution reference. Consistency is imposed by construction, the prediction being a fixed low-order stencil plus a component in the null space of a truncated moment system. Zhuang et al. (2021); Kochkov et al. (2021); de Romémont et al. (2026) extend this to more complex flows, and Dresdner et al. (2023) learn corrections to a spectral solver on periodic grids. While effective, these methods are restricted to structured grids and are supervised on a high-resolution reference solution, which ties the learned discretisation to the equation and similar flow structures used to train it. The weights are also a function of the instantaneous field, so the network must be evaluated at every step of the simulation.

Beyond structured grids. Choi et al. (2025) replace the classical polynomial bases of the finite element method with locally learned ones that transfer across geometries, mesh types and boundary conditions. The method remains mesh-based, requires data generated over many meshes and PDEs, and its convergence and stability remain an open problem. In the mesh-free setting, Starepravo et al. (2026) learn discretisations by imposing the Taylor moment conditions as a loss penalty, removing the need for PDE data at training time. Thus, the resulting weights are only approximately consistent, with moment residuals well above machine precision, which restricts the framework to low orders of approximation. Furthermore, as in classical mesh-free schemes, the modal response is left to whatever the construction yields rather than being targeted.

3 METHOD

3.1 PROBLEM SETUP

Let $\mathcal{P}$ be an unstructured point cloud in Ω. We associate to each point $\mathbf{x}_i$ the local neighbourhood $\mathcal{N}_i := \{j : \mathbf{x}_j \in \mathcal{P},\ \|\mathbf{x}_{ji}\|_2 \leq R_i\}$, where R_i is the Euclidean distance from $\mathbf{x}_i$ to the farthest of its N nearest nodes, $\mathbf{x}_i$ included, so that $|\mathcal{N}_i| = N$. Fixing the neighbour count rather than the support radius is common practice in mesh-free frameworks (Benito et al., 2001; Flyer et al., 2020). Our objective is to construct, at each node, a discrete approximation L^D of a differential operator D acting on ϕ, taking the form of the weighted sum equation 1 over $\mathcal{N}_i$. The approximation is therefore fully specified by the stencil weights $\mathbf{w}_i^D$.

In SpeND, we formulate the construction of the stencil weights as a learning problem. Rather than fixing the map from local geometry to weights through a kernel and a closure, we parametrise it by a neural network and constrain its output so that consistency holds exactly.

The network acts on the local geometry alone. Neighbour positions are normalised onto the unit circle $\bar{\mathbf{x}}_{ji} = \mathbf{x}_{ji}/R_i$, so the learned map depends only on the local node arrangement, and can be applied to different resolutions. A network f_θ maps the normalised positions to candidate weights $\tilde{\mathbf{w}}_i^D = f_\theta(\{\bar{\mathbf{x}}_{ji}\}_{j\in\mathcal{N}_i})$, which are then projected by Π_i onto the affine set of solutions of the moment system $\bar{\mathbf{V}}_i\bar{\mathbf{w}}_i^D = \mathbf{d}^D$, i.e. equation 2 assembled from the normalised positions. The projection is affine and determined entirely by $\bar{\mathbf{V}}_i$, hence every stencil predicted by the network is consistent to order p by construction. The projected weights are returned to the physical scale of the point cloud by a factor R_i^{-m}, giving

$$\mathbf{w}_i^D = R_i^{-m}\,\Pi_i\big[f_\theta(\{\bar{\mathbf{x}}_{ji}\}_{j\in\mathcal{N}_i})\big]. \tag{3}$$

While the projection layer ensures the predicted weights are consistent, the system remains underdetermined for $N > N_p$. SpeND acts on this freedom, training θ so that the operator assembled from the projected weights optimises the modal response of D across the resolved band.

3.2 Consistency Constraints – Moment Projection

Since $N > N_p$ in the regime of interest, $\bar{\mathbf{V}}_i$ is wide and, provided it has full row rank, the consistent weights form an affine subspace of dimension $N - N_p$. We map the candidate weights $\tilde{\mathbf{w}}_i^D$ onto this subspace through the minimum-norm correction

$$\bar{\mathbf{w}}_i^D = \Pi_i\big[\tilde{\mathbf{w}}_i^D\big] = \tilde{\mathbf{w}}_i^D - \bar{\mathbf{V}}_i^+\big(\bar{\mathbf{V}}_i\,\tilde{\mathbf{w}}_i^D - \mathbf{d}^D\big)\,, \tag{4}$$

where $\bar{\mathbf{V}}_i^+$ is the Moore–Penrose pseudoinverse. The correction vanishes whenever $\tilde{\mathbf{w}}_i^D$ is already consistent, so consistency of order p holds for any network output and is a property of the architecture rather than of the converged parameters. Since the map is affine, its Jacobian is the orthogonal projector $\mathbf{P}_i = \mathbf{I} - \bar{\mathbf{V}}_i^+\bar{\mathbf{V}}_i$ onto $\ker \bar{\mathbf{V}}_i$, and differentiating through the layer requires no implicit solve.

3.3 Training

With consistency enforced by the projection, the objective determines which point of the consistent subspace is selected. Since both the network and the objective depend only on the normalised geometry, we write the objective in the normalised coordinates of a single stencil, with $\mathbf{x}_i$ at the origin. Let Φ be a class of test functions and D the continuous operator that L^D approximates. For each node i and each $\phi \in \Phi$, the stencil residual is

$$r_i[\phi] \;=\; \sum_{j\in\mathcal{N}_i} \phi_{ji}\,\bar{w}_{j,i}^D(\theta) \;-\; D[\phi](\mathbf{0}), \tag{5}$$

where $\bar{w}_{j,i}^D(\theta)$ are the projected network outputs, ϕ is evaluated at the normalised positions $\bar{\mathbf{x}}_{ji}$, and $D[\phi](\mathbf{0})$ is available analytically. No reference solutions are therefore required at any point in training.

In this work we take Φ to be the Fourier modes $\phi_{\mathbf{k}}(\mathbf{x}) = e^{\mathrm{i}\mathbf{k}\cdot\mathbf{x}}$, with $\mathbf{k}$ measured in normalised coordinates, so that equation 5 measures the departure of the discrete operator from the exact continuous one at wavenumber $\mathbf{k}$. Optimising at a single wavenumber would leave the response at all others unconstrained. We therefore optimise the modal response over a band, $0 \le \|\mathbf{k}\| \le \eta\, k_{\mathrm{Ny}}$, where $\eta \in (0, 1]$ and $k_{\mathrm{Ny}} = \pi/\bar{s}_N = \sqrt{\pi N}$ is the Nyquist wavenumber (i.e. the mininum wavenumber a given discretisation can resolve) at the mean normalised spacing $\bar{s}_N = \sqrt{\pi/N}$, obtained for $d = 2$ by distributing the N stencil nodes uniformly over the unit disk.

The residual is a single complex number per mode. We make this concrete for $D = \partial_x$; the remaining operators follow analogously (Appendix C). The continuous operator returns $\partial_x e^{\mathrm{i}\mathbf{k}\cdot\mathbf{x}}\big|_{\mathbf{x}=\mathbf{0}} = \mathrm{i}k_x$, while the discrete operator returns its modal response $\hat{L}_i^{\partial_x}(\mathbf{k}) := \sum_{j\in\mathcal{N}_i} \bar{w}_{j,i}^{\partial_x}\big(e^{\mathrm{i}\mathbf{k}\cdot\bar{\mathbf{x}}_{ji}} - 1\big)$. We then divide the discrete response by i and define the effective wavenumber $k_{\text{eff}}(\mathbf{k}) := -\mathrm{i}\,\hat{L}_i^{\partial_x}(\mathbf{k})$ (the stencil index is omitted for brevity), the wavenumber the discrete operator reproduces when differentiating $\phi_{\mathbf{k}}$. An exact discrete operator would give $k_{\text{eff}} = k_x$, so the deviation of k_{eff} from k_x measures the error of the discretisation at that mode. Separating real and imaginary parts,

$$k_{\text{eff}}(\mathbf{k}) = \sum_{j\in\mathcal{N}_i} \bar{w}_{j,i}^{\partial_x}\sin(\mathbf{k}\cdot\bar{\mathbf{x}}_{ji}) \;+\; \mathrm{i}\sum_{j\in\mathcal{N}_i} \bar{w}_{j,i}^{\partial_x}\,[1 - \cos(\mathbf{k}\cdot\bar{\mathbf{x}}_{ji})]\,. \tag{6}$$

The real part governs the phase speed of the mode, $\mathrm{Re}\{k_{\text{eff}}\}\,\mathrm{sign}(k_x) < |k_x|$ makes the mode lag the exact solution and $\mathrm{Re}\{k_{\text{eff}}\}\,\mathrm{sign}(k_x) > |k_x|$ makes it lead, exceeding the physical propagation speed (Kim & Lee, 1996). The imaginary part governs the amplitude: for advection with positive velocity, $\mathrm{Im}\{k_{\text{eff}}\} < 0$ damps the mode, whereas $\mathrm{Im}\{k_{\text{eff}}\} > 0$ injects energy and usually has to be suppressed by hyperviscous stabilisation (Jameson et al., 1981); see Appendix C.3.

Both parts of k_{eff} vary continuously with wavenumber magnitude and propagation angle, so we evaluate them on a finite set of modes. Let $\mathcal{K} \subset \{\mathbf{k} : \|\mathbf{k}\| \le \eta k_{\mathrm{Ny}},\, k_y \ge 0\}$ be a fixed set

of wavenumbers discretising the resolved half-disk. The omitted half is determined by the parity of equation 6, $\mathrm{Re}\{k_{\text{eff}}(-\mathbf{k})\} = -\mathrm{Re}\{k_{\text{eff}}(\mathbf{k})\}$ and $\mathrm{Im}\{k_{\text{eff}}(-\mathbf{k})\} = \mathrm{Im}\{k_{\text{eff}}(\mathbf{k})\}$, so no information is lost. The per-stencil dispersion loss is

$$\mathcal{L}_i^{\text{disp}} = \frac{1}{|\mathcal{K}|} \sum_{\mathbf{k}\in\mathcal{K}} \sigma(\mathbf{k})\, \omega_>(\mathbf{k})\, \left(\mathrm{Re}\{k_{\text{eff}}(\mathbf{k})\} - k_x\right)^2, \qquad \omega_>(\mathbf{k}) = \begin{cases} \lambda_> & \mathrm{Re}\{k_{\text{eff}}(\mathbf{k})\}\ \mathrm{sign}(k_x) > |k_x|, \\ 1 & \text{otherwise}, \end{cases} \tag{7}$$

with $\lambda_> > 1$ and $\sigma(\mathbf{k}) = \max(|k_x|/k_{\text{Ny}}, \epsilon)^{-2}$, normalised to unit mean over $\mathcal{K}$. The two factors act independently; $\omega_>$ breaks the symmetry of the squared error, penalising $\lambda_>$ times more for a leading response, while σ converts the objective from absolute to relative error, so that modes with a large exact response do not dominate the accuracy a solver depends on at small k_x. The floor ϵ bounds the weight where the exact response vanishes, avoiding blow-up as $|k_x|/k_{\text{Ny}} \to 0$.

We penalise dissipation with the same asymmetric form, now penalising λ_+ times more for amplification,

$$\mathcal{L}_i^{\text{diss}} = \frac{1}{|\mathcal{K}|} \sum_{\mathbf{k}\in\mathcal{K}} \omega_+(\mathbf{k})\, \left(\mathrm{Im}\{k_{\text{eff}}(\mathbf{k})\}\right)^2, \qquad \omega_+(\mathbf{k}) = \begin{cases} \lambda_+ & \mathrm{Im}\{k_{\text{eff}}(\mathbf{k})\} > 0, \\ 1 & \text{otherwise}. \end{cases} \tag{8}$$

The unweighted branch still penalises damping, since excess dissipation removes energy from resolved modes, but the asymmetry reflects that the two failures are not comparable: damping degrades accuracy, whereas amplification is a stability failure. We apply no relative weighting here, since the exact imaginary response is zero.

The network predicts stencil weights for arbitrary local geometries, so the objective is averaged over a corpus of stencils sampled from point clouds generated in advance (Appendix A),

$$\mathcal{L}(\theta) = \mathbb{E}_{i\sim\mathcal{S}} \left[\mathcal{L}_i^{\text{disp}} + \gamma\, \mathcal{L}_i^{\text{diss}} \right], \tag{9}$$

where $\mathcal{S}$ is the corpus of stencils and $\gamma > 0$ balances the two error types. Every term is computed from the predicted weights and the exact response of D, so training requires no PDE solutions at any point.

3.4 Architecture

The contribution of this work lies in the discretisation framework rather than the network, so the architecture is kept deliberately simple. We instantiate f_θ as an adapted PointNet (Qi et al., 2017) acting on each stencil independently, with shared per-point maps and a symmetric pool over the neighbours. The construction is permutation-equivariant, matching the absence of any physical meaning in the ordering of a neighbour list. Details are given in Appendix B.

4 Experiments

We compare SpeND with RBF-FD (Tolstykh, 2000; Flyer et al., 2016c), using polyharmonic splines with polynomial augmentation, and LABFM (King et al., 2020), using the Wendland C2 kernel with Hermite polynomials. Both are mature schemes that have been applied to complex flow problems (Shankar et al., 2014; Bayona et al., 2017; King & Lind, 2022; King, 2024), and both take the same inputs as SpeND, an unstructured mesh-free stencil, so no mesh or structured grid is assumed anywhere in the comparison. Learned solvers are frequently compared against baselines run at a different accuracy (McGreivy & Hakim, 2024). We therefore sweep the resolution of each method and report wall-clock time against error across the range, so the methods are compared at equal accuracy rather than at a single configuration, and hold the order of approximation and stencil size matched throughout, so that differences reflect only how each method spends the degrees of freedom left after consistency is imposed.

The experiments test whether a single trained model transfers, without retraining, across the settings a discretisation must handle: collocation point distributions, orders of approximation, and PDEs.

All test cases are two-dimensional and admit analytical solutions, so measured error is attributable to the spatial discretisation rather than to a numerical reference carrying error of its own. Every stencil is limited to 30 neighbours, unless otherwise stated; the effect of stencil size is examined in Appendix H.

4.1 Toy Problem

We begin with a pure approximation test, which isolates the accuracy of the discrete operators from any solver. We evaluate six operators, pairing ∂_x and ∇^2 with consistency order $p \in \{2, 3, 4\}$. The $p = 2$ and $p = 4$ networks are trained. Since consistency is imposed by the projection rather than learned, the approximation order of a trained network can also be changed at inference, and to demonstrate this the $p = 3$ operators reuse the $p = 2$ network with only the moment constraints changed. The target function on the unit square $(x, y) \in [0, 1]^2$ is a unit square wave in x, truncated to its first four non-zero harmonics,

$$\phi(x, y) = \frac{4}{\pi} \sin(2\pi y) \sum_{n=1}^{4} \frac{\sin\big(2\pi(2n-1)(x - \frac{1}{4})\big)}{2n - 1}. \tag{10}$$

The function contains several discrete wavenumbers rather than a single one. The nodes are distributed in the domain with the particle shifting iterative procedure (Flyer et al., 2016a), which is standard in mesh-free simulations.

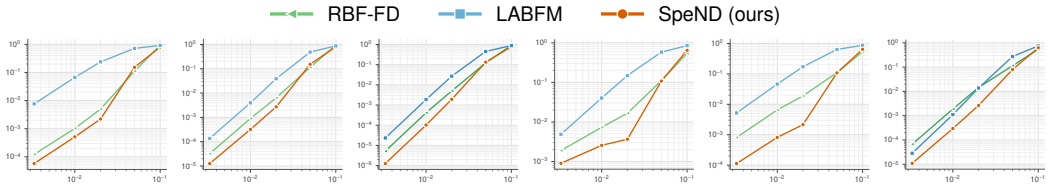

Figure 1: Convergence of the error in $\partial_x \phi$ (left three panels) and $\nabla^2 \phi$ (right three panels) for the test function (equation 10), at approximation orders $p = 2, 3, 4$ from left to right within each group. The vertical axis is the relative L_2 error, the horizontal axis the average node spacing. SpeND is compared against the two baselines at matched order and stencil size.

Figure 1 shows the error against mean spacing s, with the node count scaling as s^{-2} in two dimensions. At the coarsest resolution, $s = 10^{-1}$, all three methods coincide for every operator and order. The highest harmonic, $k_x = 14\pi$, sits at $k_x s/\pi = 1.4$ of the Nyquist wavenumber π/s, so no discretisation recovers it, hence the errors are identical. As s decreases the harmonics enter the resolvable band, SpeND separates from both baselines and attains the lowest error at every order.

For ∂_x, SpeND reaches a given error with roughly $2\times$ fewer nodes than RBF-FD at every order at fine resolutions. Against LABFM the saving depends strongly on order, falling from roughly $45\times$ at $p = 2$ to $4\times$ at $p = 4$. For ∇^2 the improvement is stronger still, particularly at $p = 2$ and $p = 3$, where, in the best case, SpeND requires up to $15\times$ fewer nodes than RBF-FD to reach the same error.

4.2 Viscous Burgers' Equation

We next solve the viscous Burgers equation on the doubly periodic unit square at Reynolds number 100, with $\mathbf{u} = (u, v)$ and $\mathbf{u}(x, y, 0) = (\sin 2\pi x, 0)$. The solution remains one-dimensional, the sinusoid steepening into a viscous shock at $x = 0.5$ before decaying, but is discretised with two-dimensional operators. A reference solution is obtained to high precision through the Cole–Hopf transformation. Nodes are placed uniformly and each coordinate is perturbed by $\varepsilon s \mathcal{U}(-\frac{1}{2}, \frac{1}{2})$ with $\varepsilon = 0.8$. This level of disorder is well beyond what particle simulations typically encounter (Suchde et al., 2023), since particle shifting (Xu et al., 2009; Lind et al., 2012) keeps the local spacing uniform while the distribution remains unstructured, and serves here as a stress test. We use the $p = 2$ gradient and Laplacian operators from Section 4.1 without retraining or modification, with ∂_y obtained by applying the ∂_x network to the stencil with swapped coordinates. The right-hand

side is assembled from the discrete operators of each method and integrated with classical 4th order Runge–Kutta to $t = 1$, and we report the L_2 error in u against the reference at $t = 1$.

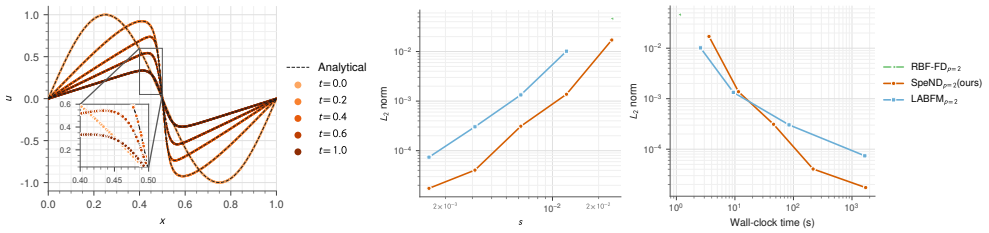

Figure 2: Viscous Burgers' equation at $Re = 100$. Left, the SpeND solution at successive times against the reference; middle, the L_2 error in u at $t = 1$ against average node spacing; right, the same error against total wall-clock time.

Figure 2 shows the results. All three methods share the same solver, they differ only in their discrete operators. RBF-FD completes the simulation only at the coarsest resolution and diverges at every finer one. We hypothesise this is due to its dispersion errors (Appendix C.2): the real part of its first-derivative response exceeds the exact response above $0.2k_{\text{Ny}}$, and its Laplacian carries a large imaginary part above $0.4k_{\text{Ny}}$. Dispersion errors of this kind generate spurious oscillations at steep gradients, and as the shock steepens the nonlinear term transfers energy into these wavenumbers, where the oscillations can grow until the solution becomes unstable. LABFM diverges at the coarsest resolution but converges at finer ones. Its modal response departs from the exact one at a higher fraction of k_{Ny} than SpeND's (Appendix C.2), so at coarse resolution the wavenumbers excited by the steepening shock fall where its errors are large; refining the nodes moves these wavenumbers to a smaller fraction of k_{Ny}, into the range the operator resolves. SpeND completes every run and converges throughout, with an error roughly $10\times$ below LABFM.

Figure 2 right panel shows the error against wall-clock time. Below roughly 3 s LABFM attains lower error at equivalent wall-clock time, since at these resolutions SpeND's cost is dominated by its more expensive weight computation. On a fixed node set this one-off cost scales linearly with the number of nodes, whereas time integration scales with the number of nodes times the number of steps. LABFM reaches its finest-resolution error of 7×10^{-5} in about 10^3 s, whereas SpeND reaches the same error at a coarser resolution in about 10^2 s, a $10\times$ speedup.

4.3 Poisson's Equation

We solve $\nabla^2\phi = f$ on the unit square with a circular obstacle of radius 0.2 at its centre, doubly periodic on the outer boundaries and with Dirichlet conditions on the obstacle, using the manufactured solution $\phi = \sin(6\pi x)\sin(6\pi y)$. Nodes are relaxed by particle shifting into the isotropic distribution typical of particle simulations, and stencils adjacent to the obstacle are truncated by the wall. We use the $p = 3$ Laplacian of Section 4.1 without retraining. Since ∇^2 has order $m = 2$, the expected convergence rate is $p + 1 - m = 2$.

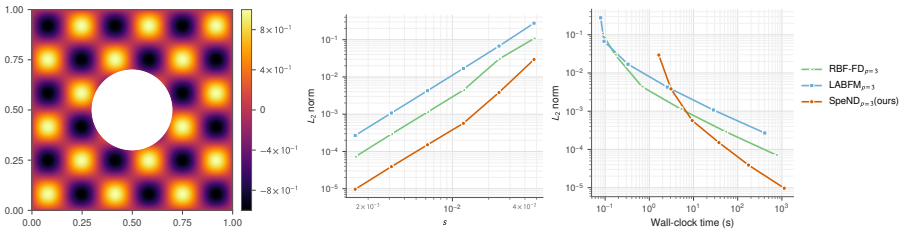

Figure 3: Poisson equation on the unit square with a central circular obstacle. Left, analytical solution; middle, the L_2 error in ϕ against average node spacing; right, the L_2 error against total wall-clock time.

Figure 3 shows the results. All methods converge at second order, and SpeND attains the lowest error at every resolution: at the finest, its error is approximately $28\times$ below LABFM and $7\times$ below RBF-FD. As in Section 4.2, SpeND's cost at coarse resolution is dominated by computing the weights, but this is amortised rapidly with refinement. RBF-FD reaches its lowest error, 7×10^{-5}, in about 740 s, which SpeND matches in about 90 s, a speedup of roughly $8\times$. LABFM reaches 3×10^{-4} in about 400 s, which SpeND matches in about 20 s, roughly $20\times$ faster.

4.4 Weakly Compressible Navier–Stokes

Lastly, we simulate the decaying two-dimensional Taylor–Green vortex with the weakly compressible Navier–Stokes equations at Mach number 0.1 and Reynolds number of 100 on the periodic domain $[-0.5, 0.5]^2$. Nodes are relaxed by particle shifting as in Section 4.3. All methods use $p = 4$ gradient and Laplacian operators, which for SpeND are those of Section 4.1 used without retraining. All runs share the same node sets, time integrator and hyperviscous filter, and differ only in the discrete operators.

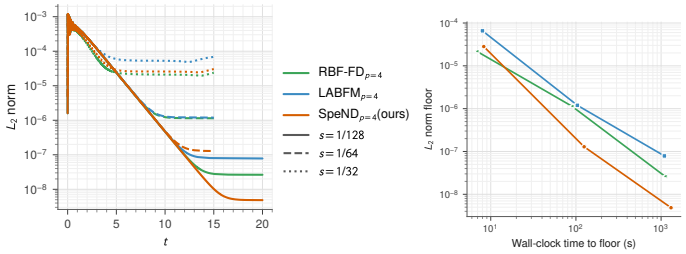

Figure 4: Taylor–Green vortex at Mach 0.1 and Reynolds number 100. Left: L_2 error in the velocity, normalised by the velocity magnitude at the first output time, against time at node spacings $s = 1/32$ and $1/64$ (to $t = 15$) and $s = 1/128$ (to $t = 20$). Right: error floor of each run against the wall-clock time taken to reach it, for each method and resolution.

At the coarsest resolution ($s = 1/32$) the three schemes plateau at the same order of magnitude, RBF-FD attaining 2.0×10^{-5} and SpeND 2.5×10^{-5}. The separation appears with refinement. At $s = 1/64$ SpeND is roughly an order of magnitude below both baselines, and at $s = 1/128$ RBF-FD and LABFM plateau at 2.8×10^{-8} and 8×10^{-8}, while SpeND reaches approximately 5×10^{-9}.

The advantage persists once cost is accounted for. Plotting the error floor against wall-clock time, all three methods sit close together at about 20 s. As resolution increases the curves separate. RBF-FD reaches its lowest error floor, 2.8×10^{-8}, in about 10^3 s, which SpeND matches in about 3.5×10^2 s, a speedup of roughly $3\times$. LABFM reaches 8×10^{-8} in about 10^3 s, which SpeND matches in about 1.8×10^2 s, roughly $5.5\times$ faster.

5 Conclusion

We have introduced a framework for learning differential operators on unstructured point clouds, in which a network is trained per operator to optimise the modal response while polynomial consistency is enforced by construction. Formal order of accuracy is therefore preserved, and the training objective carries a direct numerical interpretation rather than acting as a generic regression loss. The same trained operators are applied without retraining to an analytic test function, to Burgers' equation, to a Poisson problem with an embedded obstacle and to the Taylor–Green vortex, reaching matched accuracy between $3\times$ and $20\times$ faster than the classical baselines at the finer resolutions.

In the experiments, at coarse resolutions, the classical baselines remain competitive; the gains appear once the active wavenumbers fall inside the band on which the modal response is optimised. Because the learned operators enter a solver only through their stencil weights, they can replace the operators of an existing mesh-free code without altering the rest of the pipeline, and without giving up the formal order of accuracy that code relies on. More broadly, we see this as a step towards learned components for physical simulation that retain the structural properties classical numerical methods are built on.

AI USE STATEMENT

In this work, we used generative AI tools to help implement methods, primarily providing feedback on the experimental methodology, in particular on how to construct fair comparisons between methods; to help refine the paper's claims in light of the obtained results; and to check and correct the logic of the authors' mathematical derivations. We have not used generative AI tools to develop the conceptual framework, formulate mathematical claims, propose hypotheses, generate synthetic data, or interpret results, and the remaining required disclosure tasks (providing critical ingredients for or writing proofs, language translation, dataset cleaning, qualitative data analysis) are not applicable to this work.

Additionally, we used generative AI tools to debug, edit, reorganise, and comment on existing software code, to identify relevant literature when studying specific ideas, to search for related work during the literature review, and to edit the paper for readability. We have reviewed all AI-assisted work: all AI-generated code was verified and tested for correctness by the authors, all mathematical content was derived by the authors and any AI-suggested corrections were independently verified, all claims were checked by the authors against the experimental evidence, all AI-suggested references were read and verified against the original sources, and all edited text was reviewed by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work develops learned discretisation operators for mesh-free numerical methods and does not involve human subjects, personal data, or sensitive information. All training and evaluation data are synthetic point distributions and analytical test functions generated by the authors, so no dataset licensing, privacy, or consent issues arise. The methods target general-purpose scientific computing, and we do not foresee direct harmful applications beyond those of numerical simulation tools in general.

REPRODUCIBILITY STATEMENT

All data in this work were generated by the authors. The training corpus consists only of synthetic stencil geometries, and every test case has an analytical or semi-analytical reference solution, so no external datasets are required. Upon publication, we will release as open source the data generation pipeline, the training code and the trained SpeND operators, together with the Fortran solver used for all experiments, its TorchFort interface to the trained networks, and our implementations of the LABFM and RBF-FD baselines with the configurations used in this paper. Within the paper, the problem setup, moment projection and training objective are described in Sections 3.1–3.3. Data generation is detailed in Appendix A, the architecture and training hyperparameters in Appendix B, the modal-response objective for other operators in Appendix C, the baselines in Appendix E, the test-case setups in Appendix F, and hardware, software and training cost in Appendix I.

REFERENCES

A. Anandkumar, K. Azizzadenesheli, K. Bhattacharya, N. Kovachki, Z. Li, B. Liu, and A. Stuart. Neural operator: Graph kernel network for partial differential equations. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2019. URL https://openreview.net/forum?id=fg2ZFmXFO3.

Y. Bar-Sinai, S. Hoyer, J. Hickey, and M. P. Brenner. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31):15344–15349, 2019. doi: 10.1073/pnas.1814058116.

V. Bayona and M. Kindelan. Propagation of premixed laminar flames in 3D narrow open ducts using RBF-generated finite differences. *Combustion Theory and Modelling*, 17(5):789–803, 2013. doi: 10.1080/13647830.2013.801519.

V. Bayona, N. Flyer, G. M. Lucas, and A. J. G. Baumgaertner. A 3-d RBF-FD solver for modeling the atmospheric global electric circuit with topography (GEC-RBFFD v1.0). *Geoscientific Model Development*, 8(10):3007–3020, 2015. doi: 10.5194/gmd-8-3007-2015.

Victor Bayona, Natasha Flyer, Bengt Fornberg, and Gregory A. Barnett. On the role of polynomials in RBF-FD approximations: II. numerical solution of elliptic PDEs. *Journal of Computational Physics*, 332:257–273, 2017. doi: 10.1016/j.jcp.2016.12.008.

J.J. Benito, F. Ureña, and L. Gavete. Influence of several factors in the generalized finite difference method. *Applied Mathematical Modelling*, 25(12):1039–1053, 2001. ISSN 0307-904X. doi: https://doi.org/10.1016/S0307-904X(01)00029-4. URL https://www.sciencedirect.com/science/article/pii/S0307904X01000294.

John P. Boyd. *Chebyshev and Fourier Spectral Methods*. Dover Publications, Mineola, NY, 2nd edition, 2001.

J. Brandstetter, D. Worrall, and M. Welling. Message passing neural pde solvers, 2023. URL https://arxiv.org/abs/2202.03376.

H.M. Broadley, J.R.C. King, and S.J. Lind. Compact labfm: A framework for meshless methods with spectral-like resolving power. *Journal of Computational Physics*, 567:115308, 2026a. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2026.115308. URL https://www.sciencedirect.com/science/article/pii/S0021999126006571.

H.M. Broadley, J.R.C. King, and S.J. Lind. Improving the accuracy of meshless methods via resolving power optimisation using multiple kernels, 2026b. ISSN 0021-9991. URL https://www.sciencedirect.com/science/article/pii/S0021999126005899.

Y. Choi, S. W. Cheung, Y. Kim, P.-H. Tsai, A. N. Diaz, I. Zanardi, S. W. Chung, D. M. Copeland, C. Kendrick, W. Anderson, T. Iliescu, and M. Heinkenschloss. Defining foundation models for computational science: A call for clarity and rigor, 2025. URL https://arxiv.org/abs/2505.22904.

R. W. Clough. Original formulation of the finite element method. *Finite Elements in Analysis and Design*, 7(2):89–101, 1990. doi: 10.1016/0168-874X(90)90001-U.

R. Courant, K. Friedrichs, and H. Lewy. On the partial difference equations of mathematical physics. *IBM Journal of Research and Development*, 11(2):215–234, 1967. doi: 10.1147/rd.112.0215.

G. de Romémont, F. Renac, J. Nunez, D. Gueyffier, and F. Chinesta. A data-driven learned discretization approach in finite volume schemes for hyperbolic conservation laws and varying boundary conditions. *Computers & Fluids*, 307:106978, 2026. ISSN 0045-7930. doi: 10.1016/j.compfluid.2026.106978. URL https://www.sciencedirect.com/science/article/pii/S0045793026000204.

W. Dehnen and H. Aly. Improving convergence in smoothed particle hydrodynamics simulations without pairing instability. *Monthly Notices of the Royal Astronomical Society*, 425(2):1068–1082, 09 2012. ISSN 0035-8711. doi: 10.1111/j.1365-2966.2012.21439.x. URL https://doi.org/10.1111/j.1365-2966.2012.21439.x.

G. Dresdner, D. Kochkov, P. Norgaard, L. Zepeda-Núñez, J. A. Smith, M. P. Brenner, and S. Hoyer. Learning to correct spectral methods for simulating turbulent flows, 2023. URL https://arxiv.org/abs/2207.00556.

N. Flyer, G. A. Barnett, and L. J. Wicker. Enhancing finite differences with radial basis functions: Experiments on the navier–stokes equations. *Journal of Computational Physics*, 316:39–62, 2016a. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2016.02.078. URL https://www.sciencedirect.com/science/article/pii/S0021999116300195.

N. Flyer, B. Fornberg, V. Bayona, and G. A. Barnett. On the role of polynomials in RBF-FD approximations: I. interpolation and accuracy. *Journal of Computational Physics*, 321:21–38, 2016b. doi: 10.1016/j.jcp.2016.05.026.

N. Flyer, G. B. Wright, and B. Fornberg. *Radial Basis Function-Generated Finite Differences: A Mesh-Free Method for Computational Geosciences*, pp. 1–30. Springer Berlin Heidelberg, 2020. doi: 10.1007/978-3-642-27793-1_61-1.

Natasha Flyer, Bengt Fornberg, Victor Bayona, and Gregory A. Barnett. On the role of polynomials in RBF-FD approximations: I. interpolation and accuracy. *Journal of Computational Physics*, 321:21–38, 2016c. doi: 10.1016/j.jcp.2016.05.026.

B. Fornberg and E. Lehto. Stabilization of RBF-generated finite difference methods for convective PDEs. *Journal of Computational Physics*, 230(6):2270–2285, 2011. doi: 10.1016/j.jcp.2010.12.014.

David Gottlieb and Steven A. Orszag. *Numerical Analysis of Spectral Methods: Theory and Applications*, volume 26 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1977. doi: 10.1137/1.9781611970425.

D. Greenfeld, M. Galun, R. Kimmel, I. Yavneh, and R. Basri. Learning to optimize multigrid pde solvers, 2019. URL https://arxiv.org/abs/1902.10248.

J.T. Hsieh, S. Zhao, S. Eismann, L. Mirabella, and S. Ermon. Learning neural pde solvers with convergence guarantees, 2019. URL https://arxiv.org/abs/1906.01200.

A. Jameson, W. Schmidt, and E. Turkel. Numerical solution of the Euler equations by finite volume methods using Runge–Kutta time stepping schemes. In *Proceedings of the 14th Fluid and Plasma Dynamics Conference*. AIAA, 1981. doi: 10.2514/6.1981-1259.

P. S. Jensen. Finite difference techniques for variable grids. *Computers & Structures*, 2(1):17–29, 1972. ISSN 0045-7949. doi: https://doi.org/10.1016/0045-7949(72)90020-X. URL https://www.sciencedirect.com/science/article/pii/004579497290020X.

B. M. Johnston, P. R. Johnston, S. Corney, and D. Kilpatrick. Non-Newtonian blood flow in human right coronary arteries: transient simulations. *Journal of Biomechanics*, 39(6):1116–1128, 2006. doi: 10.1016/j.jbiomech.2005.01.034.

K. Jordan, Y. Jin, V. Boza, J. You, F. Cesista, L. Newhouse, and J. Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. *URL https://kellerjordan. github. io/posts/muon*, 6(3):4, 2024.

G. E. Karniadakis and S. J. Sherwin. *Spectral/hp Element Methods for Computational Fluid Dynamics*. Oxford University Press, Oxford, 2nd edition, 2005. ISBN 9780198528692.

C. A. Kennedy, M. H. Carpenter, and R. M. Lewis. Low-storage, explicit Runge–Kutta schemes for the compressible Navier–Stokes equations. *Applied Numerical Mathematics*, 35(3):177–219, 2000. doi: 10.1016/S0168-9274(99)00141-5.

J. W. Kim and D. J. Lee. Optimized compact finite difference schemes with maximum resolution. *AIAA Journal*, 34(5):887–893, 1996. ISSN 0001-1452. URL https://eprints.soton.ac.uk/id/eprint/23038/.

J. R. C. King and S. J. Lind. High-order simulations of isothermal flows using the local anisotropic basis function method (LABFM). *Journal of Computational Physics*, 449:110760, 2022. doi: 10.1016/j.jcp.2021.110760.

J. R. C. King, S. J. Lind, and A. M. A. Nasar. High order difference schemes using the local anisotropic basis function method. *Journal of Computational Physics*, 415:109549, 2020. doi: 10.1016/j.jcp.2020.109549.

J.R.C. King. A mesh-free framework for high-order direct numerical simulations of combustion in complex geometries. *Computer Methods in Applied Mechanics and Engineering*, 421:116762, 2024. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2024.116762. URL https://www.sciencedirect.com/science/article/pii/S0045782524000185.

D. Kochkov, J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner, and S. Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118 (21):e2101784118, 2021. doi: 10.1073/pnas.2101784118.

P. D. Lax and R. D. Richtmyer. Survey of the stability of linear finite difference equations. *Communications on Pure and Applied Mathematics*, 9(2):267–293, 1956. doi: https://doi.org/10.1002/cpa.3160090206. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cpa.3160090206.

Sabine Le Borne and Willi Leinen. Guidelines for rbf-fd discretization: Numerical experiments on the interplay of a multitude of parameter choices. *Journal of Scientific Computing*, 95(1):8, 2023. ISSN 1573-7691. doi: 10.1007/s10915-023-02123-7. URL https://doi.org/10.1007/s10915-023-02123-7.

S. K. Lele. Compact finite difference schemes with spectral-like resolution. *Journal of Computational Physics*, 103(1):16–42, 1992. doi: 10.1016/0021-9991(92)90324-R.

Z. Li, N. B. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. In *Proceedings of the International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=c8P9NQVtmnO.

S. J. Lind, R. Xu, P. K. Stansby, and B. D. Rogers. Incompressible smoothed particle hydrodynamics for free-surface flows: A generalised diffusion-based algorithm for stability and validations for impulsive flows and propagating waves. *Journal of Computational Physics*, 231(4):1499–1523, 2012. doi: 10.1016/j.jcp.2011.10.027.

L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3): 218–229, 2021. doi: 10.1038/s42256-021-00302-5.

L. B. Lucy. A numerical approach to the testing of the fission hypothesis. *The Astronomical Journal*, 82:1013–1024, 1977. doi: 10.1086/112164.

J. Martínez, F. Piscaglia, A. Montorfano, A. Onorati, and S.M. Aithal. Influence of momentum interpolation methods on the accuracy and convergence of pressure–velocity coupling algorithms in openfoam®. *Journal of Computational and Applied Mathematics*, 309:654–673, 2017. ISSN 0377-0427. doi: https://doi.org/10.1016/j.cam.2016.03.037. URL https://www.sciencedirect.com/science/article/pii/S0377042716301649.

N. McGreivy and A. Hakim. Weak baselines and reporting biases lead to overoptimism in machine learning for fluid-related partial differential equations. *Nature Machine Intelligence*, 6(10):1256–1269, oct 2024. ISSN 2522-5839. doi: 10.1038/s42256-024-00897-5. URL https://doi.org/10.1038/s42256-024-00897-5.

S. A. Orszag. Numerical simulation of incompressible flows within simple boundaries. I. Galerkin (spectral) representations. *Studies in Applied Mathematics*, 50(4):293–327, 1971. doi: 10.1002/sapm1971504293.

C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017. URL https://arxiv.org/abs/1612.00593.

N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. A. Hamprecht, Y. Bengio, and A. Courville. On the spectral bias of neural networks, 2019. URL https://arxiv.org/abs/1806.08734.

M. Raissi, P. Perdikaris, and G. E. 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. doi: 10.1016/j.jcp.2018.10.045.

V. G. Satorras, Z. Akata, and M. Welling. Combining generative and discriminative models for hybrid inference, 2019. URL https://arxiv.org/abs/1906.02547.

V. Shankar, G. B. Wright, A. L. Fogelson, and R. M. Kirby. A radial basis function (rbf) finite difference method for the simulation of reaction–diffusion equations on stationary platelets within the augmented forcing method. *International Journal for Numerical Methods in Fluids*, 75(1):1–22, 2014. doi: https://doi.org/10.1002/fld.3880. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.3880.

L. G. Starepravo, G. Fourtakas, S. Lind, A. B. Harish, T. Tang, and J. R. C. King. Learning mesh-free discrete differential operators with self-supervised graph neural networks, 2026. URL https://arxiv.org/abs/2603.24641.

T. F. Stocker, D. Qin, G.-K. Plattner, M. Tignor, S. K. Allen, J. Boschung, A. Nauels, Y. Xia, V. Bex, P. M. Midgley, et al. *Climate Change 2013: The Physical Science Basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*. Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA, 2013.

P. Suchde, T. Jacquemin, and O. Davydov. Point cloud generation for meshfree methods: An overview. *Archives of Computational Methods in Engineering*, 30(2):889–915, 2023. doi: 10.1007/s11831-022-09820-w.

Christopher K.W. Tam and Jay C. Webb. Dispersion-relation-preserving finite difference schemes for computational acoustics. *Journal of Computational Physics*, 107(2):262–281, 1993. ISSN 0021-9991. doi: https://doi.org/10.1006/jcph.1993.1142. URL https://www.sciencedirect.com/science/article/pii/S0021999183711423.

A. I. Tolstykh. On using rbf-based differencing formulas for unstructured and mixed structured-unstructured grid calculations. In *Proceedings of the 16th IMACS world congress*, volume 228, pp. 4606–4624, 2000.

N. Trask, M. Maxey, and X. Hu. Compact moving least squares: An optimization framework for generating high-order compact meshless discretizations. *Journal of Computational Physics*, 326:596–611, 2016. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2016.08.045. URL https://www.sciencedirect.com/science/article/pii/S0021999116304028.

N. Trask, M. Perego, and P. Bochev. A high-order staggered meshless method for elliptic problems. *SIAM Journal on Scientific Computing*, 39(2):A479–A502, 2017. doi: 10.1137/16M1055992. URL https://doi.org/10.1137/16M1055992.

K. Um, R. Brand, Y. Fei, P. Holl, and N. Thuerey. Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers, 2021. URL https://arxiv.org/abs/2007.00016.

S. Wang, X. Yu, and P. Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2021.110768. URL https://www.sciencedirect.com/science/article/pii/S002199912100663X.

R. Xu, P. Stansby, and D. Laurence. Accuracy and stability in incompressible SPH (ISPH) based on the projection method and a new approach. *Journal of Computational Physics*, 228(18):6703–6725, 2009. doi: 10.1016/j.jcp.2009.05.032.

J. Zhuang, D. Kochkov, Y. Bar-Sinai, M. P. Brenner, and S. Hoyer. Learned discretizations for passive scalar advection in a two-dimensional turbulent flow. *Physical Review Fluids*, 6:064605, 2021. doi: 10.1103/PhysRevFluids.6.064605.

APPENDIX

Nomenclature **16**

A Data generation **18**

B Architecture & Training Details **19**

C Modal Response & Analysis **20**

C.1 Modal Response of the Laplacian 20

C.2 Modal Response of the Trained Operators 20

C.3 Dispersion and Dissipation in Advection and Diffusion 21

D Stability Analysis **22**

E Traditional Mesh-Free Discretisations **22**

E.1 LABFM 23

E.2 RBF-FD 24

F Experimental Details & Additional Results **24**

F.1 Viscous Burgers Equation 24

F.2 Poisson's Equation 25

F.3 Weakly Compressible Navier–Stokes Equations 26

G Limitations **27**

H Ablations **28**

I Hardware & Computational Cost **31**

NOMENCLATURE

Domain and discretisation	
$\Omega \subset \mathbb{R}^d$	Spatial domain of dimension d
$\mathcal{P}$	Point cloud (collocation nodes) in Ω
$\mathbf{x}_i$	Position of node i
$(\cdot)_{ji}$	Difference $(\cdot)_j - (\cdot)_i$
$\mathcal{N}_i$	Index set of the stencil of node i, node i included
N	Stencil size, $N = \lvert\mathcal{N}_i\rvert$
R_i	Distance from $\mathbf{x}_i$ to the farthest node of $\mathcal{N}_i$
s	(Mean) node spacing
ϕ	Generic scalar field
Operators and consistency	
D	Continuous differential operator, e.g. ∂_x, ∂_y, ∇^2
m	Order of D
L^D	Discrete approximation of D
$w^D_{j,i}$, $\mathbf{w}^D_i$	Physical stencil weight of node j at node i; weight vector of node i
p	Consistency order (polynomial degree reproduced exactly)
$\boldsymbol{\alpha}$	Multi-index, $1 \le \lvert\boldsymbol{\alpha}\rvert \le p$
N_p	Number of moment conditions, $\binom{p+d}{d} - 1$
$\mathbf{V}_i$	Moment matrix of node i
$\mathbf{d}^D$	Stencil moment vector of D
SpeND	
$\bar{(\cdot)}$	Quantity in normalised stencil coordinates
$\bar{\mathbf{x}}_{ji}$	Normalised offset, $\mathbf{x}_{ji}/R_i$
$\bar{\mathbf{V}}_i$	Moment matrix assembled from $\bar{\mathbf{x}}_{ji}$
f_θ, θ	Weight-predicting network and its parameters
$\tilde{\mathbf{w}}^D_i$	Candidate (unprojected) weights
Π_i	Moment projection onto the consistent affine subspace
$\bar{\mathbf{w}}^D_i$	Projected weights, $\mathbf{w}^D_i = R_i^{-m}\bar{\mathbf{w}}^D_i$
$\mathbf{P}_i$	Orthogonal projector onto $\ker \bar{\mathbf{V}}_i$
$(\cdot)^+$	Moore–Penrose pseudoinverse
σ_m	Output scaling factor, $\bar{s}_N^{-m}$
Modal response	
$\mathbf{k} = (k_x, k_y)$	Wavevector
$\phi_{\mathbf{k}}$	Fourier mode $e^{\mathrm{i}\mathbf{k}\cdot\mathbf{x}}$
$\hat{L}^D_i(\mathbf{k})$	Modal response of L^D at node i
k_{eff}	Effective wavenumber of ∂_x, $-\mathrm{i}\hat{L}_i^{\partial_x}$
q^2_{eff}	Effective squared wavenumber of ∇^2, $-\hat{L}_i^{\nabla^2}$
$\bar{s}_N$	Mean normalised spacing, $\sqrt{\pi/N}$
k_{Ny}	Nyquist wavenumber, $\pi/\bar{s}_N$
$\hat{k}$	Normalised wavenumber, $\lVert\mathbf{k}\rVert/k_{\text{Ny}}$
η	Band limit, $\lVert\mathbf{k}\rVert \le \eta\, k_{\text{Ny}}$
Training	
Φ	Class of test functions
$r_i[\phi]$	Stencil residual
$\mathcal{K}$	Training wavenumbers on the resolved half-disk
$\sigma(\mathbf{k})$	Relative-error weight
ϵ	Floor of $\sigma(\mathbf{k})$
$\omega_>$, $\lambda_>$	Asymmetric weight and penalty for over-predicted real part
ω_+, λ_+	Asymmetric weight and penalty for positive imaginary part
$\mathcal{L}_i^{\text{disp}}$, $\mathcal{L}_i^{\text{diss}}$	Per-stencil dispersion and dissipation losses
γ	Weight of $\mathcal{L}_i^{\text{diss}}$
$\mathcal{S}$	Corpus of training stencils
λ_{b}	Adaptive boundary-stencil weight
B, C	Number of network blocks; feature width

Data generation	
ε	Lattice disorder amplitude
n_{it}	Number of particle-shifting iterations
h	Kernel smoothing length (particle shifting, LABFM)
$\mathbf{n}$, ϑ	Wall normal and its angle
β	Wall distance from the central node, in units of R_i
Model problems and analysis	
t	Time
$\mathbf{u} = (u, v)$	Velocity
ρ	Density
f	Poisson source term
Re, Ma	Reynolds and Mach numbers
a, $\tilde{a}$	Advection speed; numerical phase speed
ν	Diffusivity
$\hat{u}(t)$	Amplitude of a single Fourier mode
$\mathbf{G}^D$, μ	Global operator matrix and its eigenvalues
$\boldsymbol{\phi}$	Vector of nodal values
Baselines	
$\mathbf{W}_{ji}$, $\boldsymbol{\Psi}_i^D$	LABFM anisotropic basis functions and coefficients
$\mathbf{A}_i$, $\mathbf{X}_{ji}$	LABFM system matrix; vector of Taylor monomials
κ, H_n	Wendland C^2 kernel; Hermite polynomial of degree n
χ	Polyharmonic spline r^5
$\mathbf{K}_i$, $\mathbf{M}_i$	RBF-FD kernel and polynomial matrices
$\boldsymbol{\xi}_i$, $\mathbf{c}_i^D$	RBF-FD Lagrange multipliers; kernel right-hand side

A DATA GENERATION

The dataset contains stencil geometries only, with no function values or reference solutions. Since all experiments are two-dimensional, each sample is a two-dimensional stencil of N nodes, the central node included. The generation procedure below extends directly to three dimensions, the only change being the dimension of the node coordinates. Rather than being extracted from a single discretised domain, each stencil is generated from its own independent point cloud. Stencils are stored as the normalised offsets $\bar{\mathbf{x}}_{ji} = \mathbf{x}_{ji}/R_i$, which places every stencil within the unit circle.

Node distributions. Stencils are generated from two families of node sets. The first is a Cartesian lattice in which every node is displaced independently along each axis by $\varepsilon s\,\mathcal{U}(-\frac{1}{2}, \frac{1}{2})$, with the disorder ε drawn uniformly per stencil; this configuration is commonly referred to as random node placement (Suchde et al., 2023). The second imitates the node sets on which meshfree simulations are run in practice, and is obtained by iterative particle shifting (Xu et al., 2009; Lind et al., 2012; Flyer et al., 2016a). Starting from a lattice with the high disorder $\varepsilon = 1$, each of n_{it} iterations pushes every node away from its neighbours within a radius $2h$, driving the cloud towards a near-isotropic packing. We use $h = 2s$ and cap each displacement at $0.2s$, so that $n_{\mathrm{it}} = 0$ yields the disordered lattice and $n_{\mathrm{it}} = 30$ gives a relaxed packing. Each family is divided into four bins of equal size, $\varepsilon \in [0.2, 0.4], [0.4, 0.6], [0.6, 0.8], [0.8, 1.0]$ and $n_{\mathrm{it}} \in \{0, \dots, 7\}, \{8, \dots, 15\}, \{16, \dots, 23\}, \{24, \dots, 30\}$, with the parameter drawn uniformly within each bin. A representative stencil from each bin is shown in Figure 5.

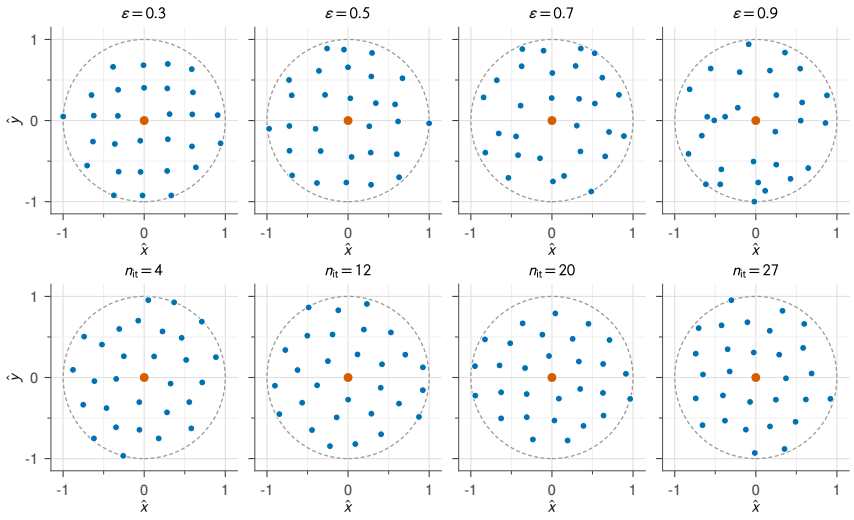

Figure 5: Representative stencils from each of the eight bins. Top row, perturbed Cartesian lattices with increasing disorder ε. Bottom row, particle-shifted node sets with increasing number of shifting iterations n_{it}. The value of ε or n_{it} is given above each panel.

Boundary stencils. Every interior stencil is paired with one copy that imitates a stencil adjacent to a wall. We draw a line with unit normal $\mathbf{n} = (\cos\vartheta, \sin\vartheta)$, $\vartheta \sim \mathcal{U}(0, 2\pi)$, at a distance $\beta \sim \mathcal{U}(0.05, 0.7)$ from the central node in units of R_i, and discard every node with $\bar{\mathbf{x}}_{ji}\cdot\mathbf{n} > \beta$. The stencil is then refilled to N nodes with the nearest remaining nodes on the domain side and renormalised by its new R_i. Since $\beta > 0$ the central node always remains in the domain, and every boundary stencil keeps exactly N nodes. Representative boundary stencils from both families are shown in Figure 6.

Dataset size and split. Each of the eight bins contributes 5×10^5 interior stencils and their 5×10^5 boundary copies, giving 8×10^6 stencils in total, half of them boundary stencils. These are split at random into training, validation and test sets in the proportions $70/20/10$.

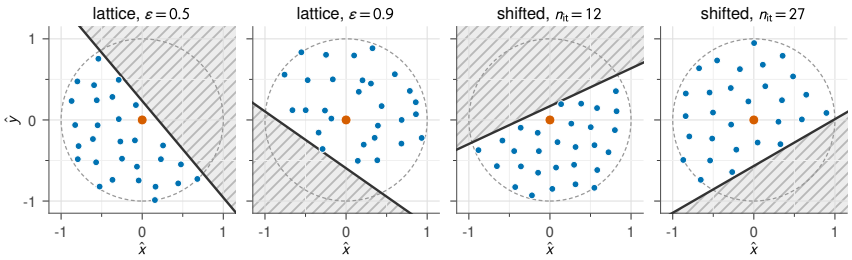

Figure 6: Representative boundary stencils. The two left panels are drawn from perturbed Cartesian lattices and the two right panels from particle-shifted node sets. The bold line marks the wall.

B Architecture & Training Details

Architecture. Each node $j \in \mathcal{N}_i$ enters the network only through its normalised offset $\bar{\mathbf{x}}_{ji}$. A shared two-layer encoder of widths 128 and $C = 256$ lifts every offset to $\mathbf{h}_j^{(0)} \in \mathbb{R}^C$. The features are then updated by $B = 4$ blocks with independent parameters,

$$\mathbf{z}_j = \tanh\left(W_1^{(\ell)}\mathbf{h}_j^{(\ell-1)} + \mathbf{b}_1^{(\ell)}\right), \qquad \mathbf{g} = \max_{j' \in \mathcal{N}_i} \mathbf{z}_{j'}, \qquad \mathbf{h}_j^{(\ell)} = \tanh\left(W_2^{(\ell)}[\mathbf{z}_j; \mathbf{g}] + \mathbf{b}_2^{(\ell)}\right), \quad (11)$$

where the maximum is taken elementwise. Pooling occurs in every block, so each node is conditioned on the whole stencil repeatedly while the output remains one value per node. A shared decoder of widths 128 and 1, with a linear final layer, returns the candidate weight of each node. All maps are shared across nodes and the pooling is symmetric. The weights are therefore permutation equivariant, and no parameter depends on N. The network uses $\tanh$ activations for a total of roughly 855 thousand parameters. A separate network with this architecture is trained for each discrete differential operator.

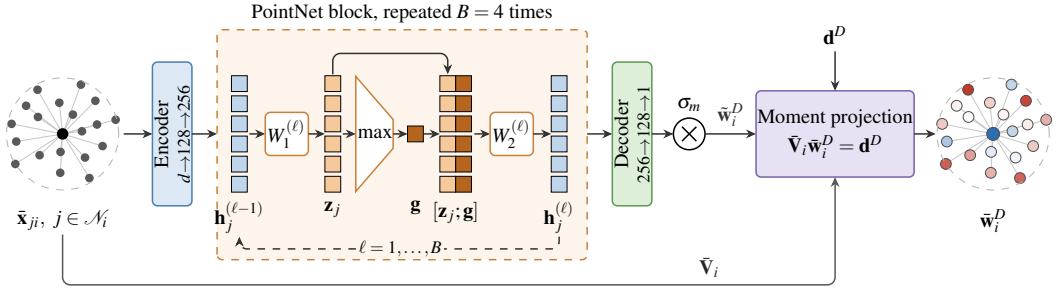

Figure 7: Architecture of SpeND. Normalised offsets $\bar{\mathbf{x}}_{ji}$ are encoded, refined by $B = 4$ PointNet blocks with elementwise max pooling over the stencil, and decoded into unprojected weights $\tilde{\mathbf{w}}_i^D$. The moment projection enforces $\bar{\mathbf{V}}_i \bar{\mathbf{w}}_i^D = \mathbf{d}^D$ exactly.

Output scaling. The weights of an operator of order m scale as the node spacing to the power $-m$. The decoder output is therefore multiplied by the fixed factor $\sigma_m = \bar{s}_N^{-m}$ before projection, where $\bar{s}_N = \sqrt{\pi/N}$ is the mean spacing of N nodes filling the unit disk. Without this factor, the decoder would need to produce weight norms of varying orders of magnitude. With σ_m, the weights are normalised independently of the order of the operator.

Training. A base network is trained on stencils of $N = 25$ for 40 epochs with Adam at learning rate 3×10^{-4}, and then 20 epochs with Muon (Jordan et al., 2024). Operators for each configuration, differing in stencil size, consistency order or boundary treatment, are obtained by fine-tuning this base network with Muon for 40 epochs, with a narrower training band and a smaller ϵ. Fine-tuning from a common base reduces training cost rather than improving the final operators. Muon applies only to weight matrices, so during these phases the biases are updated with Adam. The learning

rates are 2.7×10^{-3} for Muon and 9×10^{-5} for Adam, with a linear warm-up over 1000 optimiser steps. All phases use cosine decay to 10^{-7} and batches of 1024 stencils. The loss on boundary stencils is weighted by λ_{b}, set each epoch to the square root of the boundary-to-interior loss ratio. Remaining hyperparameters are listed in Table 2.

Table 2: Training hyperparameters of the modal-response objective for the base network and the fine-tuned operators. Operators with $p = 2$ and $p = 4$ are trained with these values. Operators with $p = 3$ are not trained separately, and instead reuse the $p = 2$ network with the $p = 3$ moment constraints imposed by the projection.

	Base		Fine-tuned	
	∂_x	∇^2	∂_x	∇^2
Band limit η	0.7	0.7	0.4	0.4
ϵ	0.25	0.1	0.1	0.1
Over-prediction penalty, real part $\lambda_>$	50	50	50	50
Over-prediction penalty, imaginary part λ_+	10	1	10	1
Imaginary-part weight γ	1	1	1	1
Boundary weight λ_{b}	adaptive, clipped to $[1, 50]$			

C Modal Response & Analysis

C.1 Modal Response of the Laplacian

In Section 3.3 we illustrate how the exact modal response is obtained for ∂_x. Here we repeat the procedure for the Laplacian, which illustrates how it extends to other operators.

As before, we place $\mathbf{x}_i$ at the origin and apply the continuous operator to a Fourier mode, which gives $\nabla^2 e^{\mathrm{i}\mathbf{k}\cdot\mathbf{x}}\big|_{\mathbf{x}=\mathbf{0}} = -\|\mathbf{k}\|^2$. The exact response is now real and negative, so we normalise the discrete response by -1 and define the effective squared wavenumber $q^2_{\text{eff}}(\mathbf{k}) := -\hat{L}_i^{\nabla^2}(\mathbf{k})$, which the exact operator reproduces as $\|\mathbf{k}\|^2$ (Lele, 1992). Separating real and imaginary parts,

$$q^2_{\text{eff}}(\mathbf{k}) = \sum_{j\in\mathcal{N}_i} \bar{w}_{j,i}^{\nabla^2} \left[1 - \cos(\mathbf{k}\cdot\bar{\mathbf{x}}_{ji})\right] \; - \; \mathrm{i} \sum_{j\in\mathcal{N}_i} \bar{w}_{j,i}^{\nabla^2} \sin(\mathbf{k}\cdot\bar{\mathbf{x}}_{ji}). \tag{12}$$

Compared with equation 6, the two parts exchange roles. The real part now carries the physical response and is compared against $\|\mathbf{k}\|^2$, while the imaginary part has no continuous counterpart and should vanish. The parity also reverses, since the real part is now even in $\mathbf{k}$ and the imaginary part odd, so the half-disk $\mathcal{K}$ again suffices.

The losses of Section 3.3 carry over with k_{eff} replaced by q^2_{eff} and k_x by $\|\mathbf{k}\|^2$ in equation 7, so that $\lambda_>$ now penalises a real part exceeding the exact response, which corresponds to excess diffusion, and $\sigma(\mathbf{k}) = \max(\|\mathbf{k}\|^2/k^2_{\mathrm{Ny}}, \epsilon)^{-2}$. Since the roles of the two parts are exchanged, equation 7 now controls dissipation and equation 8 dispersion. The imaginary part is penalised symmetrically, with $\lambda_+ = 1$. Unlike for ∂_x, where a leading response exceeds a nonzero physical speed, the exact drift here is zero, so both signs produce a spurious drift of equal speed in opposite directions and neither is the faster side (Appendix C.3).

Other operators follow the same procedure. Evaluating the continuous operator on $e^{\mathrm{i}\mathbf{k}\cdot\mathbf{x}}$ at the origin gives a real polynomial in $\mathbf{k}$ multiplied by i^m, where m is the order of the operator. Dividing the discrete response by i^m makes the exact response real, and the loss then compares the part that carries it while driving the other part to zero.

C.2 Modal Response of the Trained Operators

We report the modal response of the operators used in Section 4, comparing SpeND against the classical baselines for each operator and consistency order. The modal response measures how a discrete operator acts on a single Fourier mode, compared with the exact operator, with the real and imaginary parts governing dispersion and dissipation respectively (Section 3.3).

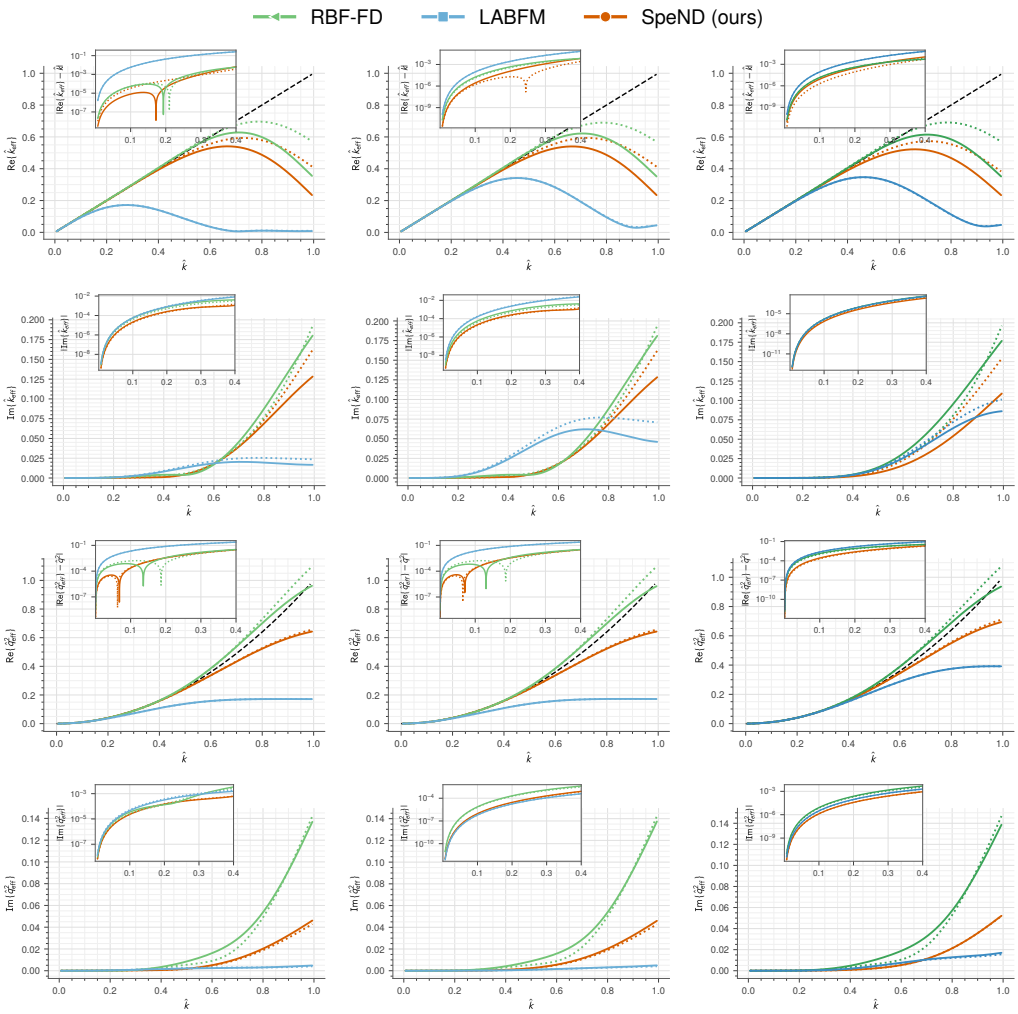

Figure 8: Modal response of the ∂_x (rows one and two) and ∇^2 (rows three and four) operators, showing the real and imaginary parts for consistency orders $p = 2, 3, 4$ (left to right). Wavenumbers are normalised by k_{Ny}. Solid lines correspond to modes aligned with the x-axis, $k_y = 0$, and dotted lines to diagonal modes, $k_x = k_y$. The dashed line is the exact response, and the insets show the error on a logarithmic scale. Stencils were generated with iterative particle shifting with $N = 30$.

Figure 8 shows the real and imaginary parts of the modal response of the ∂_x and ∇^2 operators for consistency orders $p = 2, 3, 4$ against wavenumbers normalised by the Nyquist wavenumber. The operators were trained with $\eta = 0.4$, so the objective constrains the modal response only for $\hat{k} \leq 0.4$. Within this band, SpeND matches or improves on both baselines in every panel, in both the real and imaginary parts. The largest gains occur at the lowest wavenumbers, where the logarithmic insets show errors up to several orders of magnitude below LABFM and consistently below RBF-FD. Beyond the training band the response is unconstrained, and the ranking between methods varies with operator and order.

C.3 Dispersion and Dissipation in Advection and Diffusion

We show how the two parts of the modal response enter the solution of the advection and diffusion equations. In both cases we discretise in space only, keeping time continuous, and substitute a single Fourier mode $u(\mathbf{x}, t) = \hat{u}(t)\, e^{\mathbf{i}\mathbf{k}\cdot\mathbf{x}}$.

Advection. For the one-dimensional equation $\partial_t u + a\,\partial_x u = 0$ with constant velocity $a > 0$, substituting the Fourier mode gives $\partial_x u = \mathrm{i}k_x u$, so $\hat{u}(t) = \hat{u}(0)\,e^{-\mathrm{i}ak_x t}$ and $u = \hat{u}(0)\,e^{\mathrm{i}k_x(x-at)}$, a wave travelling at speed a. Replacing ∂_x by its discrete counterpart, whose response we write as $\hat{L}_i^{\partial_x} = \mathrm{i}\,k_{\text{eff}}$ (so that $k_{\text{eff}} = k_x$ for an exact operator), gives $\mathrm{d}\hat{u}/\mathrm{d}t = -\mathrm{i}a\,k_{\text{eff}}\,\hat{u}$. Splitting k_{eff} into real and imaginary parts, its solution is

$$\hat{u}(t) = \hat{u}(0)\,\underbrace{e^{-\mathrm{i}a\,\mathrm{Re}\{k_{\text{eff}}\}\,t}}_{\text{phase}}\,\underbrace{e^{a\,\mathrm{Im}\{k_{\text{eff}}\}\,t}}_{\text{amplitude}}. \tag{13}$$

The first factor has a purely imaginary exponent and hence unit modulus, so it rotates $\hat{u}$ in the complex plane without changing its size; the second is real and positive, so it changes the size only. Thus $|\hat{u}(t)| = |\hat{u}(0)|\,e^{a\,\mathrm{Im}\{k_{\text{eff}}\}t}$, while multiplying by $e^{\mathrm{i}k_x x}$ turns the phase factor into $e^{\mathrm{i}k_x(x-\tilde{a}t)}$ with $\tilde{a} = a\,\mathrm{Re}\{k_{\text{eff}}\}/k_x$, i.e. a translation at speed $\tilde{a}$ instead of a. Any deviation of $\mathrm{Re}\{k_{\text{eff}}\}$ from k_x is therefore a dispersion error, with the mode leading when $\mathrm{Re}\{k_{\text{eff}}\}/k_x > 1$ and lagging if $\mathrm{Re}\{k_{\text{eff}}\}/k_x < 1$. Any non-zero $\mathrm{Im}\{k_{\text{eff}}\}$ is a dissipation error: the mode is damped when $\mathrm{Im}\{k_{\text{eff}}\} < 0$ and amplified when $\mathrm{Im}\{k_{\text{eff}}\} > 0$.

Diffusion. For $\partial_t u = \nu\nabla^2 u$ with $\nu > 0$, the exact solution is $\hat{u}(t) = \hat{u}(0)\,e^{-\nu\|\mathbf{k}\|^2 t}$, a stationary mode decaying at rate $\nu\|\mathbf{k}\|^2$. With $\hat{L}_i^{\nabla^2} = -q_{\text{eff}}^2$, the semi-discrete solution is

$$\hat{u}(t) = \hat{u}(0)\,\underbrace{e^{-\nu\,\mathrm{Re}\{q_{\text{eff}}^2\}\,t}}_{\text{amplitude}}\,\underbrace{e^{-\mathrm{i}\nu\,\mathrm{Im}\{q_{\text{eff}}^2\}\,t}}_{\text{phase}}. \tag{14}$$

In this case, the real part sets the decay rate, so $\mathrm{Re}\{q_{\text{eff}}^2\} > \|\mathbf{k}\|^2$ over-damps the mode, a smaller positive value under-damps it. The imaginary part sets a phase that is absent from the exact solution, translating the mode at speed $\nu\,\mathrm{Im}\{q_{\text{eff}}^2\}/\|\mathbf{k}\|$.

D Stability Analysis

The modal analysis of Appendix C.3 treats each stencil as if it were repeated at every node, i.e. as a translation-invariant discretisation, in which case the modal response fully determines stability. On an irregular point cloud neighbouring stencils differ, and since $w_{j,i}^D$ is not tied to $w_{i,j}^D$, the symmetries that constrain the spectrum of a translation-invariant discretisation are not valid. Stencils that are individually well behaved can then still produce growing global modes. Since SpeND is trained on a local objective, its global stability must be verified directly.

We assemble the local operators into a global matrix $\mathbf{G}^D \in \mathbb{R}^{|\mathcal{P}|\times|\mathcal{P}|}$, so that applying L^D at every node is the product $\mathbf{G}^D\boldsymbol{\phi}$, with $\boldsymbol{\phi}$ the vector of nodal values. Under $\mathrm{d}\boldsymbol{\phi}/\mathrm{d}t = \mathbf{G}^D\boldsymbol{\phi}$, global modes with eigenvalue μ grow if $\mathrm{Re}\{\mu\} > 0$, decay if $\mathrm{Re}\{\mu\} < 0$, and oscillate with constant amplitude if μ is purely imaginary. The first derivative only translates modes, so an ideal discretisation has $\mathrm{Re}\{\mu\} = 0$ for all μ, whereas the Laplacian only damps them, so a stable discretisation requires $\mathrm{Re}\{\mu\} \leq 0$ (Fornberg & Lehto, 2011).

Figure 9 shows the spectra of $\mathbf{G}^D$ on a node set generated by iterative particle shifting, for consistency orders $p = 2, 3, 4$. For both operators, the eigenvalues of RBF-FD generally spread furthest along both axes, followed by SpeND, while those of LABFM are the most concentrated. The one exception is the fourth-order ∂_x operator, for which LABFM spreads furthest along the real axis. Spread away from the ideal axis, along the real axis for ∂_x and the imaginary axis for ∇^2, measures spurious growth, decay or drift, and its ordering is consistent with the imaginary parts of the modal response (Figure 8). Spread along the ideal axis instead reflects the range of wavenumbers the operator resolves, so the compact spectrum of LABFM there, particularly at second order, mirrors the early collapse of its modal response.

E Traditional Mesh-Free Discretisations

Both baselines share the stencil definition of SpeND, with $\mathcal{N}_i$ containing the $N = 30$ nearest nodes to $\mathbf{x}_i$, the node itself included. Both methods reproduce polynomials of degree up to p exactly, giving a truncation error of $\mathcal{O}(s^{p+1-m})$ for an operator of order m.

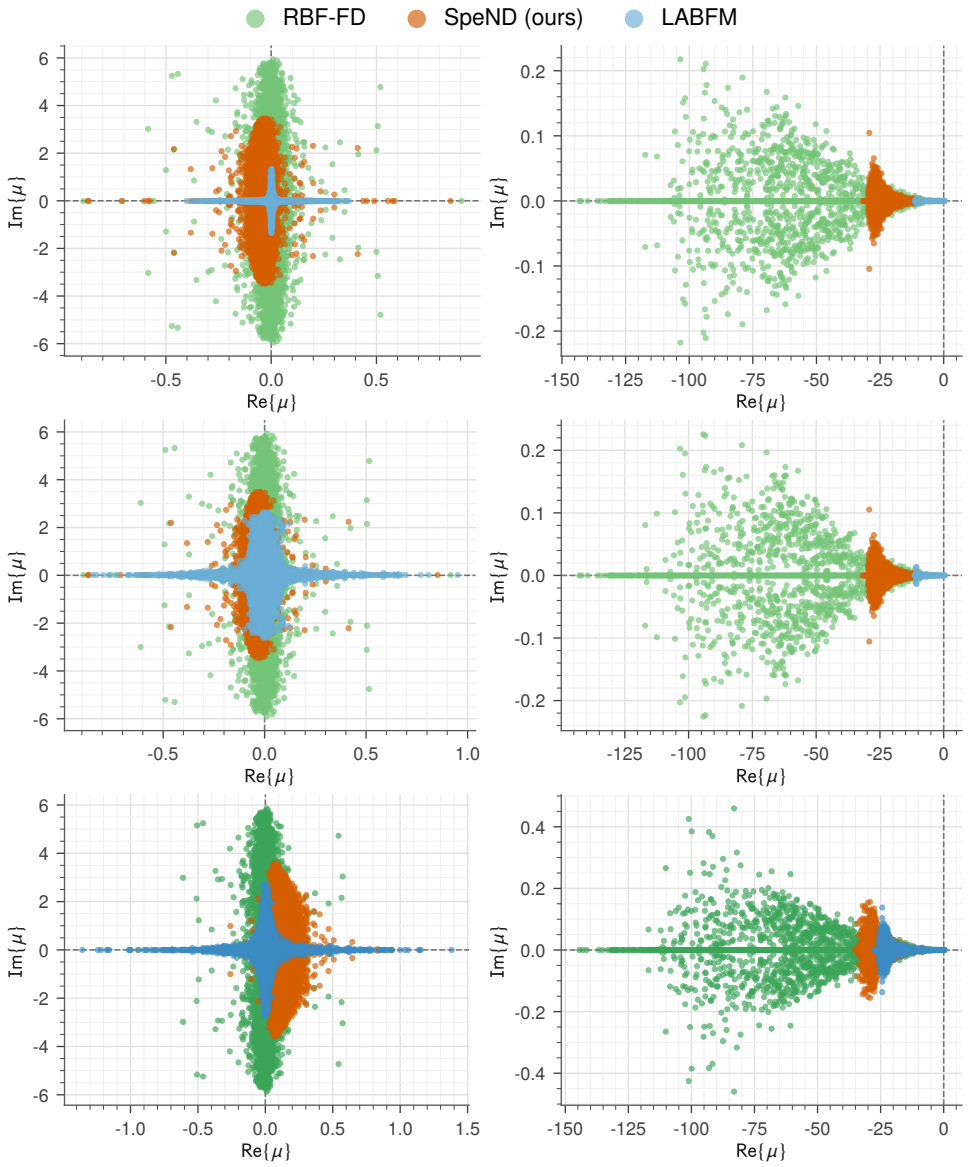

Figure 9: Normalised eigenvalues μ of the global operator $\mathbf{G}^D$ on a node set generated by iterative particle shifting, for ∂_x (left) and ∇^2 (right) at consistency orders $p = 2, 3, 4$ (top to bottom). The ideal spectrum lies on the imaginary axis for ∂_x and on the non-positive real axis for ∇^2.

E.1 LABFM

In LABFM (King et al., 2020), the weights of equation 1 are expanded in a set of anisotropic basis functions (ABFs),

$$w_{j,i}^D = \mathbf{W}_{ji}^\top \mathbf{\Psi}_i^D = W_{ji}^1 \Psi_{i,1}^D + W_{ji}^2 \Psi_{i,2}^D + W_{ji}^3 \Psi_{i,3}^D + \dots, \tag{15}$$

where $\mathbf{W}_{ji}$ collects the ABFs evaluated at $\mathbf{x}_{ji}$ and $\mathbf{\Psi}_i^D$ is a vector of coefficients. Substituting this expansion into the moment system gives a square linear system for the coefficients,

$$\mathbf{A}_i \mathbf{\Psi}_i^D = \mathbf{d}^D, \qquad \mathbf{A}_i = \sum_{j \in \mathcal{N}_i} \mathbf{X}_{ji} \otimes \mathbf{W}_{ji}, \tag{16}$$

where $\mathbf{X}_{ji}$ is the vector of Taylor monomials forming the columns of $\mathbf{V}_i$,

$$\mathbf{X}_{ji} = \left[x_{ji},\, y_{ji},\, \frac{x_{ji}^2}{2},\, x_{ji}y_{ji},\, \frac{y_{ji}^2}{2},\, \frac{x_{ji}^3}{6},\, \dots \right]^\top. \tag{17}$$

For the operators used in this work, the target vector takes the form

$$\mathbf{d}^D = \begin{cases} [1,0,0,0,0,0,\dots]^\top & \text{if } D = \partial_x, \\ [0,1,0,0,0,0,\dots]^\top & \text{if } D = \partial_y, \\ [0,0,1,0,1,0,\dots]^\top & \text{if } D = \nabla^2. \end{cases} \tag{18}$$

Since the constant monomial is excluded, $\mathbf{A}_i \in \mathbb{R}^{N_p \times N_p}$, matching the number of rows of $\mathbf{V}_i$. It is non-symmetric and is solved by LU factorisation with partial pivoting.

Each ABF is paired with one monomial of $\mathbf{X}_{ji}$. If an entry of $\mathbf{X}_{ji}$ corresponds to the multi-index $\boldsymbol{\alpha} = (\alpha_1, \alpha_2)$, the paired ABF combines a radial kernel with a product of Hermite polynomials,

$$W_{ji}^{\boldsymbol{\alpha}} = \frac{\kappa\left(\|\mathbf{x}_{ji}\|/h\right)}{\sqrt{2^{|\boldsymbol{\alpha}|}}}\, H_{\alpha_1}\left(\frac{x_{ji}}{h\sqrt{2}}\right) H_{\alpha_2}\left(\frac{y_{ji}}{h\sqrt{2}}\right), \tag{19}$$

where H_n is the physicists' Hermite polynomial of degree n and κ is the Wendland $C2$ kernel, following King & Lind (2022). The smoothing length is $h = 2.5s$.

E.2 RBF-FD

The RBF-FD baseline uses the polyharmonic spline $\chi(r) = r^5$ augmented with all monomials up to degree p, with the zeroth-order moment included (Flyer et al., 2016b). The kernel has no shape parameter and requires $p \geq 2$. On the same $N = 30$ node stencils, the weights solve the system

$$\begin{bmatrix} \mathbf{K}_i & \mathbf{M}_i \\ \mathbf{M}_i^\top & \mathbf{0} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{w}}_i^D \\ \boldsymbol{\xi}_i \end{bmatrix} = \begin{bmatrix} \mathbf{c}_i^D \\ \mathbf{d}_0^D \end{bmatrix}, \tag{20}$$

where $(\mathbf{K}_i)_{jj'} = \|\bar{\mathbf{x}}_{ji} - \bar{\mathbf{x}}_{j'i}\|^5$, the j-th column of $\mathbf{M}_i^\top$ is $\begin{bmatrix} 1 \\ \mathbf{X}(\bar{\mathbf{x}}_{ji}) \end{bmatrix}$, and $\mathbf{d}_0^D = \begin{bmatrix} 0 \\ \mathbf{d}^D \end{bmatrix}$. The entries of $\mathbf{c}_i^D$ are the continuous differentiator operator applied to the kernel at the central node, $c_{j,i}^D = D\,\chi(\|\mathbf{x} - \bar{\mathbf{x}}_{ji}\|)\big|_{\mathbf{x}=\mathbf{0}}$. The system is solved by dense LU and $\boldsymbol{\xi}_i$ is discarded. Physical weights are recovered as $w_{j,i}^D = \tilde{w}_{j,i}^D / R_i^m$, similarly to SpeND.

F Experimental Details & Additional Results

All three methods run within the same Fortran solver. SpeND weights are generated by calling the trained operators from Fortran through TorchFort (with inference also performed on the CPU), which leaves the rest of the pipeline identical across methods.

F.1 Viscous Burgers Equation

F.1.1 Setup

We solve the two-dimensional viscous Burgers equation in advective form,

$$\partial_t \mathbf{u} + (\mathbf{u} \cdot \nabla)\mathbf{u} = Re^{-1}\nabla^2 \mathbf{u}, \qquad \mathbf{u} = (u, v), \tag{21}$$

with $Re = 100$ on $\Omega = [0,1]^2$, periodic in both directions, for $t \in [0,1]$. The initial condition $u = \sin(2\pi x)$, $v = 0$ keeps $v \equiv 0$ exactly, so the solution is one-dimensional. The equation for u is advanced with the full two-dimensional operators on scattered nodes. A shock forms at $x = 0.5$ and then decays. The reference solution is the Cole–Hopf series, truncated at 29 terms.

Nodes are placed at the cell centres of a Cartesian lattice with spacing $s = 1/(n+1)$, $n \in \{40, 80, 160, 320\}$. Each coordinate is then perturbed independently by $\varepsilon s\,\mathcal{U}(-\frac{1}{2}, \frac{1}{2})$ with $\varepsilon = 0.8$, with a new perturbation at each resolution. All methods use $p = 2$ operators, and SpeND obtains ∂_y by applying the trained ∂_x operator to stencils with the coordinates swapped. Time integration uses 4th order Runge–Kutta. The L_2 error is the root-mean-square difference between the computed and reference u over all nodes at the final time.

F.1.2 Additional Results

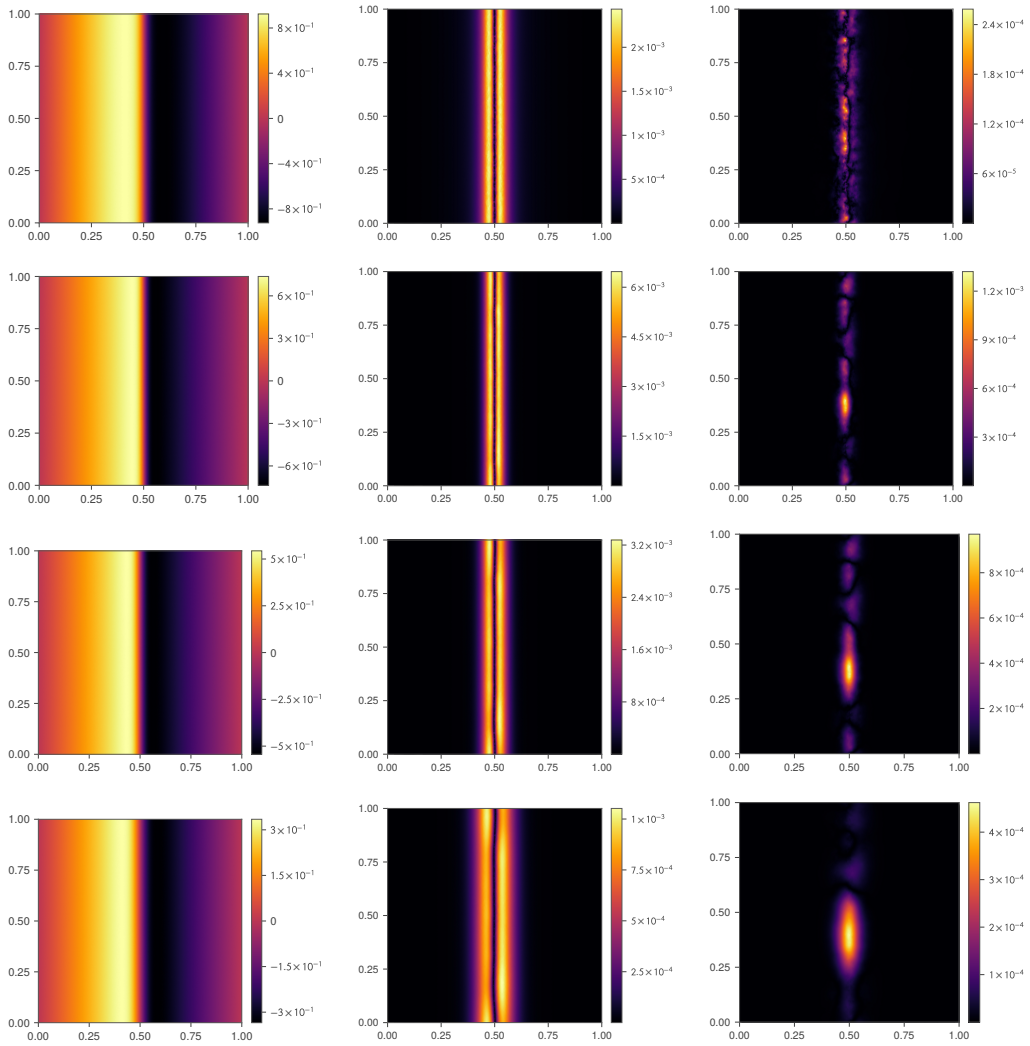

Figure 10: Viscous Burgers problem at mean node spacing $s \approx 3 \times 10^{-3}$. Left: analytical solution, coloured by the u component of velocity. Middle and right: absolute error with respect to the analytical solution, for LABFM and SpeND, respectively. Rows correspond, from top to bottom, to $t = 0.19$, 0.39, 0.59, and 0.99.

Figure 10 shows the error fields of LABFM and SpeND at the finest resolution. The LABFM error is concentrated in a region around the shock and is nearly uniform in y. The SpeND error instead peaks on the shock itself and varies with y. Throughout the simulation SpeND's error remains smaller than LABFM's.

F.2 Poisson's Equation

F.2.1 Setup

We solve Poisson's equation

$$\nabla^2 \phi = f \quad \text{in } \Omega, \tag{22}$$

where Ω is the unit square with a circular obstacle of radius $r_c = 0.2$ removed from its centre. The outer boundaries are periodic, and a Dirichlet condition is imposed on the obstacle boundary. The source term follows from the manufactured solution $\phi = \sin(6\pi x)\sin(6\pi y)$, which is periodic on the unit square, giving $f = -72\pi^2 \sin(6\pi x)\sin(6\pi y)$.

Nodes are placed at the cell centres of a Cartesian lattice with spacing $s = 1/(n+1)$, $n \in \{20, 40, 80, 160, 320, 640\}$, perturbed independently by $\varepsilon s \mathcal{U}(-\frac{1}{2}, \frac{1}{2})$ with $\varepsilon = 1$ in each coordinate, and those inside the obstacle are removed. The obstacle boundary carries equispaced nodes at its surface, which remain fixed while the interior is relaxed with particle shifting. All methods use $p = 3$ Laplacian operators. The sparse system is solved for all methods by BiCGSTAB with a Jacobi preconditioner. The error is the relative root-mean-square difference between the computed and exact ϕ over all nodes.

F.2.2 Additional Results

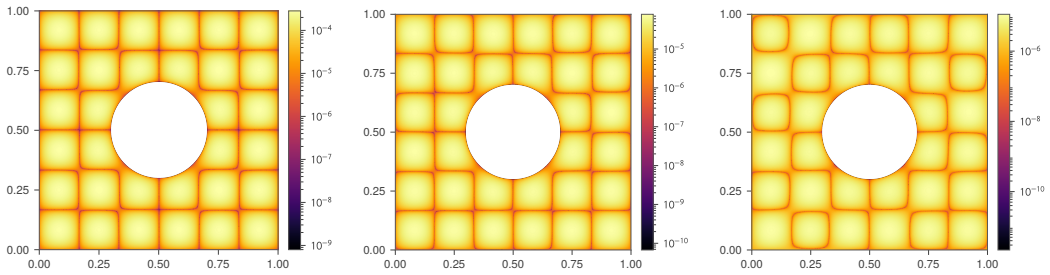

Figure 11: Poisson's equation with average particle spacing $s \approx 1.6 \times 10^{-3}$. Absolute error with respect to the analytical solution, on a log colour scale, for LABFM, RBF-FD and SpeND, from left to right.

Figure 11 shows that the three operators produce errors with a very similar spatial structure. For all methods, the error concentrates at the crests and troughs of the manufactured solution and is smallest along the lines where ϕ vanishes, so each error map inherits the symmetry of ϕ. The methods therefore differ in magnitude rather than in structure, with the maximum absolute error decreasing by roughly an order of magnitude from LABFM to RBF-FD and again from RBF-FD to SpeND.

F.3 Weakly Compressible Navier–Stokes Equations

F.3.1 Setup

We solve the isothermal, weakly compressible Navier–Stokes equations in conservative form,

$$\partial_t \rho + \nabla \cdot (\rho \mathbf{u}) = 0, \tag{23a}$$

$$\partial_t (\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u}\mathbf{u}) = -Ma^{-2} \nabla \rho + Re^{-1} \nabla^2 \mathbf{u}, \tag{23b}$$

where the system is closed by a barotropic equation of state absorbed into the momentum equation. The domain is $\Omega = [-0.5, 0.5]^2$, periodic in both directions, and the flow is advanced to $t = 15$ ($t = 20$ for $s = 1/128$) with $Re = 100$ and $Ma = 0.1$. The reference solution is the Taylor–Green vortex,

$$u(x, y, t) = -\exp\left(-\frac{8\pi^2}{Re}\, t\right) \cos(2\pi x) \sin(2\pi y), \tag{24a}$$

$$v(x, y, t) = \exp\left(-\frac{8\pi^2}{Re}\, t\right) \sin(2\pi x) \cos(2\pi y), \tag{24b}$$

evaluated at $t = 0$ for the initial velocity, with the initial density

$$\rho(x, y, 0) = 1 - \frac{Ma^2}{4} \left[\cos(4\pi x) + \cos(4\pi y)\right]. \tag{25}$$

Node sets with target spacing $s \in \{1/32, 1/64, 1/128\}$ are generated once and shared by all methods, and all methods use $p = 4$ operators. Each method is stabilised by an eighth-order hyperviscous filter applied to ρ, ρu and ρv after the final stage of every time step. The filter order and target damping are common to all methods. Time integration uses the four-stage, third-order low-storage

Runge–Kutta scheme RK3(2)4[2R+]C of Kennedy et al. (2000), with adaptive time stepping. The error is the L_2 difference between computed and reference velocity magnitudes over all nodes, normalised by the reference norm at the first output time.

F.3.2 Additional Results

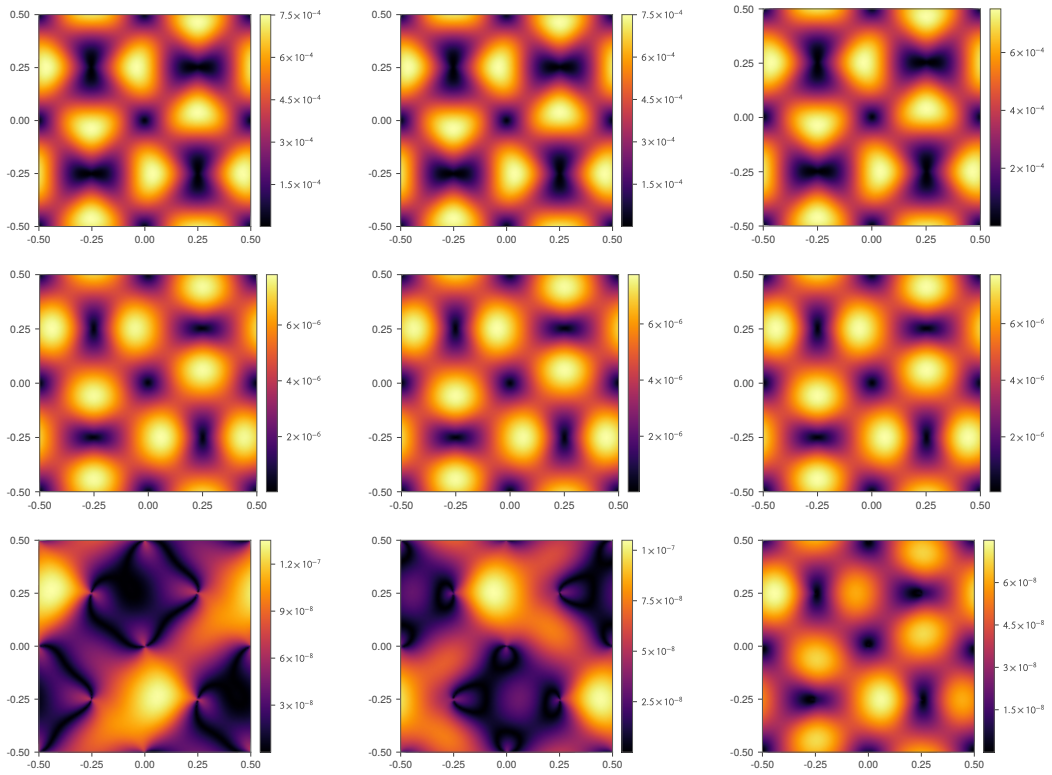

Figure 12: Taylor–Green vortex with average node spacing $s = 1/128$. Colours show the absolute error in velocity magnitude, normalised by the initial reference velocity magnitude. Columns correspond, from left to right, to LABFM, RBF-FD and SpeND, and rows, from top to bottom, to $t = 1$, 7 and 13.

Figure 12 shows the spatial distribution of the error over time. Up to $t = 7$, the three methods produce visually indistinguishable error fields, and their colour bars span the same range. By $t = 13$, only SpeND preserves the symmetric structure of the Taylor–Green vortex, while the error fields of both baselines have lost the symmetry of the original problem.

G Limitations

All experiments are two-dimensional, and although the framework is not restricted to two dimensions, the number of moment conditions and hence the stencil size grow substantially in three dimensions, which we leave to future work. Boundary stencils are trained on truncations by a straight wall and tested only on a smooth circular obstacle, so how training and generalisation behave near complex boundaries remains to be established. The test problems are smooth and laminar at low Reynolds number; they were chosen since they admit analytical or semi-analytical solutions, isolating the error of the spatial discretisation, but as a consequence chaotic flows remain untested. Lastly, predicting the weights is more expensive than constructing them with the classical baselines, so at coarse resolutions, where this cost dominates the wall-clock time, the classical baselines reach a given error faster than SpeND; its advantage appears only once the resolution is fine enough for this one-off cost to be amortised.

H ABLATIONS

All ablations evaluate the ∂_x and ∇^2 operators on the test function of equation 10, on node sets with $n \in \{10, 20, 50, 100, 300\}$ nodes per side and spacing $s = 1/(n-1)$, obtained by disturbing a lattice by up to $s/2$ and relaxing it with 100 particle-shifting iterations, unless specified otherwise. Within each ablation, every variant is fine-tuned for 40 epochs from a common checkpoint and differs from the others only in the ablated setting.

Stencil size. We train operators with $N \in \{20, 30, 40\}$ at the base configuration of Table 2, $\eta = 0.7$, each on a corpus that differs from the others only in stencil size (Figure 13). At the coarsest resolution, $n = 20$, reducing the stencil to $N = 20$ raises the error by roughly $2.5\times$, whereas enlarging it beyond $N = 30$ brings no further gain. Once the field is resolved, the two operators respond differently. The ∇^2 error barely changes across N, while the ∂_x error rises with N. A plausible explanation is the wide training band. With $\eta = 0.7$, the objective spends the additional freedom of a larger stencil on the high wavenumbers that dominate at coarse resolution rather than on the low wavenumbers that dominate once the field is resolved. The training-band ablation below supports this, since narrowing the band improves accuracy at fine resolution.

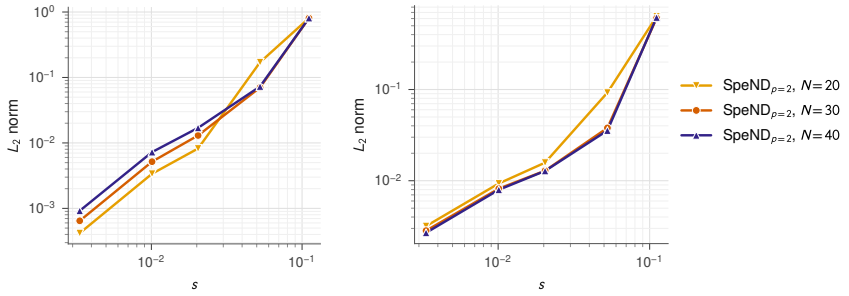

Figure 13: Effect of stencil size N at $\eta = 0.7$. Relative L_2 error under refinement for ∂_x (left) and ∇^2 (right).

Training band. We vary the band limit $\eta \in \{0.2, 0.4, 0.7\}$ at $N = 30$ and $\epsilon = 0.1$ (Figure 14). This setting has the largest effect of those ablated, and it trades accuracy between resolution regimes. At $n = 20$ the highest harmonic of equation 10 lies at $0.74\,k_{\mathrm{Ny}}$, which falls the closest to the band of the $\eta = 0.7$ operators. We can see that the operators trained with $\eta = 0.7$ generally perform better at coarser regions, particularly the Laplacian. At $n = 300$ every harmonic lies within all three bands and the ordering reverses, with $\eta = 0.2$ reducing the error by $8\times$ and $6\times$ relative to $\eta = 0.7$ for ∂_x and ∇^2, respectively. The curves cross between $n = 20$ and $n = 50$. Narrowing the band therefore improves accuracy at the wavenumbers it covers at the expense of those it excludes. The value $\eta = 0.4$ used in Section 4 is a compromise between the two regimes.

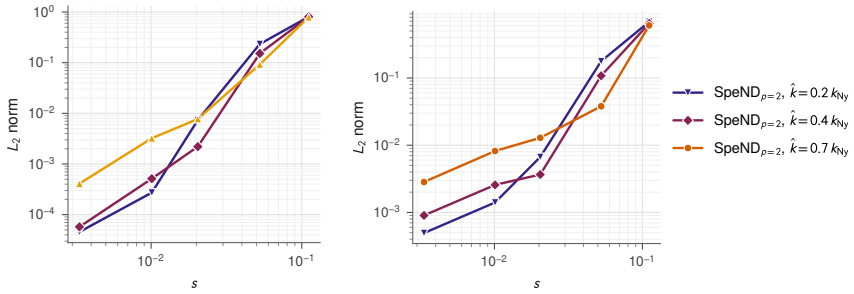

Figure 14: Effect of the band limit η at $N = 30$. Relative L_2 error under refinement for ∂_x (left) and ∇^2 (right).

Moment projection. Replacing Π_i by the identity at inference, with the network unchanged, yields relative errors between 3 and 37 for ∂_x and between 10^2 and 5×10^2 for ∇^2 (Figure 15). These curves show that the moment projection is an essential aspect of the framework provided, ensuring physical consistency and, thus, formal convergence.

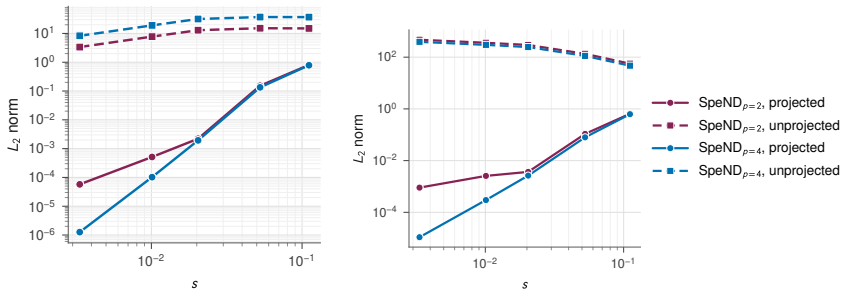

Figure 15: Effect of removing the moment projection at inference, at $p = 2$ and $p = 4$. Relative L_2 error under refinement for ∂_x (left) and ∇^2 (right).

Consistency order at inference. Since $\bar{\mathbf{V}}_i$ is assembled at inference, a network trained under one set of moment conditions can be projected under another. We denote by p_{train} the consistency order imposed by the projection during training and by p_{eval} the order imposed at evaluation. We take the $\eta = 0.4$ network with $p_{\text{train}} = 2$ and the network with $p_{\text{train}} = 4$ used in Section 4, and evaluate each at $p_{\text{eval}} \in \{2, 4\}$ (Figure 16). With $p_{\text{train}} = 2$ and $p_{\text{eval}} = 4$, the operators recover the fourth-order convergence rates and remain close to those with $p_{\text{train}} = 4$ throughout, so on this test a network can be trained at a lower order than the one imposed at evaluation. The reverse does not hold. With $p_{\text{train}} = 4$ and $p_{\text{eval}} = 2$, the error rises by roughly $10\times$ relative to $p_{\text{train}} = p_{\text{eval}} = 2$. When $p_{\text{train}} = 4$, the projection fixes the third- and fourth-order moments during training, and since these set the leading error of the low-wavenumber response, the network is never trained to control it. At $p_{\text{eval}} = 2$ these moments are no longer imposed, and the low-wavenumber response degrades accordingly.

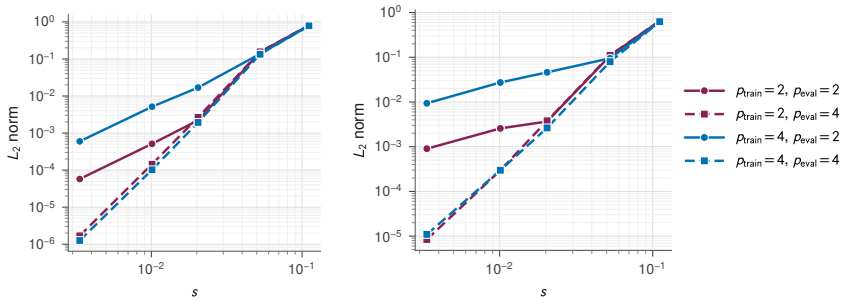

Figure 16: Networks trained at $p = 2$ and $p = 4$, each projected at $p = 2$ and $p = 4$ at inference. Relative L_2 error under refinement for ∂_x (left) and ∇^2 (right).

Node distribution. In place of the node sets above, we evaluate the $p = 2$ operators on two families (Figure 17). The first consists of Cartesian lattices disturbed by $\varepsilon s \mathcal{U}(-\frac{1}{2}, \frac{1}{2})$ per axis with $\varepsilon \in \{0.25, 0.5, 0.75, 1\}$. The second consists of snapshots after 25, 50, 75 and 100 particle-shifting iterations from the $\varepsilon = 1$ lattice, the last being the node set used in the other ablations. SpeND retains the convergence rates of $p = 2$ on every node set, and only showing vertical shift in certain locations. On the disturbed lattices the error of both operators rises monotonically with ε at every $n \geq 50$, by roughly $3\times$ from $\varepsilon = 0.25$ to $\varepsilon = 1$ at $n = 300$. Particle shifting reduces the error of both operators, with most of the gain in the first 25 iterations. Relative to the unrelaxed $\varepsilon = 1$ lattice, these iterations lower the error at $n = 300$ by $2.5\times$ for ∂_x and $3.8\times$ for ∇^2, with little improvement in the further iterations.

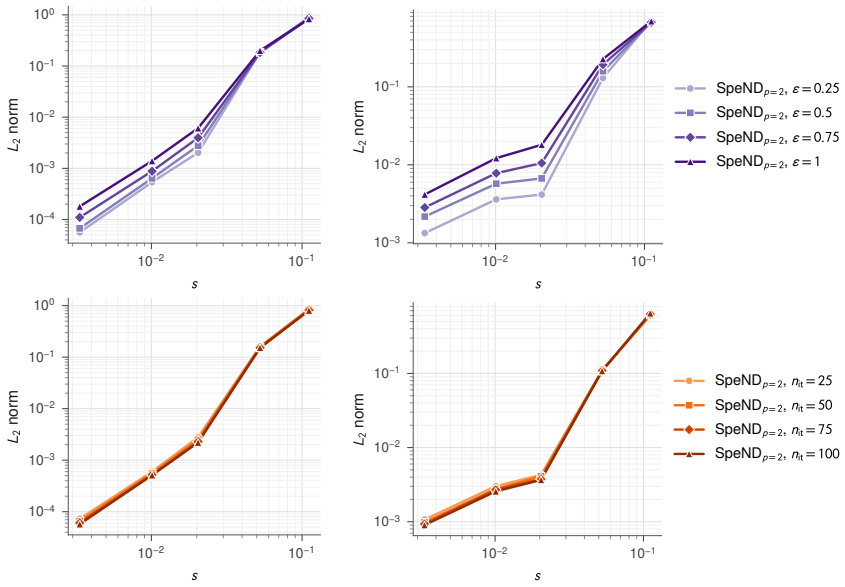

Figure 17: Effect of the node distribution on the $p = 2$ operators. Relative L_2 error under refinement for ∂_x (left) and ∇^2 (right). Top row, disturbed Cartesian lattices with disorder ε. Bottom row, particle-shifted node sets after the indicated number of iterations.

Stencil-size transfer. Since the network shares its per-point maps and pools symmetrically over the stencil, it can be applied without modification to stencils of a size other than the one it was trained on. We denote by N_{train} the stencil size used during training and by N_{eval} the stencil size at evaluation. We take the $\eta = 0.4$, $p = 2$ operators with $N_{\text{train}} = 30$ and evaluate them at $N_{\text{eval}} \in \{20, 30, 40\}$ on the same node sets, without retraining (Figure 18). The output scaling $\sigma_m = \bar{s}_N^{-m}$ of Appendix B depends on the stencil size, and we consider two choices at evaluation. In the first, σ_m is kept at its trained value, computed from N_{train}. In the second, referred to as rescaled, it is recomputed from N_{eval}. With the trained scaling, both operators converge at the tested N_{eval}, however using $N_{\text{eval}} = N_{\text{train}}$ still leads to better results. Although at $N_{\text{eval}} = 40$ is barely distinguishable than when $N_{\text{eval}} = 30$ for ∂_x. Rescaling is considerably worse, raising the error at $n = 300$ by $17\times$ and $34\times$ for ∂_x at $N_{\text{eval}} = 20$ and 40, and by $7\times$ and $8\times$ for ∇^2, relative to the trained scaling. A single network can therefore serve a range of stencil sizes at a moderate loss of accuracy, provided its output scaling is left as trained.

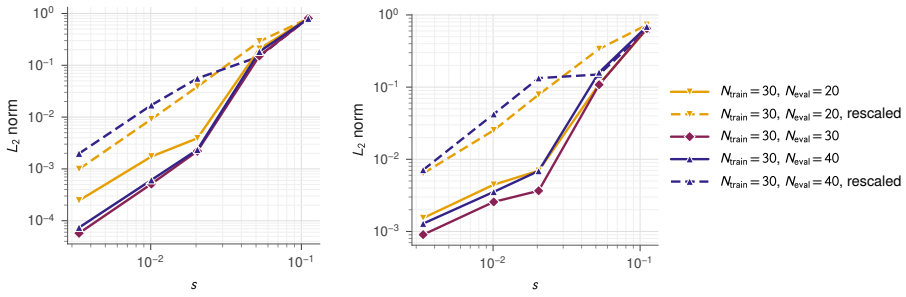

Figure 18: Operators trained at $N_{\text{train}} = 30$ evaluated at $N_{\text{eval}} \in \{20, 30, 40\}$ without retraining. Relative L_2 error under refinement for ∂_x (left) and ∇^2 (right). Solid lines keep the trained output scaling σ_m, and dashed lines recompute it for the evaluation stencil size.

I HARDWARE & COMPUTATIONAL COST

Training proceeds through a chain of warm-started stages, with the first stage trained from scratch and each later stage initialised from its predecessor. The second-order operators are taken from stage 3 and the fourth-order operators from stage 4. The first two stages ran on a cluster node with a single NVIDIA L40S and eight AMD EPYC cores, and the last two on a workstation with a single NVIDIA GeForce RTX 5080 and an AMD Ryzen 7 7800X3D. Both machines use PyTorch 2.8.0 built against CUDA 12.9, with float32 arithmetic and TF32 matrix multiplication. When the trained operators are exported for use in the solvers, all parameters and arithmetic are cast to float64, so that the projection layer enforces the moment conditions to double-precision. The stages are chained to reuse earlier checkpoints and reduce compute, rather than as part of the method.

Table 3 reports the time spent in the training loop. The fourth-order stage is roughly 20% slower per step than its second-order predecessor since the moment system grows from $N_p = 5$ to $N_p = 14$ rows, enlarging the least-squares problem solved in the projection layer.

Table 3: Training time per stage and cumulative, in hours, for a single operator on a single GPU.

			Stage		Cumulative	
Stage	GPU	Epochs	∂_x	∇^2	∂_x	∇^2
1. Pretraining (Adam)	L40S	40	1.53	1.57	1.53	1.57
2. Fine-tuning (Muon)	L40S	20	1.22	1.22	2.75	2.79
3. Modal response, $p = 2$	RTX 5080	40	2.31	2.30	5.06	5.09
4. Modal response, $p = 4$	RTX 5080	40	2.79	2.72	7.85	7.81

The Burgers and Poisson experiments run on the AMD Ryzen 7 7800X3D described above using OpenMP with 16 threads, and the Taylor–Green vortex runs on the same processor using MPI with four ranks, one per physical core. All methods share the same solver implementation and differ only in the discretisation, with all solver arithmetic carried out in float64. Reported times cover the full pipeline, from node generation and neighbour search through the computation of the stencil weights to the solve itself.