

Local and Sparse Kernel Operator Learning for Incompressible Flows: An Operator Lagrange Formulation

Ramansh Sharma^a, Matthew Lowery^a, Houman Owhadi^b and Varun Shankar^{a,*}

^aKahlert School of Computing, University of Utah, UT, USA

^bDepartment of Computing and Mathematical Sciences, California Institute of Technology, CA, USA

ARTICLE INFO

Keywords:

operator learning
incompressible flow
divergence-free kernels
compactly supported kernels
RBF-FD
sparse Cholesky
GPU computing
2020 MSC: 65D12
65D15
46E22
65F50
76D05

ABSTRACT

We present a local and sparse kernel method for learning the solution operators of incompressible flows. It builds on our earlier property-preserving kernel method, whose global output reconstruction and dense operator regression scale poorly. Our method replaces the global output reconstruction by local divergence-free interpolants with divergence-free polynomials of any degree, solved in parallel on the GPU. These interpolants are divergence free, periodic on periodic domains, and, with a multiscale kernel, approach Kolmogorov statistics under refinement. A compactly supported kernel, positive semidefinite in any input dimension, makes the operator regression sparse. Its support radius, and hence the sparsity, is fitted to data, and optional affine polynomials in the input capture global trends. Operator Lagrange weights join the two. They cost one sparse solve per new input and act on the training fields before the output reconstruction, so their local coefficients are never formed. We show that, unlike direct evaluation, Lagrange evaluation does not amplify solve errors by the cancellation among spatial coefficients. On ten fluid benchmarks, validation selects sparse matrices nearly everywhere, with densities that mostly decrease as the training set grows, and affine augmentation lowers the error by more than an order of magnitude for forced isotropic turbulence. These sparse models are at least as accurate as our earlier dense method and more accurate than neural operators. Lagrange evaluation is as accurate as direct evaluation with local output reconstructions, and up to nearly three orders of magnitude more accurate with global ones. We also present GPU timings and storage.

1. Introduction

Uncertainty quantification, design, and inverse problems for incompressible flows require the velocity field for many initial conditions, boundary conditions, or forcing functions, and the repeated solution of the Navier–Stokes equations then dominates the cost [43, 59–61]. One remedy is to learn the solution operator, the map from the problem data to the velocity, from a collection of simulations [7]. Reduced-order models have long served this purpose [5, 42], and neural operators such as DeepONets, Fourier and graph neural operators, transformer-based models, and kernel neural operators now learn such maps from large data sets [32–34, 36, 37, 70]. We seek an operator learner for incompressible flows that satisfies four criteria. First, every prediction should be divergence free, and periodic on periodic domains, however accurately the operator itself is learned. Second, the cost of the output representation should grow linearly with the number of output points, so that fine three-dimensional outputs remain affordable. Third, the regression should handle ten thousand or more training functions without factoring a dense matrix. Fourth, every stage should run in parallel on a graphics processing unit (GPU).

Kernel methods offer an attractive alternative to neural operators [2, 29, 38]. Batlle et al. [2] learn an operator from point samples in two steps. An operator regression, a kernel regression on the space of sampled inputs, maps the samples of the input function to the samples of the output function. An output reconstruction, a kernel interpolant of these samples in space (or spacetime), then recovers the output function. Since both steps are kernel interpolation or regression problems, the learned operator has a closed form, few parameters, and an error analysis. For flows, the output representation is the natural place to impose physics [30, 45]. In our previous work [53], we therefore regressed the coefficients of a divergence-free kernel interpolant of the output instead of its point samples, so that every prediction is exactly divergence free regardless of the accuracy of the operator regression. We called this the property-preserving

*Corresponding author

✉ ramansh@cs.utah.edu (R. Sharma); mlowery@cs.utah.edu (M. Lowery); owhadi@caltech.edu (H. Owhadi); shankar@cs.utah.edu (V. Shankar)

kernel method (PPKM). PPKM satisfies the first criterion, but both of its kernel problems are global. Its operator regression factors one dense kernel matrix on all N training inputs, at a cost of $O(N^3)$ operations and $O(N^2)$ storage, and its output reconstruction solves one dense system on all output points. Both limits bind for three-dimensional turbulence, where a training set has ten thousand or more functions and an output has thousands of points. Meeting the other criteria poses three tasks. The output reconstruction must become local without losing the properties of the global one, the operator regression must become sparse, and the two must be joined without forming the local coefficients of every training field.

The first task draws on radial basis function-generated finite differences (RBF-FD). RBF-FD methods replace a global radial basis function (RBF) interpolant, whose dense system couples all nodes, by small interpolants on the stencil of each node, so that every linear system has a fixed size and is independent of the others [20, 21, 55, 62, 69]. Over the last two decades, RBF-FD methods have been stabilized for convection-dominated problems [22, 49], applied to the Navier–Stokes equations [18, 55], extended to overlapped stencils, surfaces, and moving domains [48, 51, 52], and formulated in the least-squares sense [63]. They also have error bounds [10], node generators for irregular domains [50], and GPU implementations [6]. In particular, polyharmonic splines (PHS) augmented with polynomials converge at the order of the polynomial degree and have no shape parameter [3, 19]. For flows, the local interpolants must also be divergence free. Divergence-free kernels, obtained by applying $-\Delta I + \nabla \nabla^T$ to a scalar kernel [11, 24, 39, 40, 66], have this property by construction and have been used both globally and in partition-of-unity form [12, 25], and Yelnyk [71] combined divergence-free PHS with polynomials in local Hermite interpolation for Stokes problems. However, the high order of PHS-based RBF-FD comes from the polynomials, so a local divergence-free interpolant of high order needs divergence-free polynomials of any degree, in two and three dimensions, that can be evaluated anywhere on the stencil.

Our approach is to build these polynomials by differentiating products of Legendre polynomials, select an independent subset on each stencil by a column-pivoted QR factorization, and combine them with a divergence-free PHS on the stencils of RBF-FD, used for interpolation instead of differentiation. Every local interpolant is then exactly divergence free, and building the stencils from periodic images of the nodes makes the output reconstruction periodic. For turbulence, we replace the scale-free PHS by the multiscale Matérn kernel of Sharma et al. [53], whose weights encode the Kolmogorov scaling of the velocity increments, so that the output reconstruction approaches prescribed Kolmogorov statistics as the nodes are refined. Matérn kernels and PHS are Green's functions of closely related operators, and a Matérn kernel tends to a PHS as its length grows, so this change adds length scales without changing the construction. Since the local systems are independent and of fixed size, we assemble, factor, and evaluate them in batches on the GPU, and our output reconstruction is embarrassingly parallel.

For the second task, the dense kernel matrix of the operator regression can be approximated by low-rank, Nyström, and random-feature methods, inducing points, structured kernel interpolation, or sparse approximate factors of its inverse [1, 13, 17, 26, 44, 47, 56, 57, 68, 72]. In contrast, a compactly supported kernel gives a sparse matrix that can be factored exactly by sparse Cholesky. Its obstacle is the dimension of the input. A sampled input function has thousands of coordinates, but no compactly supported radial kernel is positive definite in every dimension [65]. Our kernel therefore determines its support from a few principal coordinates of the input, in which a compactly supported function is positive definite, and treats the remaining coordinates with a Matérn factor, which is positive definite in every dimension. The product is positive semidefinite in any input dimension. Its support radius sets the sparsity of the kernel matrix, and we choose it from the data instead of fixing it in advance. A restricted likelihood criterion, with a smooth count of the interacting training pairs as an optional penalty, fits the shape of the kernel and the regularization, and the error on validation fields then selects the radius, so that nearly dense matrices remain among the candidates when accuracy requires them.

However, a compactly supported kernel returns to zero away from the training inputs, so it represents trends in the input poorly, although many operators depend nearly linearly on part of their input. We therefore augment the operator regression with affine polynomials in the leading principal coordinates of the input, which it reproduces exactly, as in polynomially augmented kernel interpolation. The resulting saddle-point system is solved by a sparse Cholesky factorization of the kernel matrix followed by a QR factorization of the whitened polynomial columns, which avoids forming the Schur complement [4]. JAX has no sparse direct solver that keeps a factorization between calls, so we factor the kernel matrix on the GPU with cuDSS, through a primitive of our KernelPack-JAX framework that exposes cuDSS to compiled JAX programs [8].

The third task leads to operator Lagrange weights. Since the same kernel regresses every output coefficient, a prediction is a linear combination of the training outputs, with weights that depend only on the new input. These

weights cost one sparse solve per new input, regardless of the size of the output, and they give the coefficients on every stencil at once (Figure 1). Because the output reconstruction is linear, the weights can also be applied to the sampled training fields before the output reconstruction, so the local coefficients of the training fields, which would take tens of times the storage of the fields themselves, are never formed. Weights and coefficients give the same prediction in exact arithmetic but round differently. We show that direct evaluation through coefficients amplifies solve errors by a factor that measures the cancellation among the spatial coefficients, while Lagrange evaluation does not.

We test our method in two steps. On scattered nodes, our local output reconstruction converges at close to orders 2, 4, and 6, keeps the divergence and the periodic mismatch at rounding level, and approaches prescribed Kolmogorov statistics under refinement at the separations that the nodes resolve. Beyond a few thousand output points, it is also cheaper to set up than a global output reconstruction, and its memory grows only linearly with their number. We then compare kernels with a fixed and a fitted shape, each with and without affine augmentation, on ten fluid benchmarks with up to ten thousand training functions. Validation selects sparse kernel matrices on all benchmarks except forced isotropic turbulence and the unaugmented Taylor–Green models, and on most of them the selected density decreases as the training set grows. With these sparse matrices, the test errors decrease steadily with the number of training functions. At the training sizes of our earlier study [53], our models are at least as accurate as dense PPKM and the kernel method of Batlle et al. [2] on every data set, and more accurate than the neural operators reported there. Affine augmentation reduces the test error for forced isotropic turbulence by more than an order of magnitude, and together with the fitted shape it does the same for turbulent species transport. In contrast, the five flows whose training inputs vary along a single direction are insensitive to both choices. Finally, Lagrange evaluation is as accurate as direct evaluation with our local output reconstruction, whose amplification factors are at most about 10^4 , and up to nearly three orders of magnitude more accurate with global output reconstructions, whose amplification factors reach 6×10^9 .

The remainder of this paper is organized as follows. Section 2 constructs the local property-preserving output reconstruction, Section 3 the sparse operator regression and its affine augmentation, and Section 4 the operator Lagrange weights, their rounding errors, and the GPU implementation. Section 5 presents the numerical experiments, and Section 6 our conclusions.

1.1. Problem formulation

We consider an incompressible fluid of constant density, governed by

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = \nu \nabla^2 \mathbf{u} - \frac{1}{\rho} \nabla p + \frac{1}{\rho} \mathbf{f}, \quad \text{in } \Omega \times (0, T], \quad (1)$$

$$\nabla \cdot \mathbf{u} = 0, \quad \text{in } \Omega \times (0, T], \quad (2)$$

$$\mathbf{u}(\mathbf{x}, 0) = \mathbf{u}_0(\mathbf{x}), \quad \text{in } \Omega, \quad (3)$$

$$\mathcal{B} \mathbf{u} = \mathbf{g}, \quad \text{on } \partial \Omega \times (0, T], \quad (4)$$

where $\mathbf{u}$ is the velocity, p the pressure, ν the kinematic viscosity, ρ the density, $\mathbf{f}$ the body force, and $\mathcal{B}$ the boundary operator. Our operators map initial data (a velocity $\mathbf{u}_0$, a vorticity, or, for a buoyancy-driven flow, a temperature), the boundary data $\mathbf{g}$, or the forcing $\mathbf{f}$ to the velocity. The training data are simulations of (1) from Sharma et al. [53] (Appendix D.1).

Let $\mathcal{A}$ and $\mathcal{V}$ be separable Banach spaces of input and output functions, and let $\mathcal{G} : \mathcal{A} \rightarrow \mathcal{V}$ be the operator we wish to learn. An input $a \in \mathcal{A}$ and its output $v \in \mathcal{V}$ are defined on domains Ω_a and Ω_v , which may be spatial, spacetime, or parametric, with

$$a : \Omega_a \rightarrow \mathbb{R}^{d_a}, \quad v : \Omega_v \rightarrow \mathbb{R}^d. \quad (5)$$

Here d_a and d are the numbers of input and output components, and D_a and D_v are the dimensions of the domains. We sample inputs at $X = \{x_1, \dots, x_n\} \subset \Omega_a \subset \mathbb{R}^{D_a}$, which gives vectors $\mathbf{a} \in \mathbb{R}^{nd_a}$. For spatial outputs, $D_v = d$, and we sample at $Y = \{y_j\}_{j=1}^m \subset \Omega_v \subset \mathbb{R}^d$, which gives vectors $\mathbf{v} \in \mathbb{R}^{md}$ stacked in component blocks, each holding one velocity component at all points of Y , as in Sharma et al. [53]. For spacetime outputs, $D_v = d + 1$, and sampling the m spatial sites at N_t times gives vectors in $\mathbb{R}^{dmN_t}$ (Section 2.4).

Given N training pairs $\{(a_i, v_i)\}_{i=1}^N$ with sampled vectors $\{(\mathbf{a}_i, \mathbf{v}_i)\}_{i=1}^N$, we write the learned operator, as in Sharma et al. [53], as

$$v \approx \tilde{\mathcal{G}}(a) = \chi \circ \tilde{f} \circ \varphi(a), \quad (6)$$

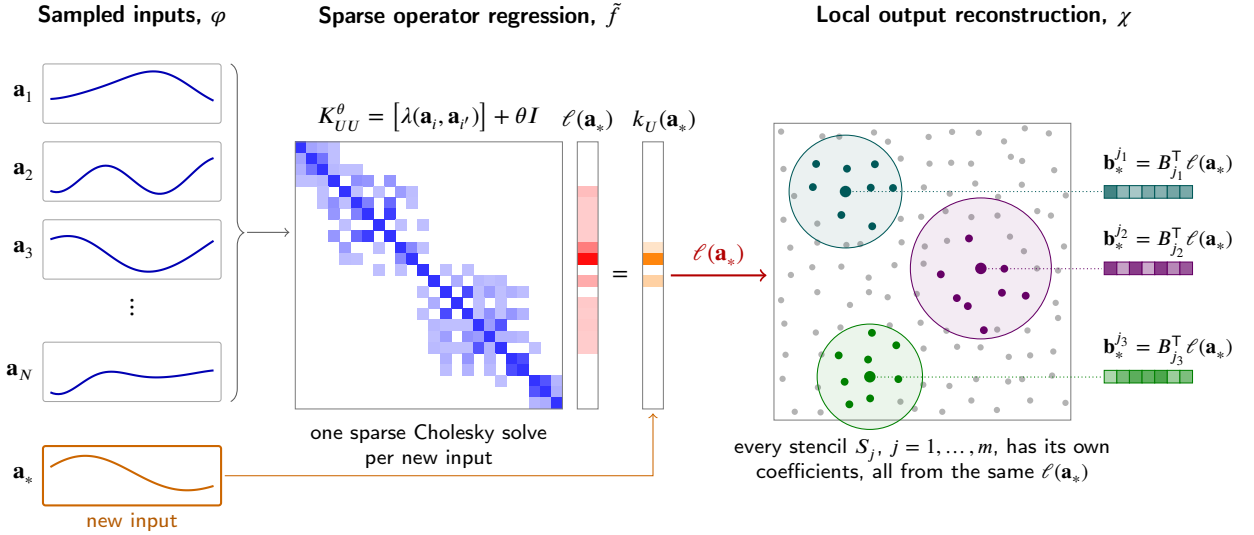


Figure 1: Overview of our method for a new input $\mathbf{a}_*$. The compactly supported kernel λ of Section 3 gives a sparse matrix K_{UU}^θ on the sampled training inputs (left and center). One sparse solve with the kernel values $k_U(\mathbf{a}_*)$ gives the operator Lagrange weights $\ell(\mathbf{a}_*)$ of Section 4, shown here without their affine correction. The same weights give the coefficients $\mathbf{b}_*^j = B_j^\top \ell(\mathbf{a}_*)$ of the local divergence-free interpolant of Section 2 on every stencil S_j of the output (right), where the rows of B_j are the coefficients of the training fields on S_j . In practice, we apply the weights to the sampled training fields before the output reconstruction, which gives the same prediction (Section 4.2).

where φ samples the input, the operator regression $\tilde{f}$ maps the samples to the coefficients of the output, and the output reconstruction χ maps these coefficients to a function in space (or spacetime). This structure follows the formulation of Batté et al. [2], in which $\tilde{f}$ maps input samples to output samples, with the difference that $\tilde{f}$ predicts the coefficients of a divergence-free interpolant. In PPKM [53], χ is a global divergence-free interpolant and $\tilde{f}$ a dense operator regression. Here, χ is a collection of local divergence-free interpolants (Section 2), and $\tilde{f}$ is a sparse operator regression (Section 3). Figure 1 shows how operator Lagrange weights join the two (Section 4).

We use the index i for training functions, j for stencils, k for components and coordinates, q for new (query) inputs and query points, and t for time. Each stencil has n_s nodes. A subscript $*$ marks a quantity of a new input, a superscript $\star$ a quantity at query points, and a hat a computed quantity in the error analysis. The superscript T denotes the transpose, I an identity matrix, and $\|\cdot\|_F$ the Frobenius norm.

2. Local property-preserving output reconstruction

We now construct the output reconstruction map χ of (6). In PPKM [53], χ is a single divergence-free kernel interpolant on all m output points, so its linear system grows with the output resolution. We replace it by local interpolants, one on the stencil of each output point. Each local interpolant combines a divergence-free kernel with divergence-free polynomials, so that it is divergence free for any coefficient vector and reproduces low-degree divergence-free fields. Its system has a fixed size as m grows. We first describe the kernel, the polynomials, and the local system, and then the extensions to turbulent fields, periodic domains, and spacetime outputs. Nothing in this section depends on the operator regression of Section 3.

2.1. Divergence-free interpolation on stencils

2.1.1. The divergence-free kernel

The output points Y serve as the nodes of the output reconstruction. For each node $y_j \in Y$, let the stencil $S_j = \{\tilde{y}_{j,1}, \dots, \tilde{y}_{j,n_s}\}$ consist of its n_s nearest nodes. In the tests of Section 5.1 on bounded domains, we instead use the boundary-aware stencils of KernelPack-JAX [8], which keep only nodes visible from the center inside the domain. In periodic domains, some of these are translated images of nodes (Section 2.3). For smooth outputs, we use

the cubic PHS $\phi(x, y) = \phi_3(\|x - y\|_2)$, with radial function

$$\phi_3(r) = r^3, \quad (7)$$

which has no shape parameter to choose (Section 2.2 relates it to the Matérn kernels that we use when a length scale is wanted). For the sixth-order tests of Section 5.1, we use $\phi(x, y) = \|x - y\|_2^5$ in the same way. Applying the operator $-\Delta I + \nabla \nabla^\top$ to a scalar kernel gives a matrix-valued kernel whose columns are divergence free [25, 39],

$$\Phi_{\text{DF}}(x, y) = (-\Delta_x I + \nabla_x \nabla_x^\top) \phi(x, y). \quad (8)$$

For the cubic PHS, with $z = x - y$ and $r = \|z\|_2 > 0$,

$$\Phi_{\text{DF}}(x, y) = -3drI + 3zz^\top/r, \quad \Phi_{\text{DF}}(y, y) = 0. \quad (9)$$

Each column of Φ_{DF} has zero divergence in the distributional sense, and zero classical divergence away from its center [25]. At the center the kernel is continuous but not differentiable. For example, along $z = t\mathbf{e}_k$, with $\mathbf{e}_k$ the k th coordinate vector, its k th diagonal entry is $-3(d-1)|t|$.

2.1.2. Divergence-free polynomials of any degree

A PHS interpolant converges at the order set by the degree of its polynomial augmentation [3, 19]. A divergence-free local interpolant of high order therefore needs divergence-free vector polynomials of any degree ϖ , in 2D and 3D, that can be evaluated at any point and not only at the stencil nodes. Our approach is to differentiate scalar potentials. On S_j , let $\xi = (y - \bar{y}_j)/\rho_j$ be centered and scaled coordinates, where $\bar{y}_j = n_s^{-1} \sum_l \bar{y}_{j,l}$ is the centroid of the stencil and $\rho_j = \max_l \|\bar{y}_{j,l} - \bar{y}_j\|_2$ its radius. From the products $J_\alpha(\xi) = \prod_k P_{\alpha_k}(\xi_k)$ of normalized Legendre polynomials P_n , of total degree $|\alpha| \leq \varpi + 1$, we form the rotated gradients in 2D and the curls of the potentials times the coordinate vectors in 3D,

$$g_\alpha(y) = \nabla_\xi^\perp J_\alpha(\xi), \quad d = 2, \quad (10)$$

$$g_{\alpha,k}(y) = \nabla_\xi \times (J_\alpha(\xi)\mathbf{e}_k), \quad d = 3, \quad k = 1, 2, 3, \quad (11)$$

where α is a multi-index, $\nabla_\xi^\perp = (\partial_{\xi_2}, -\partial_{\xi_1})^\top$. These candidates have degree at most ϖ and are divergence free in ξ , and hence in y , since the scaling is uniform. Together, they span the space of divergence-free vector polynomials of degree at most ϖ , which has dimension $d \binom{\varpi+d}{d} - \binom{\varpi+d-1}{d}$ (Appendix C.1).

The candidates are not independent. Constant potentials give zero fields, different potentials can give the same field, and a particular stencil can add further dependencies. Let $g_1, \dots, g_{n_{\text{cand},j}}$ be the candidates, let $G_j(y) = [g_1(y) \cdots g_{n_{\text{cand},j}}(y)] \in \mathbb{R}^{d \times n_{\text{cand},j}}$ hold their values at y , and let $G_j \in \mathbb{R}^{d n_s \times n_{\text{cand},j}}$ hold their values at the stencil nodes in component block order. We select an independent subset by column-pivoted QR,

$$G_j \Pi_j = Q_j R_j, \quad (12)$$

and retain the first $n_{P,j}$ pivoted columns, where $n_{P,j}$ counts the diagonal entries of R_j larger than 10^{-12} times the first. Let I_j hold the indices of the retained candidates, $G_{j,I}$ the corresponding columns of G_j , $Q_{j,1}$ the first $n_{P,j}$ columns of Q_j , and $R_{j,11} \in \mathbb{R}^{n_{P,j} \times n_{P,j}}$ the leading block of R_j . Then

$$G_{j,I} = Q_{j,1} R_{j,11}, \quad G_{j,I} R_{j,11}^{-1} = Q_{j,1}. \quad (13)$$

The orthonormal columns of $Q_{j,1}$ are only values at the nodes. To obtain functions, we apply the same triangular transformation to the retained candidate functions $G_{j,I}(y) = [g_c(y)]_{c \in I_j}$,

$$\mathbf{P}_j(y) = G_{j,I}(y) R_{j,11}^{-1}. \quad (14)$$

By (13), $\mathbf{P}_j = Q_{j,1}$ at the stencil nodes, so the polynomial block of the local system is well conditioned. Every column of $\mathbf{P}_j(y)$ is a fixed combination of divergence-free fields, and is thus divergence free everywhere. At a new point, we

evaluate the retained candidates and apply $R_{j,11}^{-1}$ by a triangular solve. If the stencil geometry forces the rank below the full dimension, the interpolant reproduces only the retained subspace.

We batch this construction over the stencils. On stencils that are certainly of full rank, pivoting is unnecessary, and we use an unpivoted QR factorization instead (Appendix A.2).

2.1.3. The local system

On S_j , we determine the kernel and polynomial coefficients together, from the interpolation conditions and the polynomial moment conditions. The saddle-point matrix

$$\mathbf{A}_j = \begin{bmatrix} \Phi_j & \mathbf{P}_j \\ \mathbf{P}_j^\top & 0 \end{bmatrix} \in \mathbb{R}^{r_j \times r_j}, \quad (15)$$

has size $r_j = dn_s + n_{P,j}$, where $\Phi_j \in \mathbb{R}^{dn_s \times dn_s}$ is the Gram matrix of the divergence-free kernel on S_j . For training field i , the coefficient vector

$$\mathbf{b}_i^j = \begin{bmatrix} \mathbf{b}_{i,\Phi}^j \\ \mathbf{b}_{i,P}^j \end{bmatrix} \in \mathbb{R}^{r_j}$$

holds kernel coefficients $\mathbf{b}_{i,\Phi}^j \in \mathbb{R}^{dn_s}$ and polynomial coefficients $\mathbf{b}_{i,P}^j \in \mathbb{R}^{n_{P,j}}$, and defines the local interpolant

$$\chi_i^j(y) = \sum_{l=1}^{n_s} \Phi_{\text{DF}}(y, \tilde{y}_{j,l}) \mathbf{b}_{i,\Phi}^{j,l} + \mathbf{P}_j(y) \mathbf{b}_{i,P}^j, \quad (16)$$

where $\mathbf{b}_{i,\Phi}^{j,l} \in \mathbb{R}^d$ holds the kernel coefficients of node l . Interpolation of the training field at the stencil nodes (with the value of the original node at a periodic image) and orthogonality of the kernel coefficients to the polynomials give

$$\mathbf{A}_j \mathbf{b}_i^j = \begin{bmatrix} \mathbf{v}_i|_{S_j} \\ 0 \end{bmatrix}, \quad (17)$$

where $\mathbf{v}_i|_{S_j} \in \mathbb{R}^{dn_s}$ holds the sampled velocities on the stencil. Since $\mathbf{P}_j$ has full column rank, $\mathbf{A}_j$ is nonsingular whenever Φ_j is positive definite on the null space of $\mathbf{P}_j^\top$. Conditional positive definiteness of the scalar PHS does not guarantee this for the matrix kernel on a particular stencil, so we check the solves numerically (Appendix A.2). We factor all local systems by batched LU on the GPU.

Evaluation To reconstruct a prediction at query points $y_q^\star \in Y^\star$, $q = 1, \dots, m^\star$, we assign each query point to a stencil $S_{j(q)}$ and evaluate that stencil's basis there. Let $\Phi_{j(q),q,\text{comp}}^\star \in \mathbb{R}^{d \times dn_s}$ hold the kernel values in the component block order of the coefficients (Appendix A.1) and $\mathbf{P}_{j(q),q}^\star \in \mathbb{R}^{d \times n_{P,j(q)}}$ the polynomial values, and let

$$\mathbf{A}_q^\star = \begin{bmatrix} \Phi_{j(q),q,\text{comp}}^\star & \mathbf{P}_{j(q),q}^\star \end{bmatrix} \in \mathbb{R}^{d \times r_{j(q)}}. \quad (18)$$

For coefficients $\mathbf{b}_*^j$ predicted by the operator regression (Section 4.1), the reconstructed velocity is then

$$\tilde{\mathbf{v}}_q^\star = \mathbf{A}_q^\star \mathbf{b}_*^{j(q)} \in \mathbb{R}^d, \quad (19)$$

and $\tilde{\mathbf{v}}^\star$ stacks the $\tilde{\mathbf{v}}_q^\star$ in component blocks. In the operator benchmarks, the query points are the nodes themselves, each with its own stencil. Appendix A.3 gives the assignment of other points.

The output reconstruction is divergence free on each stencil, for any coefficients. However, the local interpolants are not coupled, so the reconstructed field can jump where the assigned stencil changes, and its divergence is controlled only within each stencil.

On the other hand, every system has the fixed size r_j , independent of the number of output points m and of the number of training functions N . Refining the output adds systems but does not enlarge them. The systems are also independent of each other, so their assembly, factorization, and evaluation are embarrassingly parallel, and we process

them in batches on the GPU with no communication between stencils. In contrast, a global output reconstruction couples all output points in one dense system with dm unknowns, whose factorization costs $O((dm)^3)$ operations and does not decompose into independent parts. Section 5.1.5 measures both costs. Section 4.2 shows how the same systems serve every prediction.

2.2. Multiscale Matérn output reconstruction for turbulence

Turbulent velocity fields vary over many length scales, which a kernel can represent only if it has length scales of its own. Polyharmonic splines have none, since they are homogeneous (dilating the argument of r^3 or r^5 multiplies it by a constant, and so does the derived matrix kernel), and sums of dilated copies add nothing. Matérn kernels have one length each, and they are close relatives of the PHS. The Matérn kernels $M_{3/2}(r/\sigma)$ and $M_{5/2}(r/\sigma)$ of length σ , with $M_{3/2}(r) = (1+r)e^{-r}$ and $M_{5/2}$ given in (22) below, have Fourier transforms proportional to $(\sigma^{-2} + \|\mathbf{k}\|_2^2)^{-(3+d)/2}$ and $(\sigma^{-2} + \|\mathbf{k}\|_2^2)^{-(5+d)/2}$ in the wave vector $\mathbf{k}$. They are therefore, up to constants, the Green's functions of $(\sigma^{-2} - \Delta)^{(3+d)/2}$ and $(\sigma^{-2} - \Delta)^{(5+d)/2}$ in $\mathbb{R}^d$ [16, 35, 58, 67]. Letting $\sigma \rightarrow \infty$ removes the length and leaves $(-\Delta)^{(3+d)/2}$ and $(-\Delta)^{(5+d)/2}$, whose fundamental solutions are multiples of r^3 and r^5 [14, 65]. The two Matérn kernels are thus the counterparts, with a length, of the PHS r^3 and r^5 . For example,

$$M_{5/2}\left(\frac{r}{\sigma}\right) = 1 - \frac{r^2}{6\sigma^2} + \frac{r^4}{24\sigma^4} - \frac{r^5}{45\sigma^5} + O\left(\frac{r^6}{\sigma^6}\right). \quad (20)$$

The even terms are polynomials in $x - y$, and after the operator $-\Delta I + \nabla \nabla^T$ their contribution to an interpolant is a divergence-free polynomial of degree at most two, which the polynomial part of the local interpolant absorbs when $\varpi \geq 2$. Since rescaling the kernel does not change the interpolant, the local interpolant with $M_{5/2}(r/\sigma)$ thus tends to the one with r^5 as $\sigma \rightarrow \infty$. We use the PHS when no length scale is wanted, for smooth outputs whose accuracy is set by the polynomial degree, and Matérn kernels when lengths are wanted. A mixture of Matérn kernels can have many lengths. This motivates localizing the multiscale Matérn construction of our earlier work [53], which follows Owhadi [41]. On S_j , we sum five Matérn kernels with dyadic lengths and weights

$$\sigma_{j,\kappa} = \frac{\sigma_{j,0}}{2^\kappa}, \quad \sigma_{j,0} = c_\sigma \rho_j, \quad \vartheta_{j,\kappa} = \left(\frac{\sigma_{j,\kappa}}{\sigma_{j,0}}\right)^{8/3}, \quad \kappa = 1, \dots, 5, \quad (21)$$

where ρ_j is the stencil radius and c_σ a fixed factor. Each scale uses the scalar Matérn function

$$M_{5/2}(r) = \left(1 + r + \frac{r^2}{3}\right) e^{-r} \quad (22)$$

at the argument $r/\sigma_{j,\kappa}$. The weights are applied to the scalar kernels, before the divergence-free operator, whose two derivatives account for the 2 in the exponent $8/3 = 2 + 2/3$. The remaining $2/3$ is the Kolmogorov exponent of the velocity increments in three dimensions. The local multiscale kernel is

$$\Phi_j^M(x, y) = \frac{1}{c_j} (-\Delta_x I + \nabla_x \nabla_x^T) \sum_{\kappa=1}^5 3\vartheta_{j,\kappa} M_{5/2}\left(\frac{\|x - y\|_2}{\sigma_{j,\kappa}}\right), \quad c_j = (d-1) \sum_{\kappa=1}^5 \vartheta_{j,\kappa} \sigma_{j,\kappa}^{-2}, \quad (23)$$

normalized so that $\Phi_j^M(y, y) = I$. The normalization rescales the coefficients and does not change the interpolant. We replace Φ_{DF} by Φ_j^M in (15) and (18) and keep the polynomials, so each stencil still solves one system with r_j unknowns for the whole mixture. Unlike the cubic kernel, the mixture is smooth at its centers, so the reconstructed fields are classically divergence free everywhere on the stencil.

The weights build the prescribed increment statistics into the kernel [41]. Whether an interpolant on finite stencils reproduces those statistics also depends on the stencil extent and the node spacing, and we test this under refinement in Section 5.1 and for the turbulent benchmarks in Appendix A.5.

2.3. Periodic stencils

In a periodic domain, the nearest neighbors of a node near one face of the box can lie near the opposite face. Our stencils are therefore built from translated images of the nodes. If coordinate k has period Λ_k , a node y_l has images

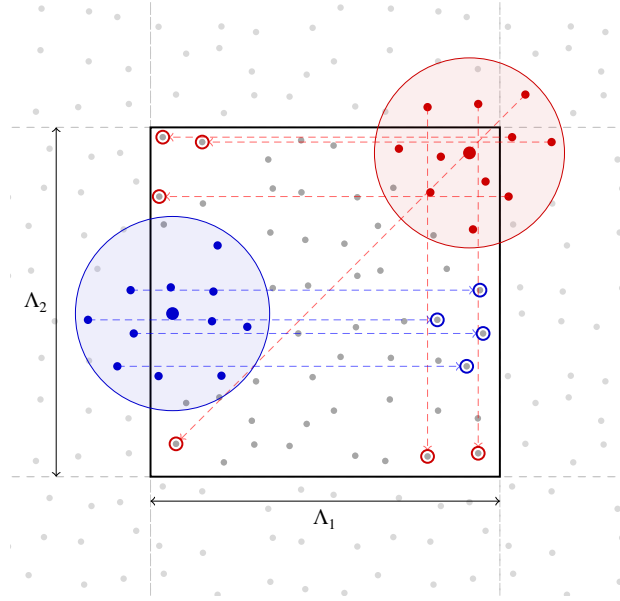


Figure 2: Periodic stencils in a box with periods Λ_1 and Λ_2 . Gray dots are nodes (dark) and their translated images (light). A stencil near a face (blue) and a stencil near a corner (red) are built from the nearest images given by (25). Dashed arrows link each image in a stencil to its node (circled), whose sampled velocity it carries.

$y_l + \sum_k v_k \Lambda_k \mathbf{e}_k$, with integers v_k over the periodic coordinates, which carry its index and its sampled velocity,

$$v_i \left(y_l + \sum_k v_k \Lambda_k \mathbf{e}_k \right) = v_i(y_l). \quad (24)$$

For a stencil center y_j , we take the displacements $\delta_l = y_l - y_j$ and shift every periodic component to its nearest representative,

$$\delta_{l,k} \leftarrow \delta_{l,k} - \Lambda_k \text{round}(\delta_{l,k}/\Lambda_k). \quad (25)$$

Since all periodic coordinates are shifted together, this also finds the right images near the edges and corners of the box, where a node's neighbors are shifted in two or three coordinates (Figure 2). The stencil consists of the n_s nearest of the points $y_j + \delta_l$, with one representative of each node, and the kernel and polynomials are evaluated in these translated coordinates. Translation does not change the divergence. Appendix A.4 gives the periods used for each benchmark.

2.4. Spacetime outputs

For outputs on $\Omega_v = \Omega \times \Gamma$, with Γ a time interval, PPKM [53] used a global product kernel

$$\Phi_{\text{ST}}((y, t), (y', t')) = \Phi_{\text{DF}}^{\text{Mat}}(y, y') \Phi_t(t, t'), \quad (26)$$

with a divergence-free Matérn kernel in space and a scalar Matérn kernel Φ_t in time, and dmN_t coefficients in one system. We instead keep each stencil and its spatial basis fixed over the sampled times $\mathcal{T} = \{t_s\}_{s=1}^{N_t}$ and interpolate in time. Let $\mathbf{b}_i^{j,s} \in \mathbb{R}^{r_j}$ be the coefficients of training field i on S_j at time t_s . Every snapshot solves (17) with the same matrix $\mathbf{A}_j$, and the target of the operator regression is the stacked vector

$$\mathbf{b}_{i,\mathcal{T}}^j = [(\mathbf{b}_i^{j,1})^\top, \dots, (\mathbf{b}_i^{j,N_t})^\top]^\top \in \mathbb{R}^{N_t r_j}. \quad (27)$$

Between the sampled times, we interpolate with the Lagrange polynomials

$$\omega_s(t) = \prod_{s' \neq s} \frac{t - t_{s'}}{t_s - t_{s'}}, \quad s = 1, \dots, N_t, \quad (28)$$

which need no temporal kernel parameter and no further linear system. With $\mathbf{A}_j^\star(y)$ the evaluation of the spatial basis on S_j as in (18), the spacetime output reconstruction of a prediction is

$$\chi_*^j(y, t) = \sum_{s=1}^{N_t} \omega_s(t) \mathbf{A}_j^\star(y) \mathbf{b}_*^{j,s}. \quad (29)$$

Its polynomial part is the tensor product of the retained spatial polynomials with polynomials of degree at most $N_t - 1$ in time. Since the weights ω_s do not depend on y , the spacetime output reconstruction is divergence free wherever the spatial one is. The local representation has $N_t \sum_j r_j$ coefficients, but each system keeps the size r_j . Another sampled time adds only a right-hand side.

3. Sparse operator regression

Having fixed the output representation, we now learn its coefficients as functions of the input. Each output coefficient is a function on the space of sampled inputs, and we approximate all of them by kernel regression with the same kernel. PPKM [53] used a kernel with global support, whose $N \times N$ kernel matrix is dense, so that its Cholesky factorization costs $O(N^3)$ operations and $O(N^2)$ storage for N training functions. We now construct a compactly supported kernel that is positive semidefinite on inputs of any dimension, so that the kernel matrix is sparse. The rest of the section defines the operator interpolant built from this kernel and its affine augmentation, derives their linear systems, factors them, and chooses the kernel parameters.

3.1. A compactly supported kernel in any input dimension

To make the kernel matrix sparse, the kernel must vanish for inputs that are far apart, and it must remain positive definite on the space of sampled inputs, whose dimension nd_a can be in the thousands. These two requirements conflict for radial kernels. By Schoenberg's theorem, a radial function that is positive definite in every dimension is a scale mixture of Gaussians and therefore never compactly supported [65]. Every compactly supported radial kernel [27, 64] is thus positive definite only up to some dimension $D_{\max}$. For the Wendland functions, the polynomial degree grows linearly with $D_{\max}$ at fixed smoothness [64], so a function valid in thousands of dimensions would have a degree in the thousands. Tapering [23], which multiplies a globally supported kernel by a compactly supported one, inherits the same restriction from its taper. Low numerical rank of the training inputs does not remove the restriction either, since a new input need not lie in their span.

Our approach is to separate the two requirements. We let a low-dimensional projection of the input determine the support, and use a compactly supported radial function only there, in a dimension in which it is positive definite. The remaining coordinates enter through a Matérn factor, which is positive definite in every dimension but not compactly supported. The product is positive semidefinite in any input dimension and has the support of its first factor. In this paper, we use the C^4 Wendland function

$$w(t) = (1 - t)_+^6 \left(1 + 6t + \frac{35}{3}t^2 \right), \quad (30)$$

which is positive definite in up to three dimensions [64], so we project onto at most three coordinates. Another member of the Wendland family would allow a projection of higher dimension.

Let $\bar{\mathbf{a}}$ be the sampled input after centering and scaling each coordinate by its training mean and standard deviation (coordinates with zero training variance are set to zero). Let the columns of V_z be the first n_z principal directions of the standardized training inputs, where n_z is the smaller of three and their numerical rank (Appendix B.1), and split

$$\mathbf{z}(\mathbf{a}) = V_z^\top \bar{\mathbf{a}}, \quad \zeta(\mathbf{a}) = (I - V_z V_z^\top) \bar{\mathbf{a}}. \quad (31)$$

For a support radius $\rho > 0$, a residual scale $\alpha_\perp > 0$, and a positive diagonal matrix $D_\parallel$, our kernel is

$$\lambda(\mathbf{a}, \mathbf{a}') = \eta(\mathbf{a})\eta(\mathbf{a}') w\left(\frac{\|D_\parallel(\mathbf{z}(\mathbf{a}) - \mathbf{z}(\mathbf{a}'))\|_2}{\rho}\right) M_{5/2}\left(\frac{\alpha_\perp \|\zeta(\mathbf{a}) - \zeta(\mathbf{a}')\|_2}{\delta}\right). \quad (32)$$

Here $M_{5/2}$ is the Matérn function (22), $\delta > 0$ is a residual length, and η is a gate. The Wendland factor sets the support, since two inputs interact only if their principal coordinates are closer than ρ in the metric $D_{\parallel}$. The Matérn factor separates inputs with the same principal coordinates. Note that the kernel is not radial in the full input.

The gate switches the kernel off for inputs whose residual $\zeta(\mathbf{a})$ is much larger than those of the training inputs. With $\zeta_i = \zeta(\mathbf{a}_i)$, we set

$$\delta = \max \left\{ 0.01 \max_i \|\bar{\mathbf{a}}_i\|_2, 2.2 \max_i \|\zeta_i\|_2, 10^{-8} \right\}, \quad (33)$$

and, with the polynomial $H(t) = 126t^5 - 420t^6 + 540t^7 - 315t^8 + 70t^9$, which increases from $H(0) = 0$ to $H(1) = 1$ since $H'(t) = 630t^4(1-t)^4$,

$$\eta(\mathbf{a}) = \hat{\eta}(\|\zeta(\mathbf{a})\|_2/\delta), \quad \hat{\eta}(r) = \begin{cases} 1, & 0 \leq r \leq 1/2, \\ H(2(1-r)), & 1/2 < r < 1, \\ 0, & r \geq 1. \end{cases} \quad (34)$$

The gate is C^4 and positive for $\|\zeta(\mathbf{a})\|_2 < \delta$. Since $\|\zeta_i\|_2 \leq \delta/2.2$, it equals one on every training input. A new input need not lie in the span of the principal directions V_z , and its gate decreases to zero as its residual grows.

Proposition 1 (Positive definiteness in any input dimension). *For any finite sampled-input dimension, the kernel (32) is positive semidefinite. Its Gram matrix is strictly positive definite on distinct standardized inputs with positive gates. Consequently, adding a positive multiple of the identity gives a positive-definite kernel matrix, including when sampled inputs coincide or gates vanish.*

Proof sketch. Both factors are positive definite functions composed with linear maps, and products and gate multiplications preserve positive semidefiniteness. Distinct inputs differ in their principal or their residual coordinates. Appendix C.2 gives the proof. $\square$

The proposition holds for any orthogonal projection, so it needs no estimate of an intrinsic dimension. The principal components only choose the directions along which the support is cut off.

Fixed and fitted shapes We compare two choices of the shape parameters. The fixed-shape kernel uses $D_{\parallel} = I$ and $\alpha_{\perp} = 1$. The fitted-shape kernel also fits $\alpha_{\perp}$ and $D_{\parallel} = \text{diag}(e^{v_1}, \dots, e^{v_{n_z}})$, with $\sum_{k=1}^{n_z} v_k = 0$ so that ρ alone sets the overall support and the v_k only redistribute it among the principal directions. Note that when $n_z = 1$ the constraint forces $D_{\parallel} = [1]$, and when the residual coordinates are negligible we also keep $\alpha_{\perp} = 1$. The fitted-shape kernel then has the shape of the fixed-shape one and differs from it only through its radius and regularization (Section 5.2.1 lists these cases).

3.2. The operator interpolant

We now use the kernel λ to approximate the map from a sampled input to the local coefficients of its output. Let $\mathbf{b}_i \in \mathbb{R}^{m_{\text{rhs}}}$ stack the local coefficients $\mathbf{b}_i^j$ of training field i on all stencils (and at all sampled times), where $m_{\text{rhs}} = \sum_j r_j$ (times N_t for spacetime outputs). Every one of these coefficients is a function of the same input, so we approximate all of them with the same scalar kernel, i.e., with the matrix-valued kernel $\lambda(\mathbf{a}, \mathbf{a}')I$. The operator regression $\tilde{f}$ of (6) is the operator interpolant

$$\tilde{f}(\mathbf{a}) = \sum_{i=1}^N \lambda(\mathbf{a}, \mathbf{a}_i) \mathbf{c}_i, \quad (35)$$

with coefficients $\mathbf{c}_i \in \mathbb{R}^{m_{\text{rhs}}}$ determined by the regularized interpolation conditions

$$\tilde{f}(\mathbf{a}_i) + \theta \mathbf{c}_i = \mathbf{b}_i, \quad i = 1, \dots, N, \quad (36)$$

where $\theta > 0$ is a regularization parameter. In the limit $\theta \rightarrow 0$, $\tilde{f}$ interpolates the training coefficients, and for $\theta > 0$ it is their kernel ridge regression. Since χ is linear, the learned operator (6) is

$$\tilde{G}(a) = \sum_{i=1}^N \lambda(\mathbf{a}, \mathbf{a}_i) \chi(\mathbf{c}_i), \quad (37)$$

a combination of N fixed fields with weights that depend on the input. Every one of these fields is a local divergence-free output reconstruction, so $\tilde{\mathcal{G}}(a)$ is divergence free on each stencil for every input, regardless of the accuracy of the operator regression.

3.3. Affine augmentation

Operators often depend nearly linearly on part of their input. A compactly supported kernel regression represents such a trend poorly, since it returns to zero away from the training inputs. As in polynomially augmented interpolation [4], we augment the operator regression with affine functions of up to 15 principal coordinates of the input (more than the at most three that set the support), which it reproduces exactly, and let the kernel represent the remainder. These polynomials in the input are unrelated to the divergence-free polynomials in space of Section 2, which are part of the output representation in every model.

Let the columns of V_p be up to 15 principal directions of the training inputs, let ς_k be the training standard deviation along direction k , and set

$$p(\mathbf{a}) = \left(1, \frac{(V_p^T \bar{\mathbf{a}})_1}{\varsigma_1}, \dots, \frac{(V_p^T \bar{\mathbf{a}})_{n_{\text{aff}}-1}}{\varsigma_{n_{\text{aff}}-1}} \right)^T, \quad (38)$$

with $n_{\text{aff}} - 1 \leq 15$ limited by the rank of the training inputs. The augmented operator interpolant is

$$\tilde{f}(\mathbf{a}) = \sum_{i=1}^N \lambda(\mathbf{a}, \mathbf{a}_i) \mathbf{c}_i + \boldsymbol{\beta}^T p(\mathbf{a}), \quad (39)$$

with kernel coefficients $\mathbf{c}_i \in \mathbb{R}^{m_{\text{rhs}}}$ and affine coefficients $\boldsymbol{\beta} \in \mathbb{R}^{n_{\text{aff}} \times m_{\text{rhs}}}$. Its coefficients are determined by the regularized interpolation conditions of (36), now for the augmented $\tilde{f}$, together with moment conditions on the kernel coefficients,

$$\tilde{f}(\mathbf{a}_i) + \theta \mathbf{c}_i = \mathbf{b}_i, \quad i = 1, \dots, N, \quad \sum_{i=1}^N p(\mathbf{a}_i) \mathbf{c}_i^T = 0. \quad (40)$$

The moment conditions make (39) reproduce every affine function of the principal coordinates exactly, for any θ . If $\mathbf{b}_i = T^T p(\mathbf{a}_i)$ for some matrix T , then $\mathbf{c}_i = 0$ and $\boldsymbol{\beta} = T$ satisfy (40). As in (37), the augmented learned operator is

$$\tilde{\mathcal{G}}(a) = \sum_{i=1}^N \lambda(\mathbf{a}, \mathbf{a}_i) \chi(\mathbf{c}_i) + \sum_{k=1}^{n_{\text{aff}}} p_k(\mathbf{a}) \chi(\boldsymbol{\beta}_{k,:}), \quad (41)$$

a combination of $N + n_{\text{aff}}$ fixed divergence-free fields. Without augmentation, $n_{\text{aff}} = 0$, the affine terms and the moment conditions disappear, and (39)–(41) reduce to (35)–(37).

Both interpolants have a variational characterization. The function $\tilde{f} = f_{\mathcal{H}} + \boldsymbol{\beta}^T p$ minimizes

$$\sum_{i=1}^N \left\| f_{\mathcal{H}}(\mathbf{a}_i) + \boldsymbol{\beta}^T p(\mathbf{a}_i) - \mathbf{b}_i \right\|_2^2 + \theta \|f_{\mathcal{H}}\|_{\mathcal{H}}^2 \quad (42)$$

over the affine coefficients $\boldsymbol{\beta}$ and the functions $f_{\mathcal{H}}$ in the reproducing kernel Hilbert space $\mathcal{H}$ of $\lambda(\mathbf{a}, \mathbf{a}')I$, since by the representer theorem the minimizing $f_{\mathcal{H}}$ has the form $\sum_i \lambda(\cdot, \mathbf{a}_i) \mathbf{c}_i$, and (40) makes the gradient of the convex objective vanish (Appendix C.3). Without augmentation, $\boldsymbol{\beta}$ is absent and (42) is the objective of kernel ridge regression.

3.4. Linear systems

We now write the conditions (36) and (40) as linear systems. For training inputs $U = \{\mathbf{a}_i\}_{i=1}^N$ and N_* query inputs $U_* = \{\mathbf{a}_*^{(q)}\}_{q=1}^{N_*}$, we define the kernel matrices

$$(K_{UU})_{ii'} = \lambda(\mathbf{a}_i, \mathbf{a}_{i'}), \quad (K_{U_*U})_{qi} = \lambda(\mathbf{a}_*^{(q)}, \mathbf{a}_i), \quad K_{UU}^\theta = K_{UU} + \theta I, \quad (43)$$

with θ the regularization of (36), and the affine matrices P_U and P_{U^*} with rows $(P_U)_{i,:} = p(\mathbf{a}_i)^\top$ and $(P_{U^*})_{q,:} = p(\mathbf{a}_*^{(q)})^\top$. We keep every nonzero entry of K_{UU} , with no neighbors discarded by an approximate search. Its density $\text{nnz}(K_{UU})/N^2$, where nnz counts the nonzero entries, is thus set by ρ and the data, and the storage of its Cholesky factor also depends on the fill.

Since the kernel $\lambda(\mathbf{a}, \mathbf{a}')I$ does not couple different output coefficients, the conditions decouple into one system per stencil. On stencil j , let $B_j \in \mathbb{R}^{N \times r_j}$ hold the $\mathbf{b}_i^j$ as rows, and let $C_j \in \mathbb{R}^{N \times r_j}$ and $\beta_j \in \mathbb{R}^{n_{\text{aff}} \times r_j}$ hold the corresponding entries of the $\mathbf{c}_i$ and the columns of β . Evaluating (39) at the training inputs turns (40) into the first equation of

$$\begin{bmatrix} K_{UU}^\theta & P_U \\ P_U^\top & 0 \end{bmatrix} \begin{bmatrix} C_j \\ \beta_j \end{bmatrix} = \begin{bmatrix} B_j \\ 0 \end{bmatrix}, \quad B_{*,j} = K_{U^*U} C_j + P_{U^*} \beta_j, \quad (44)$$

and evaluating it at the query inputs gives the second, where row q of $B_{*,j}$ holds the coefficients that $\tilde{f}$ predicts on stencil j for query input q . The block $P_U^\top C_j = 0$ is the moment condition. Without augmentation, the same steps applied to (35) and (36) give

$$K_{UU}^\theta C_j = B_j, \quad B_{*,j} = K_{U^*U} C_j. \quad (45)$$

3.5. Sparse Cholesky and whitened QR

The system (44) is an indefinite saddle-point system. Forming the Schur complement $P_U^\top (K_{UU}^\theta)^{-1} P_U$ squares the condition number of the whitened polynomial columns, as in the normal equations of least squares. We instead use the Cholesky–QR elimination of our unified kernel and polynomial interpolation [4]. Since K_{UU}^θ is sparse and positive definite, we factor it by sparse Cholesky, whiten the polynomial columns, and factor them by thin QR. Suppressing the fill-reducing permutation, we obtain

$$K_{UU}^\theta = L_K L_K^\top, \quad L_K^{-1} P_U = Q_P R_P, \quad \mathcal{E}_U = R_P^{-1} Q_P^\top L_K^{-1}. \quad (46)$$

Here Q_P has orthonormal columns and R_P is a nonsingular $n_{\text{aff}} \times n_{\text{aff}}$ triangular matrix, and we apply $\mathcal{E}_U$ by triangular solves without forming L_K^{-1} . Block elimination of (44) then gives

$$\beta_j = \mathcal{E}_U B_j, \quad K_{UU}^\theta C_j = B_j - P_U \beta_j. \quad (47)$$

Without augmentation, Q_P , R_P , and $\mathcal{E}_U$ are empty and the correction disappears. The same factors serve the parameter selection below and the prediction of Section 4.1, which never forms the C_j .

The size of the Cholesky factor, and the speed of its triangular solves, depend on the ordering of the training inputs. Since two inputs interact only if their principal coordinates are close, the sparsity pattern of K_{UU} resembles that of a discretization on a mesh in at most three dimensions, and we order it by nested dissection. For inputs of numerical rank one ($n_z = 1$), sorting the inputs by z even makes K_{UU} a band matrix (the matrix of an interval graph), whose Cholesky factor has no entries outside the band. However, the elimination of a band matrix is a single chain, which leaves the triangular solves of a prediction little parallelism on the GPU, while nested dissection gives a balanced elimination tree. Thus, nested dissection is our default, and the band ordering is preferable when memory is more limited than prediction time (Section 5.2.5). The ordering changes neither the operator regression nor its predictions beyond rounding.

3.6. Choosing the kernel parameters

The support radius ρ , the regularization θ , and, for the fitted-shape kernel, the shape parameters remain to be chosen. The support radius is the most important of these, since it trades accuracy against sparsity. We use two criteria in turn. For the fitted-shape kernel, a restricted likelihood criterion fits the shape and the regularization from the training data alone. The error on validation fields then selects the radius and refines the regularization.

Let $F \in \mathbb{R}^{N \times n_v}$ hold all sampled training outputs, with $n_v = dm$ values per field, or dmN_t for spacetime outputs, after the amplitude normalization (72) where it applies, let Q_0 be an orthonormal basis of the range of P_U , $E_0 = (I - Q_0 Q_0^\top) F$, and $E = \sqrt{N} E_0 / \|E_0\|_F$, and let

$$\Pi_U = (K_{UU}^\theta)^{-1} - (K_{UU}^\theta)^{-1} P_U [P_U^\top (K_{UU}^\theta)^{-1} P_U]^{-1} P_U^\top (K_{UU}^\theta)^{-1}, \quad (48)$$

which we apply by sparse solves and never store. With $n_{\text{res}} = N - n_{\text{aff}}$, we minimize

$$\mathcal{J}_{c_d} = \frac{1}{2} \left[\frac{\log \det K_{UU}^\theta + 2 \log |\det R_P|}{n_{\text{res}}} + \log \left(\frac{\text{tr}(E^\top \Pi_U E)}{n_{\text{res}}} \right) \right] + c_d \tilde{d}, \quad (49)$$

$$\tilde{d} = \frac{1}{N(N-1)} \sum_{i \neq i'} \text{sigmoid} \left(\frac{1 - \|D_\parallel(z_i - z_{i'})\|_2 / \rho}{0.03} \right). \quad (50)$$

The bracket can be interpreted in two ways. In kernel terms, $\text{tr}(E^\top \Pi_U E)$ is the squared norm, in the native space of the regularized kernel, of the augmented interpolants of the training outputs after their affine part is removed, and the log-determinants penalize kernels whose matrices have a large determinant, which are the more flexible ones. The bracket thus favors kernels that represent the outputs with a small norm, while penalizing overly flexible kernels. In probabilistic terms, if the outputs are modeled as samples of a Gaussian process with covariance λ , independent noise of variance θ , and an affine mean with unknown coefficients, the bracket is, up to constants, the negative log restricted likelihood with the variance profiled out (with $\det R_P = 1$ without augmentation). We borrow the criterion from this setting, but use it only to select kernel parameters, not for uncertainty quantification.

The term $\tilde{d}$, with $z_i = z(\mathbf{a}_i)$ and the logistic function $\text{sigmoid}(x) = 1/(1 + e^{-x})$, is a smooth count of the interacting pairs, and its weight $c_d \in \{0, 1\}$ trades accuracy against sparsity. We fit one candidate with each weight. All terms of $\mathcal{J}_{c_d}$ come from the sparse factors (Appendix B.3).

The criterion measures how well the kernel represents the training outputs, but our goal is the error of the predicted velocity for new inputs. Thus, we discard the radius that minimizes the likelihood criterion and choose the radius by validation alone. We first evaluate the five radii that give the densities 0.10, 0.25, 0.50, 0.75, and 0.90, with the regularization from the likelihood. We then take gradient steps in the radius and the regularization on the mean relative velocity error on validation fields, from four starts (half and a quarter of the largest of these radii, each with $\theta = 10^{-6}$ and 10^{-3}), while the fitted shape stays unchanged. Among all candidates with density at most 0.90, we keep the one with the smallest validation error. The fixed-shape kernel goes through the same search without the likelihood step, with $\theta = 10^{-6}$ at the five radii. All choices are made before the test fields are used (Appendix B.2).

We thus compare four models, the fixed-shape and the fitted-shape kernel, each with and without affine augmentation. A comparison at the same augmentation isolates the effect of fitting the shape. Since all parameters are refit when the augmentation changes, a comparison with and without it measures the whole procedure.

4. Operator Lagrange prediction

We now combine the operator regression of Section 3 with the output reconstruction of Section 2. The direct way to predict is to solve (44) for the coefficients C_j and β_j of every stencil. This needs one operator right-hand side for every spatial coefficient, and the local coefficients of every training field. However, an operator regression with the same kernel for all outputs is a weighted combination of the training outputs, and the weights depend only on the new input. It suffices to solve once per new input for these weights and apply them to the whole output at once. Since the output reconstruction is linear, we can apply them even before the output reconstruction, to the sampled training fields.

4.1. Operator Lagrange weights

We first define the weights as functions on the input space. For an input $\mathbf{a}$, let $k_U(\mathbf{a}) = [\lambda(\mathbf{a}_i, \mathbf{a})]_{i=1}^N$. We define its operator Lagrange weights $\ell(\mathbf{a}) \in \mathbb{R}^N$ by

$$\begin{bmatrix} K_{UU}^\theta & P_U \\ P_U^\top & 0 \end{bmatrix} \begin{bmatrix} \ell(\mathbf{a}) \\ \mu(\mathbf{a}) \end{bmatrix} = \begin{bmatrix} k_U(\mathbf{a}) \\ p(\mathbf{a}) \end{bmatrix}, \quad (51)$$

where the multiplier $\mu(\mathbf{a})$ enforces the reproduction of affine functions. Block elimination with the factors (46) gives

$$\ell(\mathbf{a}) = (I - \mathcal{E}_U^\top P_U^\top)(K_{UU}^\theta)^{-1} k_U(\mathbf{a}) + \mathcal{E}_U^\top p(\mathbf{a}), \quad (52)$$

so every weight ℓ_i is a combination of the kernel functions $\lambda(\cdot, \mathbf{a}_i)$ and the affine functions, of the same form as (39). Since the matrix of (51) is the symmetric matrix of (44), the operator interpolant (39) and the learned operator (41)

take the Lagrange form

$$\tilde{f}(\mathbf{a}) = \sum_{i=1}^N \ell_i(\mathbf{a}) \mathbf{b}_i, \quad \tilde{G}(a) = \sum_{i=1}^N \ell_i(\mathbf{a}) \chi(\mathbf{b}_i), \quad (53)$$

a combination of the reconstructed training fields with weights that depend only on the input (Appendix C.4). Without augmentation, $\ell(\mathbf{a}) = (K_{UU}^\theta)^{-1} k_U(\mathbf{a})$, and (53) is the Lagrange form of (35) and (37). The weights are regularized Lagrange (cardinal) functions of the training inputs [15, 46, 65]. For $\theta = 0$, they are cardinal at the training inputs. For $\theta > 0$, they are not, but they still reproduce the affine functions exactly.

Proposition 2 (Reproduction and preservation under shared weights). *Assume K_{UU}^θ is positive definite and P_U has full column rank. Then (52) is the unique weight vector in (51), and $P_U^\top \ell(\mathbf{a}) = p(\mathbf{a})$. In particular, the augmented model reproduces constants and the scaled principal coordinates in $p(\mathbf{a})$. Let χ be any common linear output reconstruction of the training coefficient vectors, and let $\mathcal{D}$ be a linear constraint with $\mathcal{D}\chi(\mathbf{b}_i) = 0$ for every training field. Then*

$$\chi\left(\sum_i \ell_i(\mathbf{a}) \mathbf{b}_i\right) = \sum_i \ell_i(\mathbf{a}) \chi(\mathbf{b}_i), \quad \mathcal{D}\chi\left(\sum_i \ell_i(\mathbf{a}) \mathbf{b}_i\right) = 0. \quad (54)$$

Proof sketch. The factors (46) give $\mathcal{E}_U P_U = I$, from which reproduction follows by substitution, and uniqueness follows from the block structure. The identities (54) are linearity. Appendix C.4 gives the details. $\square$

With $\mathcal{D}$ the divergence on a stencil, the proposition shows that every prediction is divergence free on each stencil, for both kernels and with or without augmentation. Since the temporal weights (28) do not depend on the input, the operator weights also commute with interpolation in time.

We now turn to the discrete form. For a new input $\mathbf{a}_*$, restricting the first identity of (53) to stencil j gives the predicted coefficients

$$\mathbf{b}_*^j = B_j^\top \ell(\mathbf{a}_*), \quad (55)$$

and, with the weights of N_* new inputs as the rows of $L \in \mathbb{R}^{N_* \times N}$, the second equation of (44) becomes

$$B_{*,j} = L B_j. \quad (56)$$

Figure 1 illustrates (55). One sparse solve gives the weights of a new input, and these weights produce the coefficients on every stencil. The number of operator solves now equals the number of new inputs and no longer depends on the size of the output representation. To compute (52), we first solve $K_{UU}^\theta \ell_0(\mathbf{a}) = k_U(\mathbf{a})$ and then correct for the affine augmentation,

$$\ell(\mathbf{a}) = \ell_0(\mathbf{a}) + \mathcal{E}_U^\top [p(\mathbf{a}) - P_U^\top \ell_0(\mathbf{a})]. \quad (57)$$

Without augmentation, $L = K_{U_*U} (K_{UU}^\theta)^{-1}$, so that

$$K_{UU}^\theta L^\top = K_{U_*U}^\top. \quad (58)$$

4.2. Applying the weights before the output reconstruction

With $\mathbf{b}_i$ as in Section 3.2, let $\mathcal{R}$ be the linear map from a sampled output $\mathbf{v}_i$ to its local output reconstruction $\chi(\mathbf{b}_i)$. Since the same stencils, polynomial bases, and temporal weights serve every training field, linearity gives

$$\begin{aligned} \tilde{G}(a) &= \chi\left(\sum_{i=1}^N \ell_i(\mathbf{a}) \mathbf{b}_i\right) = \sum_{i=1}^N \ell_i(\mathbf{a}) \chi(\mathbf{b}_i) \\ &= \mathcal{R}\left(\sum_{i=1}^N \ell_i(\mathbf{a}) \mathbf{v}_i\right). \end{aligned} \quad (59)$$

Our GPU implementation uses the last form. It combines the sampled training fields with the weights and then applies $\mathcal{R}$ once per new input. At setup, adjoint solves with the local matrices $\mathbf{A}_j$ turn $\mathcal{R}$ into compact evaluation weights on each stencil. Prediction then stores the sampled training fields, the evaluation weights, and the stencil indices, but none of the N sets of local coefficients. For the Taylor–Green benchmarks, whose responses we divide by the input amplitude before fitting (Section 5.2.1), the same identity holds for the normalized responses.

4.3. Rounding errors of direct and Lagrange evaluation

The forms in (59) agree in exact arithmetic, but they round differently. The difference is important when a moderate velocity is the result of cancellation among large spatial coefficients, which can happen for a global interpolant. A solve error then enters the velocity amplified by the output reconstruction. We now quantify this for the model without augmentation and for evaluation through the coefficients, which is the setting of our comparison in Section 5.3. The analysis does not cover the affine correction (57) or the order of operations of Section 4.2.

Assume K_{UU}^θ is symmetric and nonsingular. We fix one stencil and drop its index, writing $r = r_j$, $B = B_j \in \mathbb{R}^{N \times r}$, and $C = C_j \in \mathbb{R}^{N \times r}$, so that each row of B holds the spatial coefficients of one training field. Let $\mathbf{A}^\star \in \mathbb{R}^{n_{\text{eval}} \times r}$ stack the evaluation matrices (18) of the n_{eval} velocity values reconstructed from this stencil, so that $B(\mathbf{A}^\star)^\top$ holds the reconstructed training velocities. The direct coefficients C and the Lagrange weights L satisfy

$$K_{UU}^\theta C = B, \quad L K_{UU}^\theta = K_{U_\star U}, \quad (60)$$

and both give the prediction $V_\star = K_{U_\star U} C (\mathbf{A}^\star)^\top = L B (\mathbf{A}^\star)^\top$. The same notation describes a global output reconstruction, with a single stencil.

Theorem 3 (Solve errors and multiplication rounding). *For computed approximations $\hat{C}$ and $\hat{L}$, define the solve residuals by evaluating their defining equations in exact arithmetic,*

$$R_{\text{dir}} = B - K_{UU}^\theta \hat{C}, \quad R_{\text{Lag}} = K_{U_\star U} - \hat{L} K_{UU}^\theta. \quad (61)$$

Before subsequent multiplication rounding, the prediction errors satisfy

$$K_{U_\star U} \hat{C} (\mathbf{A}^\star)^\top - V_\star = -L R_{\text{dir}} (\mathbf{A}^\star)^\top, \quad (62)$$

$$\hat{L} B (\mathbf{A}^\star)^\top - V_\star = -R_{\text{Lag}} (K_{UU}^\theta)^{-1} B (\mathbf{A}^\star)^\top. \quad (63)$$

Define, with $\text{fl}(\cdot)$ denoting evaluation in floating point, $\hat{V}_{\star, \text{dir}} = \text{fl}(\text{fl}(K_{U_\star U} \hat{C}) (\mathbf{A}^\star)^\top)$ and $\hat{V}_{\star, \text{Lag}} = \text{fl}(\text{fl}(\hat{L} B) (\mathbf{A}^\star)^\top)$.

Suppose the coefficient products and the output reconstruction use standard floating-point dot products, possibly split into d component blocks followed by summation.

Assume no overflow or underflow, and set $\gamma_k = k u_{\text{mach}} / (1 - k u_{\text{mach}})$, where u_{mach} is unit roundoff and $\max\{N, r + d\} u_{\text{mach}} < 1$. With $\gamma_{\text{mult}} = \gamma_N + \gamma_{r+d} + \gamma_N \gamma_{r+d}$, we have

$$\|\hat{V}_{\star, \text{dir}} - V_\star\|_F \leq \|L\|_F \|R_{\text{dir}}\|_F \|(\mathbf{A}^\star)^\top\|_F + \gamma_{\text{mult}} \|K_{U_\star U}\|_F \|\hat{C}\|_F \|(\mathbf{A}^\star)^\top\|_F, \quad (64)$$

$$\|\hat{V}_{\star, \text{Lag}} - V_\star\|_F \leq \|R_{\text{Lag}}\|_F \|(K_{UU}^\theta)^{-1} B (\mathbf{A}^\star)^\top\|_F + \gamma_{\text{mult}} \|\hat{L}\|_F \|B\|_F \|(\mathbf{A}^\star)^\top\|_F. \quad (65)$$

Here absolute values are taken entrywise. The residual identities require symmetry and nonsingularity, but not positive definiteness.

Proof sketch. Substituting the residuals into the two predictions gives (62)–(63). The standard bound for floating-point dot products then bounds the multiplications. Appendix C.5 gives the proof. $\square$

The bounds become clearer with the amplification factor

$$\alpha_{\text{spatial}} = \frac{\|B\|_F \|(\mathbf{A}^\star)^\top\|_F}{\|B(\mathbf{A}^\star)^\top\|_F} \geq 1 \quad (66)$$

which measures how much the coefficients cancel in the output reconstruction of the training velocities. If $\|R_{\text{dir}}\|_F \leq \eta_{\text{dir}} \|B\|_F$ and $\|R_{\text{Lag}}\|_F \leq \eta_{\text{Lag}} \|K_{U_\star U}\|_F$, the solve errors satisfy

$$\|K_{U_\star U} \hat{C} (\mathbf{A}^\star)^\top - V_\star\|_F \leq \eta_{\text{dir}} \|L\|_F \alpha_{\text{spatial}} \|B(\mathbf{A}^\star)^\top\|_F, \quad (67a)$$

$$\|\hat{L}B(A^\star)^\top - V_*\|_F \leq \eta_{\text{Lag}} \|K_{U_*U}\|_F \|(K_{UU}^\theta)^{-1}\|_F \|B(A^\star)^\top\|_F. \quad (67b)$$

In direct evaluation, the relative solve error η_{dir} is amplified by α_{spatial} . In Lagrange evaluation, the coefficients have already been combined into velocities, and α_{spatial} does not appear. A global interpolant with a smooth kernel can have very large α_{spatial} , since its coefficients are large and cancel, while a local interpolant on a small stencil has moderate coefficients. The rounding of the multiplications, however, contains the entrywise product with $|(A^\star)^\top|$ in both bounds (of $|\hat{C}|$ in direct and of $|B|$ in Lagrange evaluation), so cancellation in the output reconstruction also enters the rounding, and the theorem does not make Lagrange evaluation more accurate in general. We test these predictions in Section 5.3.

4.4. GPU implementation

Algorithm 1 summarizes our implementation. Every floating-point computation runs on the GPU in double precision, from the normalization of the inputs and the stencil selection to the factorizations and the output reconstruction, using the routines of KernelPack-JAX [8]. Only the integer bookkeeping of the cell-based stencil search for large non-periodic node sets (Section 5.1.5) runs on the host. The kernel is assembled pair by pair over its support, in passes of bounded size, so that no dense kernel or distance matrix is ever formed.

However, JAX has no sparse direct solver that keeps a factorization between calls. Our implementation therefore factors K_{UU}^θ with NVIDIA's cuDSS through the primitive `kernelpack.sparse.cudss` of KernelPack-JAX, which exposes cuDSS to XLA, the compiler behind JAX. We use cuDSS in its symmetric positive-definite mode, with its nested-dissection ordering (Section 3.5). The factors stay on the GPU, and the right-hand sides of a batch of new inputs are solved as the columns of one call. A change of the support (i.e., of ρ) requires a new ordering and symbolic analysis, while a change of the values only requires a numerical refactorization. Appendix B.4 describes the batching and the checks of the timed runs.

Algorithm 1 Local and sparse kernel operator learning on the GPU.

Require: training and validation pairs, new inputs, output points, spatial kernel and polynomial degree

Output reconstruction:

- 1: select the stencils, with periodic images where needed (Section 2.3)
- 2: build the divergence-free polynomials and the local systems, and prepare $\mathcal{R}$ (Section 2.1)
- 3: for spacetime outputs, add the temporal weights (28)

Operator regression:

- 4: standardize the training inputs and compute their principal directions
- 5: for each choice of augmentation, keep the fixed shape or fit it by (49)
- 6: select the radius and regularization on the validation fields
- 7: assemble K_{UU}^θ in sparse form and factor it with cuDSS
- 8: with augmentation, form the whitened QR correction (46)

Prediction:

- 9: solve for the Lagrange weights (57) of a batch of new inputs
 - 10: combine the sampled training fields and apply $\mathcal{R}$ by (59)
 - 11: check the solve residuals and the reproduction of affine functions
-

4.5. Work and storage

Table 1 lists the work and storage of each step. Let $\mathcal{N}_K = \text{nnz}(K_{UU})$ and $\mathcal{N}_{\text{fac}} = \text{nnz}(L_K)$, where L_K is the Cholesky factor of K_{UU}^θ in (46). With $\mathcal{L}_i$ the set of rows in which column i of L_K is nonzero, so that $\sum_{i=1}^N |\mathcal{L}_i| = \mathcal{N}_{\text{fac}}$, the sparse factorization takes, up to a constant factor,

$$F_{\text{chol}} = \sum_{i=1}^N |\mathcal{L}_i|^2, \quad \frac{\mathcal{N}_{\text{fac}}^2}{N} \leq F_{\text{chol}} \leq N \mathcal{N}_{\text{fac}}, \quad (68)$$

floating-point operations [9], where the lower bound follows from the Cauchy–Schwarz inequality and the upper bound from $|\mathcal{L}_i| \leq N$. For a dense factor, $|\mathcal{L}_i| = N - i + 1$ and $F_{\text{chol}} \approx N^3/3$. Let b_q be the number of new inputs in a batch, let m_{rhs} be the number of spatial coefficients, and let n_v be the number of sampled output values per field ($n_v = dm$,

Table 1

Work and storage. Here $\mathcal{N}_K$ and $\mathcal{N}_{\text{fac}}$ are the numbers of nonzeros of the kernel matrix and of its Cholesky factor, F_{chol} the work (68) of the factorization, b_q the number of new inputs in a batch, n_{aff} the number of affine columns, n_v the number of sampled output values per field, m^* the number of query points, r_j the size of the local system on stencil j , and $m_{\text{rhs}} = \sum_j r_j$. The last two rows belong to direct evaluation through coefficients, which our predictions avoid. For spacetime outputs, the counts of these two rows are multiplied by N_t .

Step	Work	Storage
Kernel factorization	F_{chol}	$\mathcal{N}_K, \mathcal{N}_{\text{fac}}$
Affine correction (n_{aff} columns)	solves with n_{aff} right-hand sides, $O(Nn_{\text{aff}}^2)$ QR	$O(Nn_{\text{aff}})$
Lagrange weights, b_q queries	$O(b_q \mathcal{N}_{\text{fac}}) + O(b_q N n_{\text{aff}})$	$b_q N$ (dense)
Combining training fields	$O(b_q N n_v)$	$N n_v$
Local systems (setup)	$O(r_j^3)$ per stencil	
Local evaluation, m^* points	$O(b_q m^* d^2 n_s)$ per snapshot	$O(m^* d^2 n_s)$ weights, $O(m^* n_s)$ indices
Training coefficients	$O(N \sum_j r_j^2)$	$N \sum_j r_j$
Applying weights to coefficients	$O(b_q N m_{\text{rhs}})$	

or dmN_t for spacetime outputs). We count the work of a triangular solve as proportional to the number of nonzeros of the factor.

Two observations follow from the table. First, the local output reconstruction costs $O(r_j^3)$ per stencil, with $r_j = dn_s + n_{p,j}$ fixed by the stencil size and the polynomial degree, so its cost grows linearly with the number of output points instead of cubically. Second, the cost of the operator regression is governed by the factor of K_{UU}^θ , whose size depends on the fill as well as on the density of K_{UU} [9]. At a fixed density, the kernel matrix and its factor still grow as N^2 . For this reason, we report density, factor storage, and prediction time separately in Section 5.

Note that the Lagrange weights are dense even when K_{UU} is sparse. Direct evaluation instead solves m_{rhs} right-hand sides once and can reuse its coefficients for any number of new inputs, so the cheaper route depends on how many predictions are needed. Appendix B.3 gives the storage accounting of our experiments.

5. Numerical experiments

We now test our method in three steps. We first test the local output reconstruction alone, with exact fields in place of predictions, for convergence and for the three properties it preserves (Section 5.1). We then compare four operator models on ten fluid benchmarks as the training set grows, in accuracy, sparsity, and storage (Section 5.2). Finally, we compare direct and Lagrange evaluation in floating point and measure the cost of prediction (Section 5.3).

5.1. Local output reconstruction

Exact velocity fields in place of predictions separate the error of the output reconstruction from that of the operator regression. We use these tests to answer three questions. Does it converge at the order set by its polynomial degree on scattered nodes? Are its divergence and its periodic mismatch at rounding level? And does the multiscale Matérn kernel reproduce prescribed turbulence statistics as the nodes are refined?

5.1.1. Convergence on scattered nodes

We interpolate manufactured divergence-free fields in the smooth domains $\Omega_d = \{F_d > 0\}$, $d = 2, 3$, bounded by a quartic curve and a quartic surface. Each field is the curl of a trigonometric stream function ($d = 2$) or vector potential ($d = 3$) multiplied by F_d^2 , so it vanishes on the boundary (Appendix D.2 gives F_d and the potentials). We use divergence-free polynomials of degree $\varpi = 1, 3$, and 5 for target orders 2, 4, and 6. Each stencil has $n_s = 2\lceil n_p/d \rceil + 1$ nodes, where n_p is the dimension of the polynomial space, and both the PHS (r^3 for degrees 1 and 3, and r^5 for degree 5) and the multiscale Matérn kernel ($c_\sigma = 16$, which sets the Matérn lengths relative to the stencil radius in (21)) use the same stencils and polynomials. The node sets are four refinements generated by the node generator of KernelPack-JAX [8], with 172 to 1907 nodes in 2D and 197 to 2570 in 3D, and the stencils are boundary aware (Section 2.1).

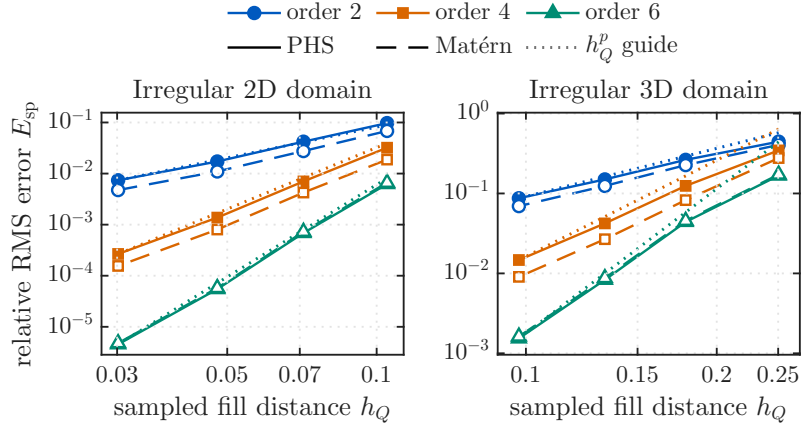


Figure 3: Convergence of the local output reconstruction on scattered nodes in the domains Ω_2 (left) and Ω_3 (right), with PHS and multiscale Matérn kernels and divergence-free polynomials. Relative error E_{sp} at random evaluation points that are not nodes, against the sampled fill distance h_Q of (69). Dotted lines: h_Q^2 , h_Q^4 , h_Q^6 .

Let v_h be the output reconstruction of the exact field v from the nodes Y of a node set with spacing h . We measure its relative error on a fixed set Q of uniformly random interior evaluation points (4096 in 2D and 32768 in 3D), against the fill distance sampled on the same set,

$$E_{\text{sp}} = \left(\frac{\sum_{y \in Q} \|v_h(y) - v(y)\|_2^2}{\sum_{y \in Q} \|v(y)\|_2^2} \right)^{1/2}, \quad h_Q = \max_{y \in Q} \min_{y' \in Y} \|y - y'\|_2, \quad (69)$$

where each evaluation point uses the stencil of its nearest node.

Figure 3 shows the results. In 2D, both kernels converge at close to the target orders, with least-squares rates of 2.1, 3.9, and 5.8. In 3D, the rates over all four node sets are lower, 1.7–1.9, 3.4–3.7, and 5.0, but the sixth-order rate rises to 5.5 over the three finest node sets. The 3D deficit is likely due to the coarsest node sets, which are not yet in the asymptotic range. The two kernels give nearly the same errors on smooth fields, so the multiscale kernel does not reduce the accuracy on smooth fields.

5.1.2. Divergence and periodicity

Since every basis function is divergence free, the divergence of the output reconstruction should vanish up to rounding, however large its approximation error. Differentiating each local expansion on its own stencil, we find a normalized root-mean-square (RMS) divergence at rounding level for both kernels, all degrees, and all node sets (Appendix D.2). We test periodic stencils on scattered nodes in $[0, 2\pi]^d$, comparing the reconstructed velocities and their Jacobians at periodically equivalent points on faces, edges, and corners. The differences are again at rounding level for both kernels and all degrees, while the approximation error varies with degree and resolution as expected.

5.1.3. Kolmogorov statistics

The multiscale kernel builds a Kolmogorov law into its weights. We now test whether interpolation on finite stencils preserves this law. We reconstruct a three-dimensional solenoidal Fourier ensemble whose velocity increments follow the Kolmogorov law $S_2(r) \propto r^{2/3}$ over the separations $0.03 \leq r \leq 0.8$. The ensemble is a sum of 1514 Fourier modes with Gaussian coefficients perpendicular to their wave vectors and a $-5/3$ energy spectrum, restricted to Ω_3 (Appendix D.2.3). Below $r = 0.03$, its finite bandwidth makes the increments smooth. For pairs of points $x_k(r)$ and $y_k(r)$ at distance r along eight fixed directions $k = 1, \dots, 8$, we compute the second-order structure function exactly from the covariance of the ensemble,

$$S_{2,k}(r) = \mathbb{E}[\|v(x_k(r)) - v(y_k(r))\|_2^2], \quad \|x_k(r) - y_k(r)\|_2 = r, \\ S_2(r) = \frac{1}{8} \sum_{k=1}^8 S_{2,k}(r), \quad (70)$$

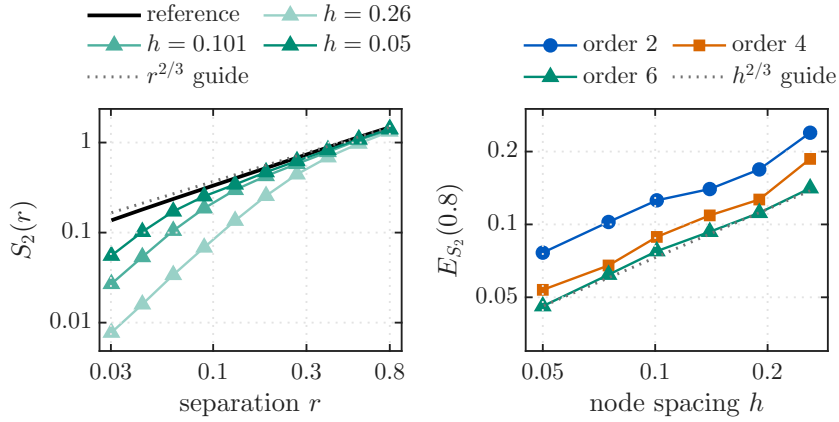


Figure 4: Kolmogorov statistics of the local multiscale Matérn output reconstruction. Left: structure functions for degree five on three of the six node sets, and the exact reference. Right: worst relative error $E_{S_2}(0.8)$ of (71) of the reconstructed structure function against the node spacing h , with an $h^{2/3}$ guide.

both for the ensemble and, with v_h in place of v , for its output reconstruction ($S_{2,k,h}$). We measure the worst relative error over the directions,

$$E_{S_2}(r) = \max_{1 \leq k \leq 8} \frac{|S_{2,k,h}(r) - S_{2,k}(r)|}{S_{2,k}(r)}. \quad (71)$$

We use the multiscale Matérn kernel with polynomial degrees 1, 3, and 5 on six node sets, with node spacings h from 0.26 to 0.05 (128 to 13567 nodes).

Figure 4 shows the structure functions approaching the reference under refinement. At the largest separation, $r = 0.8$, the error decays like $h^{0.64}$, $h^{0.73}$, and $h^{0.67}$ for the three degrees. These are fits over a finite range of h , not asymptotic rates. For degree five, the fitted slope of S_2 over the upper part of the range moves toward the reference slope. However, the error at small separations stays large even on the finest node set, since these separations are below the node spacing. To see whether locality is to blame, we compare local and global interpolation on one node set, with the same kernel and polynomials (Appendix D.2.3). Both have nearly the same error, at $r = 0.8$ and over the whole range, so the small-separation error is likely due to the resolution of the nodes and not to the stencils.

5.1.4. Output reconstruction in the fluid benchmarks

We now test the output reconstruction on the predictions of the operator benchmarks (Section 5.2.1). The data sets hold the velocity at more mesh points than the m output points that we use as nodes, and the remaining points lie between the nodes. We reconstruct the predicted nodal velocities of the model with the smallest validation error at $N = 10000$ with the global divergence-free Matérn interpolant of our earlier work and with our local output reconstruction, and measure the error at these points (Table 2, Appendix A.5). Neither output reconstruction is uniformly more accurate. The global one is more accurate for both cylinder flows, the backward-facing step, and Taylor–Green vortices (space), by factors between 1.06 and 2.7. The local one is more accurate for the other six, by factors between 1.04 (turbulent species transport) and 10 (buoyancy-driven cavity), although each local system has at most 62 unknowns in 2D and 262 in 3D, against up to 21000 in the global system. For several flows, the errors between the nodes are much larger than those at the nodes, so they are set by the resolution of the m nodes and not by the operator regression. For both turbulent flows, the multiscale Matérn kernel has a lower error than the PHS, and validation selects $c_\sigma = 16$ for both. Almost all stencils keep the full polynomial space, except in the species-transport mixer, whose geometry truncates the rank on more than half the stencils (Table 7 in Appendix A.2).

5.1.5. Cost of the local output reconstruction

We now compare the setup cost of our local output reconstruction with that of a global divergence-free output reconstruction, as in PPKM, on scattered nodes in the unit square and the unit cube. The global setup assembles the dense $dm \times dm$ Gram matrix of a divergence-free Matérn kernel and factors it by Cholesky. The local setup selects the stencils, builds the polynomial bases and the local systems, factors them, and forms the evaluation weights of

Table 2

Test error between the nodes. We reconstruct the predicted nodal velocities of the 100 test inputs (model with the smallest validation error at $N = 10000$) and measure the mean relative velocity error at the evaluation points, the mesh points of each data set that are not among its m nodes. Global: the divergence-free Matérn interpolant of Sharma et al. [53] on all nodes. Local: our output reconstruction with the cubic PHS, or with the multiscale Matérn kernel ($c_\sigma = 16$) where marked (Appendix A.5 compares it with the PHS). Bold: smaller error in each row.

Data set	Nodes m	Evaluation points	Global	Local
Forced isotropic turbulence	7000	3000	1.88×10^{-2}	1.65×10^{-2} (multiscale)
Turbulent species transport	7000	4388	1.52×10^{-1}	1.47×10^{-1} (multiscale)
Merging vortices	500	6947	1.88×10^{-1}	1.35×10^{-1}
Lid-driven cavity	1000	8566	6.15×10^{-1}	8.71×10^{-2}
Cylinder flow (no shedding)	1000	8520	8.67×10^{-2}	9.19×10^{-2}
Cylinder flow (vortex shedding)	1000	8520	9.65×10^{-2}	1.12×10^{-1}
Backward-facing step	1000	6359	5.84×10^{-3}	1.04×10^{-2}
Buoyancy-driven cavity	5000	5514	9.42×10^{-2}	9.31×10^{-3}
Taylor–Green vortices (space)	500	6947	2.30×10^{-4}	6.20×10^{-4}
Taylor–Green vortices (spacetime)	500	6947	1.99×10^{-3}	6.20×10^{-4}

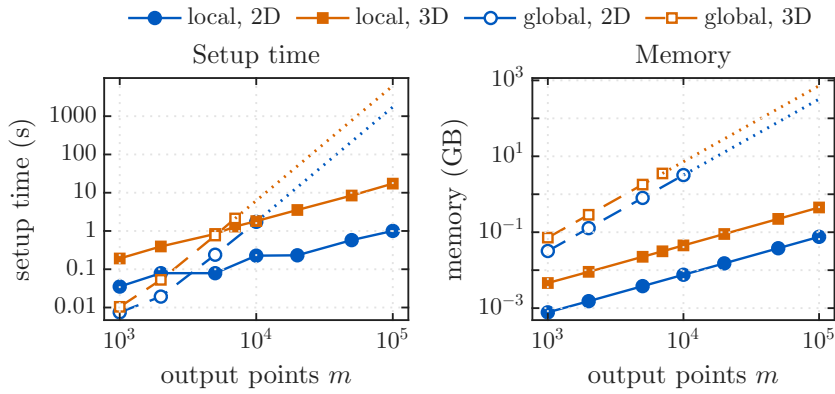


Figure 5: Setup time and memory of our local output reconstruction (degree four, cubic PHS) and of a global divergence-free Matérn output reconstruction (dense Gram matrix and Cholesky factor) against the number m of output points, on scattered nodes in the unit square (2D) and cube (3D). Dotted: the global cost extrapolated as $(dm)^3$ in time and $(dm)^2$ in memory beyond the largest size that a single dense factorization fits on our GPU. All times are measured after compilation.

Section 4.2, with the degree-four output reconstruction of the operator benchmarks. Both run on the GPU in double precision, and we time them after compilation. Figure 5 shows the results.

The local setup grows at most linearly in m , in both time and memory, as Table 1 predicts. The global setup grows like $(dm)^3$ in time and $(dm)^2$ in memory, and a single dense factorization no longer fits on our GPU beyond $dm \approx 2.5 \times 10^4$. It could be factored in blocks, at the same cubic cost. On small node sets, the dense global factorization is faster, since its work is done by a few large and efficient GPU kernels. The local setup becomes the faster one near $m = 3 \times 10^3$ in 2D and $m = 5 \times 10^3$ in 3D, and its memory is smaller at every size we tested, by a factor that grows linearly with m .

Two choices keep the local setup linear in m , and neither changes the output reconstruction beyond rounding. The polynomial bases are built by QR factorizations instead of singular value decompositions (Appendix A.2), and non-periodic node sets of more than 4096 nodes find their stencils by an exact search on a grid of cells, whose work grows linearly with m . Together, they make the setup of the output reconstruction a small part of the offline cost of every operator benchmark (Appendix D.6.1).

Table 3

The ten benchmarks: input and output of the operator, number of sampled input coordinates, number n_z of principal input coordinates (the numerical rank of the training inputs, capped at three), spatial dimension d , number m of output points, and spatial kernel. Taylor–Green vortices (spacetime) has spacetime outputs, at the same 500 points at four times.

Data set	Input $\rightarrow$ output	Input coord.	n_z	d	m	Spatial kernel
Forced isotropic turbulence	$\mathbf{f} \rightarrow \mathbf{u}(\cdot, 0.5)$	21000	3	3	7000	Five-scale Matérn
Turbulent species transport	inflow $\rightarrow \mathbf{u}(\cdot, 0.5)$	1388	2	3	7000	Five-scale Matérn
Merging vortices	$\omega(\cdot, 0) \rightarrow \mathbf{u}(\cdot, 0.4)$	500	3	2	500	Cubic PHS
Lid-driven cavity	$\mathbf{u}(\cdot, 0) \rightarrow \mathbf{u}(\cdot, 5)$	2000	1	2	1000	Cubic PHS
Cylinder flow (no shedding)	$\mathbf{u}(\cdot, 0) \rightarrow \mathbf{u}(\cdot, 10)$	2000	1	2	1000	Cubic PHS
Cylinder flow (vortex shedding)	$\mathbf{u}(\cdot, 0) \rightarrow \mathbf{u}(\cdot, 10)$	2000	1	2	1000	Cubic PHS
Backward-facing step	inflow $\rightarrow \mathbf{u}(\cdot, 5)$	11	1	2	1000	Cubic PHS
Buoyancy-driven cavity	$T(\cdot, 0) \rightarrow \mathbf{u}(\cdot, 5)$	5000	1	2	5000	Cubic PHS
Taylor–Green vortices (space)	$\mathbf{u}(\cdot, 0) \rightarrow \mathbf{u}(\cdot, 1)$	1000	1	2	500	Cubic PHS
Taylor–Green vortices (spacetime)	$\mathbf{u}(\cdot, 0) \rightarrow \mathbf{u}(\cdot, t), t \in [0.7, 1]$	1000	1	2	500	Cubic PHS

5.2. Sparse operator regression

5.2.1. Benchmarks and models

We use the ten data sets of our previous work [53], listed in Table 3. Eight are two-dimensional laminar flows, two of them the analytical Taylor–Green vortices, and two are three-dimensional turbulent flows, a Reynolds-averaged simulation of a static mixer and a large-eddy simulation of forced isotropic turbulence. In each, the input is an initial velocity, temperature, or vorticity, an inflow, or a body force, sampled at points of the domain or of its inflow boundary, and the output is the velocity at a fixed later time (or, for Taylor–Green vortices (spacetime), at four times), sampled at m output points. The output points are approximate Fekete points, i.e., well-spaced points selected from the mesh [53]. Appendix D.1 gives the geometries, parameter distributions, and simulations.

Each data set has 10000 training pairs, 100 validation pairs, and 100 test pairs. We fit every model on the first $N \in \{500, 1000, 5000, 10000\}$ training pairs, so the training sets are nested, and the validation and test pairs are the same for every N . The validation and test pairs were also used in our previous study and during the development of the method, so our comparisons are retrospective. Freezing all choices before the final test runs prevents test-based selection in this study but does not make the test set new. Our error measure, the test error, is the relative velocity error $\|\tilde{\mathbf{v}}^* - \mathbf{v}^*\|_2 / \|\mathbf{v}^*\|_2$ of the prediction $\tilde{\mathbf{v}}^*$ of a test field $\mathbf{v}^*$ at the output points, averaged over the 100 test fields (and, for Taylor–Green vortices (spacetime), over the four times). It measures the prediction for inputs that are used neither for fitting nor for validation.

The benchmarks differ less in their flows than in their inputs, and this affects how we interpret the results. In three of them, forced isotropic turbulence, turbulent species transport, and merging vortices, the inputs vary in at least two independent directions, and the operator is a multivariate map. In the other seven, the training inputs have numerical rank one. These inputs are multiples of one input function (e.g., the speed of the lid or the amplitude of the inflow), so the operator restricted to the data is a curve in the output space, and the operator regression is one-dimensional. Two of these, the Taylor–Green vortices, are degenerate in a further sense. Their outputs are proportional to the input amplitude, so we divide the responses by the amplitude before fitting and multiply by it after prediction. With $\mathcal{A}_i > 0$ the amplitude of training input i relative to the median training amplitude and $\mathcal{A}(\mathbf{a})$ that of a new input, we fit the normalized responses $\mathbf{v}_i / \mathcal{A}_i$ and predict

$$\tilde{\mathbf{v}}(\mathbf{a}) = \mathcal{A}(\mathbf{a}) \sum_{i=1}^N \ell_i(\mathbf{a}) \frac{\mathbf{v}_i}{\mathcal{A}_i}. \quad (72)$$

The normalized operator is then constant, which affine augmentation reproduces, and Proposition 2 applies to the normalized responses. All four models use this normalization for both Taylor–Green data sets, and the other eight data sets use unnormalized responses. Taylor–Green vortices (spacetime) thus tests our spacetime output reconstruction but not the operator regression.

We compare four models (Section 3.6). The *fixed-shape* kernel uses $D_{\parallel} = I$ and $\alpha_{\perp} = 1$ in (32), an isotropic metric on the principal coordinates and a unit scale for the residual coordinates. The *fitted-shape* kernel has its shape

fitted by the restricted likelihood. Each is used with and without affine augmentation, and each chooses its support radius and regularization by validation. For inputs of rank one, the shape has no free parameters, so the fitted-shape kernel differs from the fixed-shape one only through the regularization that the likelihood proposes, which enters the validation search.

All four models share one local output reconstruction, with divergence-free polynomials of degree $\varpi = 4$ (dimension 20 in 2D and 85 in 3D) on stencils of $n_s = 21$ and 59 nodes, the cubic PHS for the smooth flows, and the multiscale Matérn kernel with $c_\sigma = 16$ (which sets its lengths $\sigma_{j,k} = c_\sigma \rho_j / 2^k$ relative to the stencil radius ρ_j in (21)) for the two turbulent flows. Section 5.1.4 compares these choices with alternatives. All computations run in double precision on one NVIDIA RTX 5090.

5.2.2. Sparsity and convergence in N

We first ask whether a sparse kernel matrix is compatible with accurate prediction. Validation chooses the support radius, and with it the density $d_K = \text{nnz}(K_{UU})/N^2$ of the kernel matrix, among candidates with densities up to 90%. If accuracy required a dense matrix, validation would select the largest candidates. Figure 6 (right) shows the densities selected at $N = 10000$ for all ten benchmarks and all four models of Section 5.2.1. Validation selects 10% for the lid-driven cavity, both cylinder flows, and the buoyancy-driven cavity, about 33% for the backward-facing step, and between 9% and 25% for turbulent species transport and merging vortices. Nearly dense matrices are selected only for forced isotropic turbulence and for Taylor–Green without augmentation. The value 10% is the sparsest of the five prescribed densities that validation evaluates, although its gradient refinement can also reach sparser matrices.

Figure 7 shows how the selected density changes with N . For most benchmarks, it decreases as N grows, from between 20% and 90% at $N = 500$ to 10% at $N = 10000$ for the lid-driven cavity and both cylinder flows, so the sparsity of the operator regression increases with the size of the training set. Forced isotropic turbulence stays at 90% for every N , and the backward-facing step at about 33%.

With these sparse matrices, the test errors decrease steadily as the training set grows. Figure 8 shows the errors for the three multivariate flows and for the lid-driven cavity, a representative flow with rank-one inputs. Least-squares fits give errors proportional to $N^{-\tau}$ with τ between 0.4 and 1.3 for every benchmark and model (Table 9 in Appendix D.6), with two exceptions. With augmentation, the backward-facing step decays only like $N^{-0.15}$. For Taylor–Green, the augmented models reproduce the constant normalized operator to rounding error, while the error of the unaugmented models grows slowly with N .

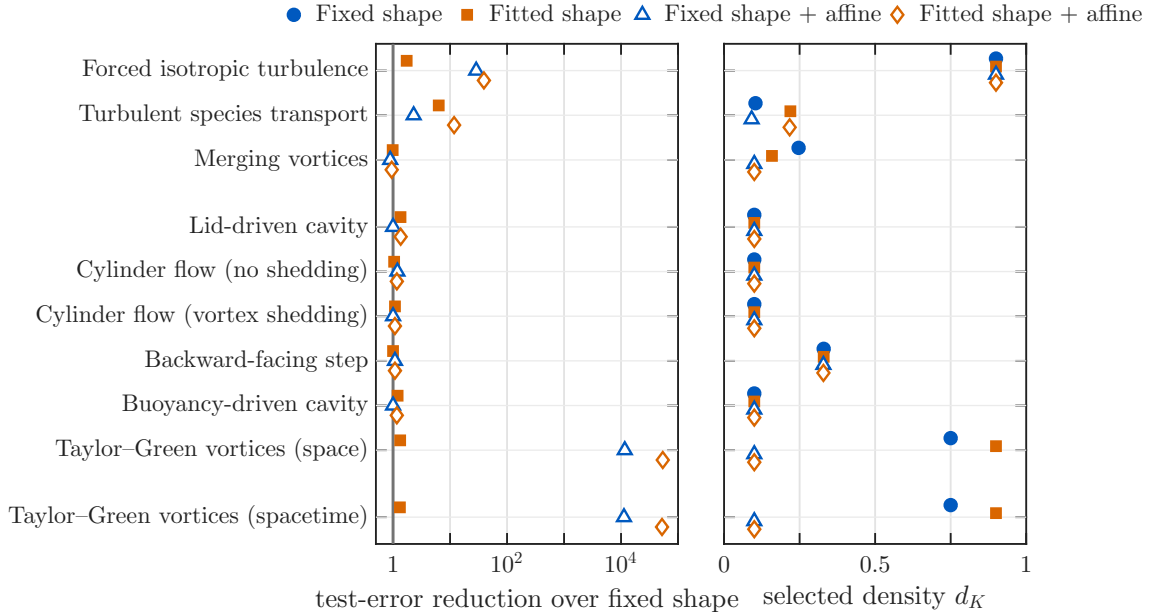


Figure 6: All benchmarks at $N = 10000$. Left: test error of the fixed-shape kernel ($D_{\parallel} = I$, $\alpha_{\perp} = 1$) without augmentation, divided by the test error of each other model (values above one are improvements). Right: density d_K of the selected kernel matrix.

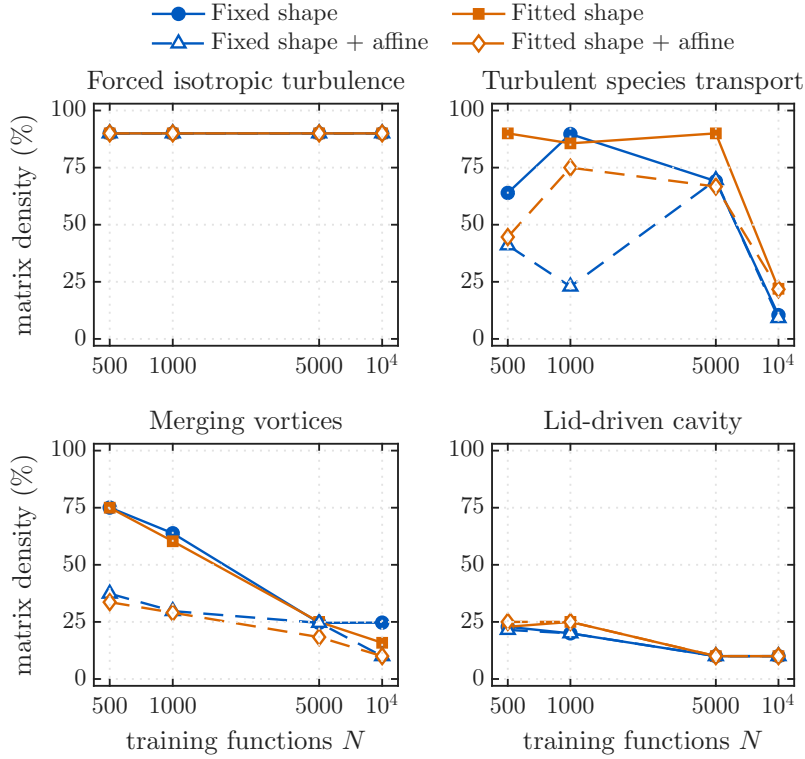


Figure 7: Selected density d_K of the kernel matrix against the number N of training functions for the three multivariate flows and the lid-driven cavity. The affine columns are not counted. Figure 21 shows the other benchmarks.

5.2.3. Kernel shape and affine augmentation

We next compare the four models. Figure 6 (left) shows, for each benchmark at $N = 10000$, the error of the fixed-shape kernel without augmentation divided by the error of each other model. On the five rank-one flows other than Taylor–Green, these ratios lie between 0.98 and 1.36, so neither the fitted shape nor the augmentation changes the error by more than about a quarter. This is consistent with the operator regression being one-dimensional on these data.

On the multivariate flows, however, the ratios are larger and depend on the flow. For forced isotropic turbulence, augmentation reduces the error by a factor of 29, and by 39 together with the fitted shape, at the same density. For turbulent species transport, the fitted shape reduces it by a factor of 6, and by 12 together with augmentation, while doubling the selected density. For merging vortices, all four errors lie within 12% of each other. Augmentation also lowers the selected density for merging vortices and for both Taylor–Green benchmarks. Neither choice is uniformly better, so we select both by validation.

5.2.4. Comparison with earlier results

We now compare our models with the results that Sharma et al. [53] report on the same data sets for dense PPKM, for the kernel method of Batlle et al. [2] (VKM), which maps the samples of the input to the samples of the output, and for two neural operators, Geo-FNO [32] and Transolver [70], each trained with and without a divergence penalty. Table 4 lists these errors at the training sizes used there, together with the error of our model that has the smallest validation error among the four. The comparison has two caveats. The earlier errors are averages over all 200 held-out pairs, which include our 100 validation pairs, and the neural operators were trained in single precision.

Our models are at least as accurate as dense PPKM on all ten data sets, by factors between 1.06 (cylinder flow with vortex shedding) and about 56 (merging vortices), and the same holds against VKM. Against the better of the two neural operators, the factor ranges from 2 for forced isotropic turbulence to more than 10^6 for Taylor–Green. The gains over dense PPKM are likely due to the affine augmentation, the fitted kernel shape, and the validation of the radius

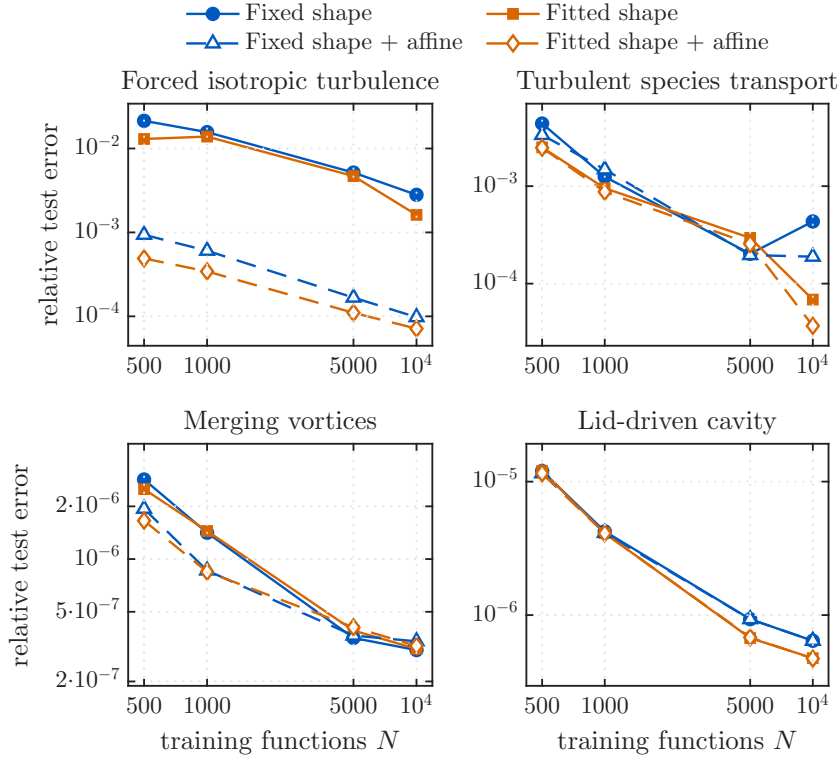


Figure 8: Test error (mean relative velocity error of the predictions for the 100 test inputs) against the number N of training functions for the three multivariate flows and the lid-driven cavity. Dashed: with affine augmentation. Figure 20 shows the other benchmarks.

Table 4

Test error (mean relative velocity error of the predictions for held-out inputs) of our sparse local model (the one of the four with the smallest validation error) and of the methods reported by Sharma et al. [53] at the same training size N : dense PPKM, the kernel method of Batlle et al. [2] on output samples (VKM), and Geo-FNO and Transolver, each with the better of its errors with and without a divergence penalty. The earlier errors are averages over all 200 held-out pairs and ours over the 100 test pairs. Our models of the cylinder flow without shedding at $N = 100$ were fitted for this comparison, with the settings of the other training sizes. Bold: smallest test error in each row.

Data set	N	Ours	PPKM	VKM	Geo-FNO	Transolver
Forced isotropic turbulence	10000	7.1×10^{-5}	3.6×10^{-4}	1.6×10^{-4}	1.5×10^{-4}	1.2×10^{-3}
Turbulent species transport	10000	3.7×10^{-5}	2.4×10^{-4}	2.4×10^{-4}	5.0×10^{-4}	2.7×10^{-3}
Merging vortices	500	1.9×10^{-6}	1.1×10^{-4}	1.1×10^{-4}	1.1×10^{-3}	3.4×10^{-3}
Lid-driven cavity	10000	4.7×10^{-7}	1.4×10^{-6}	1.5×10^{-6}	1.5×10^{-4}	2.1×10^{-4}
Cylinder flow (no shedding)	100	2.8×10^{-5}	4.0×10^{-5}	5.3×10^{-5}	1.6×10^{-4}	5.7×10^{-3}
Cylinder flow (vortex shedding)	10000	3.0×10^{-7}	3.1×10^{-7}	2.3×10^{-6}	3.7×10^{-5}	1.5×10^{-4}
Backward-facing step	500	9.5×10^{-8}	1.5×10^{-6}	2.9×10^{-6}	9.3×10^{-5}	7.2×10^{-4}
Buoyancy-driven cavity	10000	4.5×10^{-6}	6.3×10^{-6}	6.1×10^{-6}	1.4×10^{-4}	1.9×10^{-4}
Taylor–Green vortices (space)	5000	6.0×10^{-11}	8.8×10^{-10}	1.7×10^{-9}	4.2×10^{-4}	6.1×10^{-5}
Taylor–Green vortices (spacetime)	5000	5.4×10^{-11}	8.7×10^{-10}	8.7×10^{-10}	7.8×10^{-4}	2.2×10^{-4}

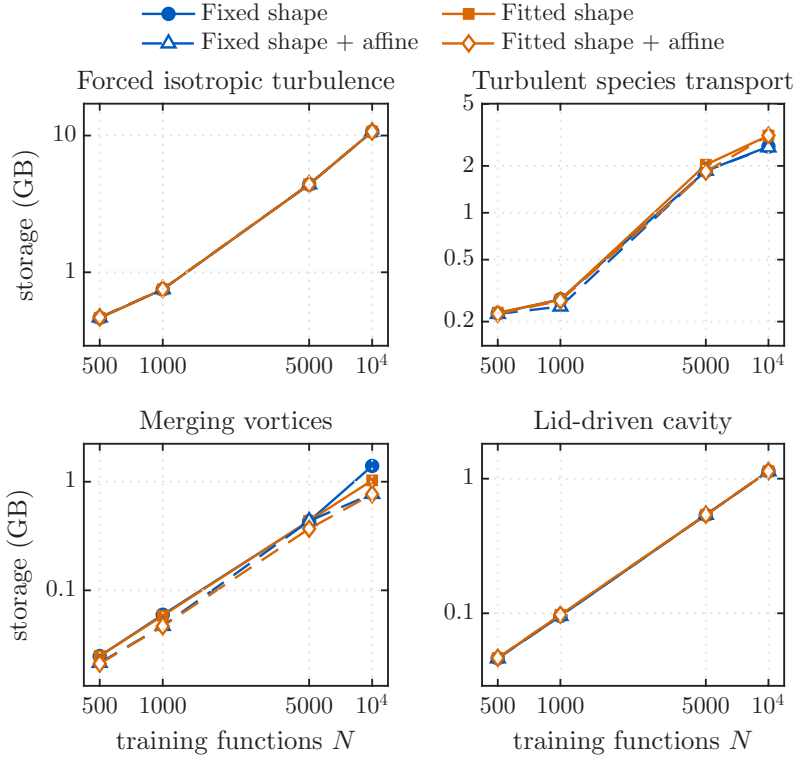


Figure 9: Accounted storage of the fitted models (stored arrays and factors, not measured peak memory) against N for the three multivariate flows and the lid-driven cavity. Figure 22 shows the other benchmarks.

and regularization, and for Taylor–Green also to the amplitude normalization. Thus, at these training sizes, the sparse operator regression is no less accurate than the dense one.

In addition, the neural-operator predictions in Sharma et al. [53] have maximum pointwise divergences between about 0.1 and 20, while ours are divergence free on each stencil. The training times reported there, from seconds to about a minute and a half for the kernel methods with fixed shape parameters and from minutes to almost a day for the neural operators, are not comparable to our selection times, which include the search over radii and regularizations (Appendix D.6.1).

5.2.5. Storage

The storage of a model consists mainly of the Cholesky factor of its kernel matrix and the sampled training fields, which the prediction (59) uses in place of their local coefficients. Figure 9 shows the storage against N . Over our range of N , the training fields dominate, and the storage of most models grows roughly linearly in N , with least-squares exponents between 0.9 and 1.1. The exceptions are merging vortices, the backward-facing step, and the unaugmented Taylor–Green models, with exponents between 1.2 and 1.7, likely because the factor is a larger share of their storage (small outputs, or densities that do not decrease). The Cholesky factor itself grows like $N^{1.4}$ to $N^{2.4}$, since even a decreasing density leaves a number of nonzeros that grows faster than N , and it would eventually dominate for larger training sets.

The size of the factor depends on the ordering as well as on the density. With the nested-dissection ordering of Section 3.5, the factors at $N = 10000$ have between 1 and 4.3 times fewer entries (a median of 2.8) than with the approximate-minimum-degree ordering, with identical predictions. Forced isotropic turbulence gains nothing, since its nearly dense kernel matrix has a dense factor in any ordering. The fill also makes storage not proportional to the density. For example, the fitted-shape augmented species model has 2.4 times the density of the fixed-shape augmented one, but only 1.4 times as many factor entries. For this reason, we report storage separately from density (Table 11).

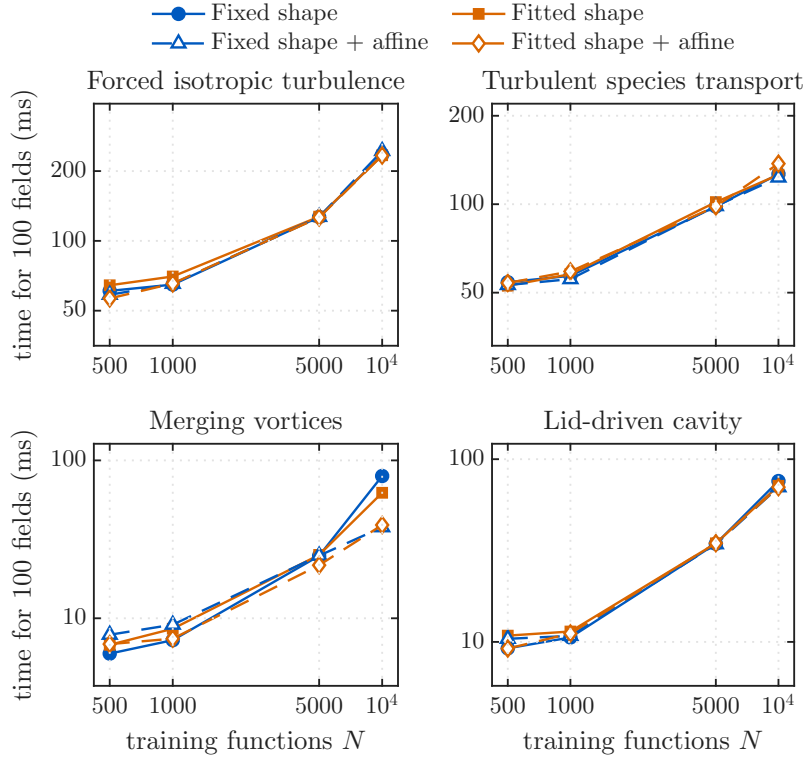


Figure 10: Median time to predict all 100 test fields after setup, including the kernel evaluation, the sparse solves, and the local output reconstruction. Fitting and setup are excluded (Appendix D.6.1).

5.3. Operator Lagrange weights

We now turn to the operator Lagrange weights of Section 4. Their main purpose is computational. They make a prediction cost one sparse solve per new input, regardless of the size of the output representation, and they let us combine the training fields before the output reconstruction. We first measure these savings. We then test the prediction of Theorem 3 that the weights are also more accurate than direct evaluation, but only when the output reconstruction amplifies solve errors.

5.3.1. Storage of the training data

Since the weights are applied to the sampled training fields, a prediction needs the Nn_v sampled values, with $n_v = dm$, and not the $N \sum_j r_j$ local coefficients of the training fields. With one stencil per output point, the ratio of the two is $\sum_j r_j / (dm) = n_s + n_p/d$ when every stencil keeps the full polynomial space, of dimension n_p . For our degree-four output reconstructions, the local coefficients would thus take 31 times the storage of the training fields in 2D and 87 times in 3D. For forced isotropic turbulence at $N = 10000$, for example, this is the difference between 1.7 and 147 GB in double precision.

5.3.2. Prediction cost

Figure 10 shows the time to predict all 100 test fields once the model is set up, including the evaluation of the kernel, the sparse solves for the weights, and the local output reconstruction. For the lid-driven cavity, a twentyfold increase in N raises this time seven- to eightfold, so prediction grows sublinearly in N over this range. The turbulent flows cost more at $N = 10000$, about three times the time of the lid-driven cavity for forced isotropic turbulence and 1.7–2.0 times for turbulent species transport, since they select denser kernels with more fill.

The ordering also affects the prediction time. Nested dissection makes the median prediction 1.3 times faster at $N = 10000$ than approximate minimum degree, since its smaller factors also give more parallel triangular solves. Fitting the kernel shape has no consistent effect on prediction time, but it adds offline cost, since a fitted-shape model takes several times longer to select than a fixed-shape one (Appendix D.6.1).

Table 5

Amplification factor $\alpha_{\text{spatial}} = \|B\|_F \|(\mathbf{A}^*)^\top\|_F / \|B(\mathbf{A}^*)^\top\|_F$ of (66), which measures the cancellation among the spatial coefficients, for $N = 1000$ training fields, for the global and for our local output reconstruction (stencils combined as in Section 5.3), and the largest ratio of the direct to the Lagrange error with global output reconstruction from Table 6. Bold: smaller amplification factor in each row.

Data set	Global α_{spatial}	Local α_{spatial}	Largest gain (global)
Forced isotropic turbulence	3.7×10^9	396	1.2
Turbulent species transport	2.9×10^9	9882	7.1
Merging vortices	4.8×10^9	73	868.1
Lid-driven cavity	5.8×10^9	13	34.5
Cylinder flow (no shedding)	5.1×10^7	149	4.8
Cylinder flow (vortex shedding)	5.2×10^7	149	2.0
Backward-facing step	5.3×10^8	38	215.3
Buoyancy-driven cavity	5.9×10^9	6	14.8
Taylor–Green vortices (space)	1.6×10^7	73	6.1
Taylor–Green vortices (spacetime)	1.9×10^2	145	1.0

5.3.3. Accuracy of direct and Lagrange evaluation

Recall the amplification factor $\alpha_{\text{spatial}} = \|B\|_F \|(\mathbf{A}^*)^\top\|_F / \|B(\mathbf{A}^*)^\top\|_F$ of (66), which compares the size of the spatial coefficients B of the training fields and of the evaluation matrix $\mathbf{A}^*$ with the size of the reconstructed velocities. It is large when large coefficients cancel in the output reconstruction. By (67), a solve error enters direct evaluation multiplied by α_{spatial} , and Lagrange evaluation without this factor. The theorem therefore predicts a gain from Lagrange evaluation where α_{spatial} is large, and none where it is small. We test this prediction in two steps.

First, we compute α_{spatial} for all ten benchmarks, from 1000 training fields, for the global divergence-free Matérn output reconstruction of our earlier work and for our local output reconstruction. For the local output reconstruction, which has one B_j and one $\mathbf{A}_j^*$ per stencil, we combine the stencils as $(\sum_j \|B_j\|_F^2 \|\mathbf{A}_j^*\|_F^2)^{1/2} / \|B(\mathbf{A}^*)^\top\|_F$, which bounds the amplification of the solve errors summed over all stencils and reduces to α_{spatial} for a single global stencil. Table 5 shows that the coefficients of the global output reconstructions are large and cancel, with α_{spatial} above 10^7 , up to 6×10^9 , on all benchmarks except Taylor–Green vortices (spacetime), whose kernel is less flat. Our local output reconstructions have α_{spatial} between 6 and 150 in 2D, 396 for forced isotropic turbulence, and about 10^4 for turbulent species transport, where more than half the stencils have a truncated polynomial rank (Table 7).

Second, we evaluate the same operator models both ways, with everything else held fixed. These are the unaugmented models of Section 5.2, which the theorem covers: the fixed-shape kernel at the radii of the five prescribed densities and at its validated radius, and the fitted-shape kernel at its validated parameters, for N from 500 to 10000 (Appendix D.4). Table 6 lists the ratio of the direct error to the Lagrange error on all ten benchmarks.

The results agree with this prediction. With global output reconstruction, Lagrange evaluation is more accurate by up to a factor of 868 (merging vortices), with median gains above 5 for merging vortices, the lid-driven cavity, and the backward-facing step. With our local output reconstruction, the two evaluations agree to within a relative difference of 3×10^{-6} for every model, as the small α_{spatial} implies. A large α_{spatial} allows a gain but does not guarantee one, since the gain also requires a solve residual that is not negligible. Forced isotropic turbulence, for example, has $\alpha_{\text{spatial}} \approx 4 \times 10^9$ with its global output reconstruction but a gain of at most 1.2.

Over all 560 pairs, Lagrange evaluation is never less accurate than direct evaluation by more than 0.02%. Appendix D.4 follows one global model of the lid-driven cavity as N grows (Table 8). There, the gain grows from 1.3 to 5.1 together with the norm of the spatial coefficients, and solving the direct system for the sampled velocities instead of the coefficients removes it, which indicates that the large coefficients are the cause.

Lagrange evaluation thus serves our local output reconstruction through its cost and storage, and it would also protect a global or otherwise ill-conditioned output reconstruction, such as that of our earlier method, from the amplification of solve errors.

Table 6

Ratio of the test error of direct evaluation to that of Lagrange evaluation, for the same unaugmented operator models (the fixed-shape kernel of Section 3.1 at the radii of the five prescribed densities and at its validated radius, and the fitted-shape kernel at its validated radius, for $N = 500, 1000, 5000$, and 10000) with global and local output reconstruction. Values above one favor Lagrange evaluation. Over all 560 pairs, the smallest ratio is 0.99987.

Data set	Global output reconstruction			Local output reconstruction	
	Models	Median	Max	Models	Max ratio-1
Forced isotropic turbulence	28	1.00	1.2	28	5.9×10^{-11}
Turbulent species transport	28	1.00	7.1	28	8.6×10^{-7}
Merging vortices	28	48.96	868.1	28	4.2×10^{-8}
Lid-driven cavity	28	5.07	34.5	28	1.4×10^{-7}
Cylinder flow (no shedding)	28	1.05	4.8	28	7.0×10^{-8}
Cylinder flow (vortex shedding)	28	1.01	2.0	28	2.4×10^{-8}
Backward-facing step	28	16.51	215.3	28	2.0×10^{-8}
Buoyancy-driven cavity	28	1.47	14.8	28	1.9×10^{-8}
Taylor–Green vortices (space)	28	1.05	6.1	28	1.7×10^{-6}
Taylor–Green vortices (spacetime)	28	1.00	1.0	28	2.4×10^{-6}

6. Conclusions

In this article, we presented a local and sparse kernel method for learning the solution operators of incompressible flows. It keeps the guarantee of our earlier property-preserving method [53], that every prediction is divergence free, and removes the two costs of that method that grow with the problem size. Local divergence-free interpolants replace the global output reconstruction, so the output can be refined without enlarging any linear system, and the local systems are independent and solved in parallel on the GPU. The output reconstruction is divergence free, periodic on periodic domains, and, with the multiscale Matérn kernel, approaches Kolmogorov statistics under refinement at the separations that the nodes resolve. A compactly supported kernel that is positive semidefinite in any input dimension replaces the dense operator regression, so the kernel matrix is sparse. Its support radius, and with it the sparsity, is fitted to the data, and optional affine polynomials in the input, which the operator regression reproduces exactly, represent trends that a compactly supported kernel cannot. Operator Lagrange weights join the two, so each new input costs one sparse solve and the local coefficients of the training fields are never formed.

The local output reconstruction converges at close to the order of its polynomial degree on scattered nodes, is about as accurate between the nodes as the global output reconstruction (more accurate on six of ten benchmarks and less accurate on the other four) with systems that are smaller by about two orders of magnitude, is cheaper to set up than a global output reconstruction beyond a few thousand output points, with memory that grows linearly with their number, keeps the divergence and the periodic mismatch at rounding level, and, with the multiscale Matérn kernel, approaches prescribed Kolmogorov statistics at the separations that the nodes resolve.

In the operator benchmarks, the test errors decrease roughly like $N^{-\tau}$ with τ between 0.4 and 1.3 in all but two degenerate cases, and the most accurate kernels are sparse on all benchmarks but one, with densities that mostly decrease as the training set grows. At the training sizes of our earlier study, these sparse models are at least as accurate as dense PPKM on every data set, and more accurate than the neural operators reported there by factors from 2 to more than 10^6 . Over our range of training sizes, the storage of most models grows roughly linearly in N . The benchmarks with rank-one inputs, apart from Taylor–Green, are insensitive to the choice of kernel and augmentation. The benchmarks with multivariate inputs are not, and the best choice differs between them. Affine augmentation gives the largest gain for forced isotropic turbulence, a fitted kernel shape for turbulent species transport, and neither helps for merging vortices. Our recommendation is therefore to select these choices by validation and to report error, density, and storage together.

Lagrange evaluation removes the need to store the local coefficients of the training fields, which would take $n_s + n_P/d$ times (the stencil size plus the number of polynomials per velocity component) the storage of the training fields themselves. Its accuracy agrees with our rounding analysis. With our local output reconstruction, whose amplification factors are at most about 10^4 , it is as accurate as direct evaluation. With global output reconstructions, whose amplification factors reach 6×10^9 , it is up to nearly three orders of magnitude more accurate.

Several extensions are natural. First, benchmarks with richer input families, and a comparison with dense operator regression and with neural operators on identical test sets and hardware, would test these conclusions beyond our ten

benchmarks. Second, our local interpolants are divergence free on each stencil but are not coupled across stencils. Blending them with a partition of unity would give a continuous field, but the blending weights introduce divergence. The divergence-free partition-of-unity method of Drake et al. [12] avoids this by blending potentials instead of velocities. Third, at a fixed density, the storage of the kernel matrix still grows quadratically with the number of training functions, which hierarchical or multilevel sparse factorizations could reduce. Finally, our rounding analysis covers the model without augmentation and evaluation through coefficients. Extending it to the affine correction and to evaluation before the output reconstruction would complete the analysis.

Credit authorship contribution statement

Ramansh Sharma: conceptualization, methodology, software, investigation, validation, visualization, writing – original draft, review, editing.

Matthew Lowery: methodology, software

Houman Owahdi: conceptualization, methodology, supervision.

Varun Shankar: conceptualization, funding acquisition, methodology, project administration, resources, software, supervision, validation, visualization, writing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Our implementation, including the scripts that build the tables and figures of this paper, is available in KernelPack-JAX at <https://github.com/VarShankar/kernelpack-jax>. The ten benchmark data sets are those of Sharma et al. [53] and are publicly available on Zenodo [54].

Acknowledgements

RS, ML, and VS were supported by the National Science Foundation under DMS 2505986, Cooperative Agreement 2421782, and the Simons Foundation grant MPS-AI-00010515 awarded to the NSF-Simons AI Institute for Cosmic Origins — CosmicAI, <https://www.cosmicai.org>. VS was additionally supported by the Air Force Office of Scientific Research (AFOSR) under LRIR award FA9550-25-1-0042. HO acknowledges support from the Air Force Office of Scientific Research under MURI award number FOA-AFRL-AFOSR-2023-0004 (Mathematics of Digital Twins), the Department of Energy under award number DE-SC0023163 (SEA-CROGS: Scalable, Efficient, and Accelerated Causal Reasoning Operators, Graphs and Spikes for Earth and Embedded Systems) and the DoD Vannevar Bush Faculty Fellowship Program under ONR award number N00014-18-1-2363.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the authors used both Codex (OpenAI) and Claude (Anthropic) to help clean up the manuscript and implement parts of the software. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

A. Details of the local output reconstruction

A.1. Ordering of vector coefficients

The coefficient vectors of (17) are ordered by component, with all values of one component before those of the next, but the kernel evaluation at a query point $y_q^\star$ is naturally ordered by stencil node,

$$\Phi_{j(q),q,\text{pt}}^\star = [\Phi_{\text{DF}}(y_q^\star, \tilde{y}_{j(q),1}) \quad \cdots \quad \Phi_{\text{DF}}(y_q^\star, \tilde{y}_{j(q),n_s})] \in \mathbb{R}^{d \times d n_s}. \quad (73)$$

Table 7

Retained divergence-free polynomial ranks of the local systems on the nodes of each benchmark, at degree four and with periodic images, from the construction above. The full ranks are 20 in 2D and 85 in 3D. Taylor–Green vortices (spacetime) has the stencils of Taylor–Green vortices (space).

Data set	Stencils	Retained rank	Truncated stencils
Turbulent species transport	7000	45–85	4338
Forced isotropic turbulence	7000	85	0
Buoyancy-driven cavity	5000	20	0
Lid-driven cavity	1000	20	0
Cylinder flow (no shedding)	1000	20	0
Cylinder flow (vortex shedding)	1000	20	0
Backward-facing step	1000	20	0
Taylor–Green vortices (space)	500	19–20	1
Merging vortices	500	19–20	1

Let Π_{d,n_s} be the permutation with $\Pi_{d,n_s} \mathbf{b}_{\Phi,\text{comp}} = \mathbf{b}_{\Phi,\text{pt}}$, where $\mathbf{b}_{\Phi,\text{pt}}$ groups the d components of each node. The evaluation matrix in (18) is then

$$\Phi_{j(q),q,\text{comp}}^* = \Phi_{j(q),q,\text{pt}}^* \Pi_{d,n_s}. \quad (74)$$

A.2. Polynomial basis and local solves on the GPU

We first remove the dependencies among the candidates that do not depend on the stencil, once, before sampling. A singular value decomposition of the fixed map from candidates to polynomial coefficients gives an orthonormal basis Ξ of its row space, shared by all stencils of one dimension and degree, with

$$G_j = G_j \Xi \Xi^T, \quad \Xi^T \Xi = I, \quad (75)$$

in exact arithmetic. Thus $G_j \Xi$ has the nonzero singular values of G_j , and we factor it by thin QR, $G_j \Xi = \tilde{Q}_j \tilde{R}_j$. On most stencils, $\tilde{R}_j$ is far from singular, and the polynomial basis is

$$\mathbf{P}_j(y) = G_j(y) \Xi \tilde{R}_j^{-1}, \quad (76)$$

whose values at the stencil nodes are the orthonormal columns of $\tilde{Q}_j$, as in (14). We use (76) when the eigenvalues of $\tilde{R}_j^T \tilde{R}_j$ certify that its smallest singular value is at least 10^{-6} times its largest, far above the rank threshold of 10^{-12} , since these eigenvalues carry absolute errors of the order of the unit roundoff times the largest one. On the remaining stencils, the column-pivoted QR factorization $\tilde{R}_j \Pi_j = \hat{Q}_j \hat{R}_j$ determines the rank as in (12), and the basis is (14) with $G_{j,I}$ replaced by the retained columns of $G_j \Xi \Pi_j$. Both factorizations are batched over the stencils, and the columns of different stencils may differ, since every system is padded to the same width (below). A singular value decomposition on every stencil would decide the rank as well, but its batched form is slow on the GPU for these matrices (85×85 in 3D at degree four, about 10 ms each). On the species-transport node set, where most stencils are rank deficient, both rank decisions agree on every stencil, and the output reconstructions between the nodes agree to 10^{-12} .

To keep the batches of fixed size, every system carries all 20 (2D) or 85 (3D) polynomial columns at degree four. The columns of discarded modes are set to zero and given a unit diagonal, which decouples their coefficients without changing the interpolant. Every system therefore has the padded width 62 (2D) or 262 (3D), and our costs and storage include the padding.

If the LU solution of a local system is not finite, we use the pseudoinverse of that system, also for its evaluation weights. The resulting field is still divergence free, but it need not interpolate or reproduce the full polynomial space. The pseudoinverse was never needed. The LU solutions are finite on every node stencil of all ten operator benchmarks, including the rank-deficient stencils of Table 7, and in the manufactured tests of Section 5.1 every stencil kept the full polynomial space.

A.3. Evaluation away from the nodes

The operator benchmarks evaluate predictions at the nodes. To evaluate at another point, our GPU implementation selects a stencil centered at that point and builds its basis there. The comparisons of Table 2 use this rule. The

manufactured tests of Section 5.1 instead use the stencil of the nearest node, and their results between the nodes refer to that rule.

A.4. Periodic stencils

The Taylor–Green flows and the merging vortices have period 2π in each direction. Forced isotropic turbulence has period 2π on the box $[0, 2\pi]^3$, which we supply explicitly, since the cell centers of its grid do not reach the faces of the box.

A.5. Output reconstruction between the nodes

Table 2 reconstructs the test predictions at the nodes of the model with the smallest validation error among the four of Section 5.2.1, at $N = 10000$, so that the operator regression is that of Section 5.2 and only the output reconstruction changes. The evaluation points of Table 2 are all mesh or grid points of the data set that are not among the m output points. The local output reconstruction selects a stencil centered at each of them (Appendix A.3). The global output reconstruction is the divergence-free interpolant of Sharma et al. [53] with the scalar kernel $3M_{5/2}(\epsilon r)$ on all nodes, with the shape parameters of that work ($\epsilon = 0.3$ for the lid-driven cavity, 0.0636 for both cylinder flows, 0.06 for the backward-facing step, 0.8 for the buoyancy-driven cavity, 0.02 for merging vortices and Taylor–Green vortices (space), and 1.5 for Taylor–Green vortices (spacetime)), applied at each sampled time for Taylor–Green vortices (spacetime). For the periodic data sets, it uses one representative of each set of periodically equivalent nodes.

We select the scale factor c_σ of the multiscale kernel on the 100 validation fields, by reconstructing their exact nodal values and measuring the error between the nodes. For turbulent species transport, $c_\sigma = 2, 4, 8$, and 16 give errors of 0.166, 0.158, 0.150, and 0.147, and the PHS gives 0.169. For forced isotropic turbulence, they give 0.0228, 0.0206, 0.0178, and 0.0165, and the PHS gives 0.0227. Both select $c_\sigma = 16$, which the operator benchmarks use. For turbulent species transport, the error between the nodes remains much larger than at the nodes, so an accurate prediction at the nodes does not guarantee one everywhere.

We also compare the second-order structure functions

$$S_2(r) = \left\langle \|v(x) - v(y)\|_2^2 \right\rangle_{\|x-y\| \approx r} \quad (77)$$

of the output reconstructions and of the reference fields on the same pairs of points between the nodes (256 anchors with up to four partners in each of 12 logarithmic distance bins, with periodic distances for forced isotropic turbulence), averaged over the 100 test fields (Figure 11). The relative discrepancy of the binned structure function is 1.7% for the PHS and 2.6% for the multiscale kernel for turbulent species transport, and 0.40% and 0.20% for forced isotropic turbulence. The multiscale kernel thus lowers the pointwise error between the nodes for both flows, but improves the binned increments only for forced isotropic turbulence. Since the reference fields come from Reynolds-averaged (RANS) and coarse large-eddy (LES) simulations, this comparison measures agreement with the reference, not with physical turbulence.

B. Details of the operator regression

B.1. Normalization and principal directions

At every N , we compute the normalization and the principal directions from the first N training inputs alone. Each coordinate is divided by its standard deviation, or set to zero if that vanishes, and the numerical rank uses the threshold $\max\{10^{-10}, 2u_{\text{mach}} \max(N, nd_a)\}$ relative to the largest singular value, with u_{mach} the unit roundoff. The four models at one N share the normalization, the principal directions, and the gate (34), and the two models with augmentation share the affine matrix.

B.2. Fitting and validation

The fitted-shape kernel starts from the radius ρ_0 that gives 25% density and from the regularization $\theta = 10^{-6}$, and takes 60 steps of the Adam optimizer [31] with learning rate 0.03 for each of $c_d = 0$ and $c_d = 1$. Its log-radius is confined to $[\log(\rho_0/8), \log(4\rho_0)]$ and its regularization to $\theta = e^{t_\theta} + 10^{-12}$ with $t_\theta \in [\log(10^{-12}), \log(0.1)]$. The anisotropy uses $n_z - 1$ free parameters $b_k \in [-2, 2]$, completed by a zero, through $\tilde{v}_k = \log(2) \tanh(b_k)$ and $v_k = \tilde{v}_k - n_z^{-1} \sum_{k'=1}^{n_z} \tilde{v}_{k'}$. The residual scale lies in $[1/4, 4]$. It is inactive for the seven benchmarks with $n_z = 1$ and stays at one for turbulent species transport.

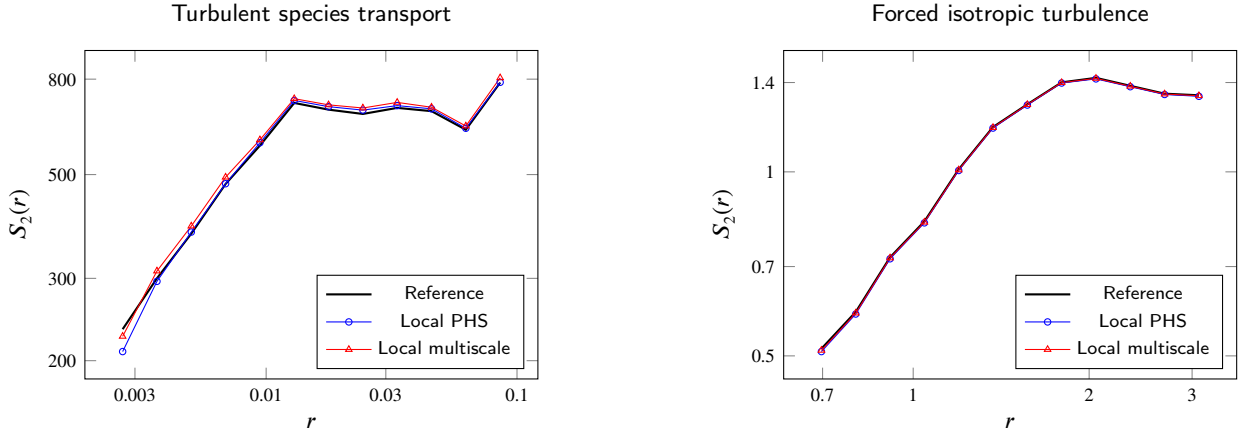


Figure 11: Second-order structure functions between the nodes of the reference fields and of the local PHS and multiscale Matérn output reconstructions of the test predictions at $N = 10000$.

Validation evaluates the radii of the five density budgets of Section 3.6. With $\varrho_{\max}$ the largest of these, it also searches radii in $[\varrho_{\max}/8, \varrho_{\max}]$ from four starts (radii $\varrho_{\max}/2$ and $\varrho_{\max}/4$, regularizations 10^{-6} and 10^{-3}), each with 35 Adam steps on all 100 validation fields. A fitted-shape model repeats this for both of its fitted shapes, so its search is twice as long. We keep the model with the smallest validation error among those with density at most 0.90, and break ties by density.

B.3. Work, storage, and timing

We assemble the kernel matrix by visiting all pairs of training inputs in blocks of bounded size and keeping those within the support. This keeps the memory bounded, but the work of the search is still quadratic in N , since compact support alone does not give a faster neighbor search. The likelihood and its gradient use the factors through

$$\Pi_U = L_K^{-\top} (I - Q_P Q_P^\top) L_K^{-1}, \quad (78)$$

with the entries of the inverse needed for the trace derivatives computed only on the support. When the outputs are wide, we replace E by the triangular factor of a thin QR factorization $E^\top = Q_E R_E$, since $EE^\top = R_E^\top R_E$.

The storage we report is the sum of the stored model arrays (including the training fields), the sparse matrix arrays, cuDSS's estimate of the memory of its factors, and the solve buffers. It is computed from the stored arrays instead of measured at its peak, and it excludes compiled code and temporary workspace. A kernel with $\mathcal{N}_K$ nonzeros in compressed sparse row form takes $12\mathcal{N}_K + 4(N + 1)$ bytes.

We report the setup of the output reconstruction, which all four models share, separately from the fitting and validation of each model. The warm prediction time of Figure 10 is the median of 11 repetitions of predicting all 100 test fields after the model is set up and compiled. The four models of a benchmark run in one process and share compiled code, so small differences between their times are not significant.

B.4. Batches and checks

The 100 test inputs are solved in four batches of 32 right-hand sides, the last one padded by repeated inputs. Repeated solves replay CUDA graphs (recorded sequences of GPU kernel launches). After every factorization, we check the residual of the solves, the reproduction of affine functions, and the agreement of the batched and compiled predictions with reference predictions. Every reported model passed these checks.

C. Proofs

C.1. Completeness of the polynomial candidates

In 2D, every divergence-free vector polynomial of degree at most ϖ has a polynomial stream function of degree at most $\varpi + 1$, so the rotated gradients of the scalar basis span the space. In 3D, split a divergence-free polynomial into

homogeneous parts v_j of degree j , each of which is divergence free. The identity

$$\nabla_\xi \times (v_j \times \xi) = (j+2)v_j \quad (79)$$

shows that v_j is the curl of the potential $(v_j \times \xi)/(j+2)$ of degree $j+1$, and expanding the components of this potential in the scalar basis gives a combination of the candidates (11). For the dimension, the divergence maps the vector polynomials of degree at most ϖ onto the scalar polynomials of degree at most $\varpi-1$ (integrate in one coordinate to find a preimage), and rank-nullity gives $d \binom{\varpi+d}{d} - \binom{\varpi+d-1}{d}$.

C.2. Positive definiteness of the kernel

Proof of Proposition 1. The scaled principal coordinates $D_{\parallel} z$ lie in $\mathbb{R}^{n_z}$ with $n_z \leq 3$, so the Wendland factor is a positive semidefinite kernel. The Matérn factor is positive semidefinite on the residual space in every finite dimension. The Gram matrix of their product is the Schur product of two positive semidefinite matrices, and multiplying it on both sides by the diagonal matrix of gates keeps it positive semidefinite.

For strictness, group distinct standardized inputs by their distinct principal coordinates $z^{(1)}, \dots, z^{(J)}$, and let W be the Wendland Gram matrix on these, which is positive definite. Let $\Psi_{\perp}$ be a feature map of the residual Matérn kernel, and for coefficients c_i let

$$f_i = \sum_{i: z_i = z^{(l)}} c_i \eta(\mathbf{a}_i) \Psi_{\perp}(\xi_i). \quad (80)$$

The quadratic form of the product Gram matrix is $\sum_{i,i'} W_{ii'} \langle f_i, f_{i'} \rangle \geq \lambda_{\min}(W) \sum_i \|f_i\|^2$. Inputs in the same group differ in their residuals, so strict positive definiteness of the Matérn kernel gives $f_i \neq 0$ whenever a coefficient of the group is nonzero and the gates are positive. (If the residual space is trivial, distinct inputs lie in distinct groups.) Finally, $c^T(K_{UU} + \theta I)c \geq \theta \|c\|_2^2 > 0$ for every $c \neq 0$ and $\theta > 0$. $\square$

The proof does not assume that the inputs lie in a low-dimensional affine subspace. Note that standardization can make two inputs equal if they differ only in coordinates with zero training variance. The regularization keeps the matrix positive definite in that case.

C.3. Variational form of the operator regression

Let $f_{\mathcal{H}} \in \mathcal{H}$, and split $f_{\mathcal{H}} = f_{\parallel} + f_{\perp}$, where $f_{\parallel}$ lies in the span of the functions $\lambda(\cdot, \mathbf{a}_i) \mathbf{e}_k$ and $f_{\perp}$ is orthogonal to it. By the reproducing property, $f_{\perp}(\mathbf{a}_i) = 0$ for every i , so $f_{\perp}$ leaves the data term of (42) unchanged and only increases the norm. A minimizer therefore has the form $f_{\mathcal{H}} = \sum_i \lambda(\cdot, \mathbf{a}_i) \mathbf{c}_i$, with $\|f_{\mathcal{H}}\|_{\mathcal{H}}^2 = \text{tr}(C^T K_{UU} C)$, where C has the rows $\mathbf{c}_i^T$. With B the matrix with rows $\mathbf{b}_i^T$ and $P = P_U$, the objective becomes the convex quadratic

$$J(C, \beta) = \|K_{UU}C + P\beta - B\|_F^2 + \theta \text{tr}(C^T K_{UU} C), \quad (81)$$

with gradients

$$\nabla_C J = 2K_{UU}(K_{UU}C + P\beta - B + \theta C), \quad \nabla_{\beta} J = 2P^T(K_{UU}C + P\beta - B). \quad (82)$$

The conditions (40) state that $K_{UU}C + P\beta - B = -\theta C$ and $P^T C = 0$. Both gradients therefore vanish, and every solution of (40) minimizes J , and hence (42). Without augmentation, the same argument applies with P and β removed.

C.4. Block elimination and reproduction

Proof of Proposition 2. Write $K = K_{UU}^{\theta} = L_K L_K^T$ and $P = P_U$. Since P has full column rank, $L_K^{-1} P = Q_P R_P$ with R_P invertible, and

$$\mathcal{E}_U P = R_P^{-1} Q_P^T L_K^{-1} P = I, \quad K \mathcal{E}_U^T = P R_P^{-1} R_P^{-T}. \quad (83)$$

Let $\ell_0 = K^{-1} k_U(\mathbf{a})$ and $\tilde{p} = p(\mathbf{a}) - P^T \ell_0$. Then $\ell = \ell_0 + \mathcal{E}_U^T \tilde{p}$ satisfies

$$P^T \ell = P^T \ell_0 + \tilde{p} = p(\mathbf{a}), \quad K \ell = k_U(\mathbf{a}) + P R_P^{-1} R_P^{-T} \tilde{p}, \quad (84)$$

so (51) holds with $\mu = -R_p^{-1} R_p^{-\top} \tilde{p}$. If (v, ω) solves the homogeneous system, then $P^\top v = 0$ and $v^\top K v = 0$, so $v = 0$, and full column rank gives $\omega = 0$. The solution is unique.

For targets in the space of affine functions, $B_j = P_U T_j$, we get $B_j^\top \ell = T_j^\top p(\mathbf{a})$, which is reproduction. The coefficients of (44) are

$$C_j = \Pi_U B_j, \quad \beta_j = \mathcal{E}_U B_j, \quad k_U(\mathbf{a})^\top C_j + p(\mathbf{a})^\top \beta_j = \ell(\mathbf{a})^\top B_j, \quad (85)$$

where the last identity follows by transposing the augmented system. Direct and Lagrange evaluation thus agree in exact arithmetic. Finally, linearity of χ and $\mathcal{D}$ gives (54). For a nonzero common constraint $\mathcal{D}\chi(\mathbf{b}_i) = f$, the affine model also gives $\mathcal{D}\chi(\sum_i \ell_i \mathbf{b}_i) = f$, since its constant column enforces $\sum_i \ell_i = 1$. $\square$

Without augmentation, $\ell = (K_{UU}^\theta)^{-1} k_U(\mathbf{a})$. If $\theta = 0$, then $k_U(\mathbf{a}_i)$ is the i th column of K_{UU} , so $\ell(\mathbf{a}_i) = \mathbf{e}_i$ and the weights are cardinal. With $\theta > 0$, the vector k_U excludes the regularization, and the weights are no longer cardinal.

C.5. Rounding errors of direct and Lagrange evaluation

Proof of Theorem 3. The first residual equation gives $\hat{C} = (K_{UU}^\theta)^{-1}(B - R_{\text{dir}})$, and multiplying by K_{U*U} and $(\mathbf{A}^\star)^\top$ gives (62). The second gives $\hat{L} = (K_{U*U} - R_{\text{Lag}})(K_{UU}^\theta)^{-1}$, and hence (63).

For the rounding, let $Z = \text{fl}(M_1 M_2)$ with $(M_1, M_2) = (K_{U*U}, \hat{C})$ or $(\hat{L}, B)$. The standard bound for matrix products [28, Chapter 3] gives $|Z - M_1 M_2| \leq \gamma_N |M_1| |M_2|$, and the product with $(\mathbf{A}^\star)^\top$ satisfies $|\text{fl}(Z(\mathbf{A}^\star)^\top) - Z(\mathbf{A}^\star)^\top| \leq \gamma_{r+d} |Z| |(\mathbf{A}^\star)^\top|$, also when it is computed in component blocks (dot products of length at most r followed by at most $d - 1$ additions). Together,

$$|\text{fl}(\text{fl}(M_1 M_2)(\mathbf{A}^\star)^\top) - M_1 M_2(\mathbf{A}^\star)^\top| \leq \gamma_{\text{mult}} |M_1| |M_2| |(\mathbf{A}^\star)^\top|. \quad (86)$$

The triangle inequality and submultiplicativity of the Frobenius norm then give (64)–(65). $\square$

D. Details of the numerical experiments

D.1. Benchmarks

The ten data sets are those of our previous work [53], where their simulations are described in full. Each figure shows the first simulation of a data set. The color fields interpolate the simulation samples linearly (Appendix D.3).

D.1.1. Flow past a cylinder

Two-dimensional flow passes a cylinder of radius 0.8 centered at (1.5, 7) in the channel $\Omega_a = \Omega_v = [0, 20] \times [0, 14]$, with an inflow on the left, freestream conditions at the top and bottom, no slip on the cylinder, and zero pressure at the outlet. The constant initial velocity varies within the vortex-shedding regime in one data set and below the onset of vortex shedding in the other. We learn $\mathcal{G} : \mathbf{u}(x, 0) \rightarrow \mathbf{u}(y, 10)$ on $m = 1000$ approximate Fekete points (well-spaced points selected from the mesh as in Sharma et al. [53]) of the 9520 mesh points.

D.1.2. Taylor–Green vortices

The analytical Taylor–Green vortices decay on the periodic domain $\Omega_a = \Omega_v = [0, 2\pi]^2$, sampled at 500 Fekete points. For the spatial data set, we learn $\mathcal{G} : \mathbf{u}(x, 0) \rightarrow \mathbf{u}(y, 1)$. For the spacetime data set, we learn the map from $\mathbf{u}(x, 0)$ to the velocity on $\Omega \times [0.7, 1]$, with snapshots at $\mathcal{T} = \{0.7, 0.8, 0.9, 1.0\}$ and cubic interpolation in time (Section 2.4). Both use the amplitude normalization (72).

D.1.3. Lid-driven cavity

In the cavity $\Omega_a = \Omega_v = [0, 1]^2$, the speed of the lid is drawn from a truncated normal distribution, and the fluid starts at rest, so the initial velocity is nonzero only on the lid. We learn $\mathcal{G} : \mathbf{u}(x, 0) \rightarrow \mathbf{u}(y, 5)$ on $m = 1000$ Fekete points selected from the 9566 mesh points.

D.1.4. Backward-facing step

A parabolic inflow, scaled by a factor drawn from a truncated normal distribution, drives the flow over a step into the channel $\Omega_a = \Omega_v = [0, 15] \times [-0.5, 0.5]$. We learn $\mathcal{G} : \mathbf{u}(x, 0) \rightarrow \mathbf{u}(y, 5)$ on $m = 1000$ Fekete points selected from the 7359 mesh points. The input has only 11 coordinates, the samples of the inflow profile.

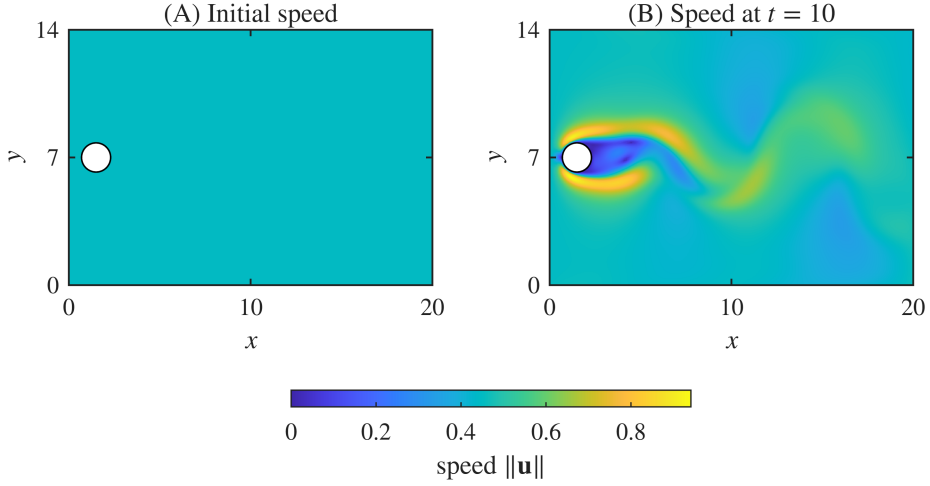


Figure 12: Cylinder flow. (A) Initial speed. (B) Speed at $t = 10$, on the same color scale.

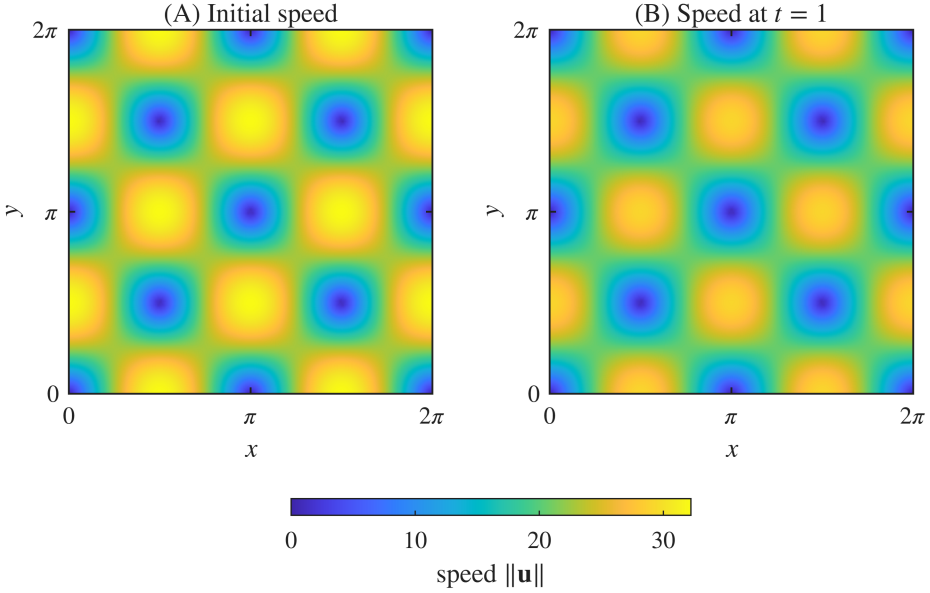


Figure 13: Taylor–Green vortices. (A) Initial speed. (B) Speed at $t = 1$, on the same color scale.

D.1.5. Buoyancy-driven cavity

In the cavity $\Omega_a = \Omega_v = [0, 1]^2$, an initial temperature field $T(x, 0)$ drives the flow by buoyancy. We learn the map from $T(x, 0)$ to the velocity at $t = 5$ on $m = 5000$ Fekete points selected from the 10514 mesh points.

D.1.6. Merging vortices

Two vortices merge on the periodic domain $\Omega_a = \Omega_v = [0, 2\pi]^2$, and the width of their cores varies across the data set. We learn $\mathcal{G} : \omega(x, 0) \rightarrow \mathbf{u}(y, 0.4)$ from the initial vorticity on 500 Fekete points selected from the 7447 mesh points.

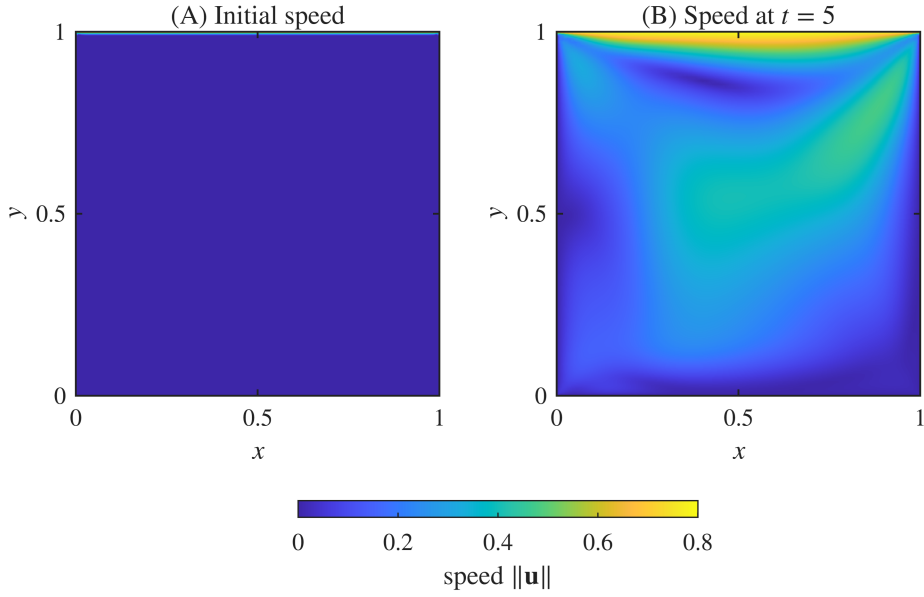


Figure 14: Lid-driven cavity. (A) Initial speed. (B) Speed at $t = 5$, on the same color scale.

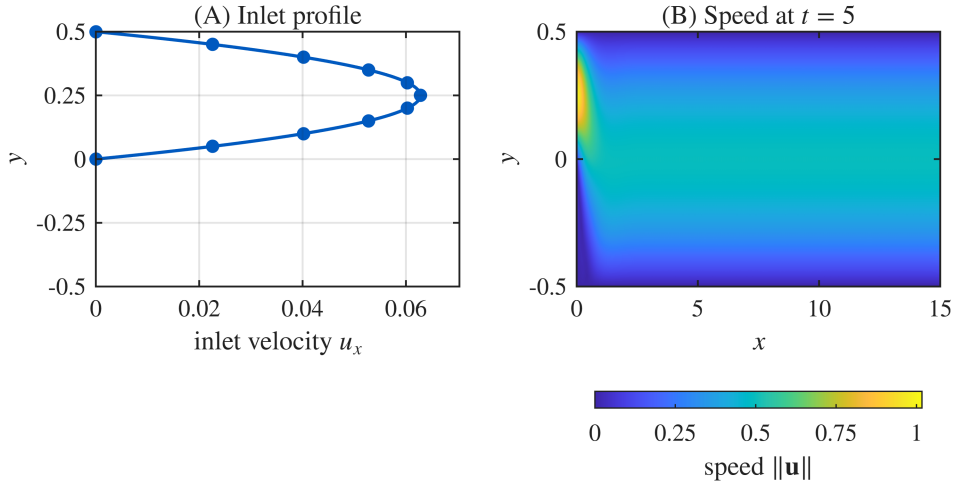


Figure 15: Backward-facing step. (A) Inflow profile. (B) Speed at $t = 5$, with the transverse direction expanded.

D.1.7. Turbulent species transport

Air and methane mix turbulently in a static Kenics mixer, whose three blades twist along the z axis, at Reynolds numbers between 10^5 and 10^6 . We learn the map from the inlet velocity (1388 samples) to the velocity at $t = 0.5$ on 7000 Fekete points selected from 11388 locations.

D.1.8. Forced homogeneous isotropic turbulence

Forced isotropic turbulence evolves in the periodic box $\Omega_a = \Omega_v = [0, 2\pi]^3$, simulated by large-eddy simulation with the Smagorinsky model on 32^3 cells, with $\Delta t = 0.002$, final time $T = 0.5$, and Reynolds number 6×10^6 . Every simulation starts from the divergence-free field

$$u_x(x, 0) = \sin(2x_3) + \cos(2x_2) + \frac{1}{2}(\sin(3x_3) + \cos(3x_2)) + \frac{1}{4}(\sin(5x_3) + \cos(5x_2)),$$

$$u_y(x, 0) = \sin(2x_1) + \cos(2x_3) + \frac{1}{2}(\sin(3x_1) + \cos(3x_3)) + \frac{1}{4}(\sin(5x_1) + \cos(5x_3)),$$

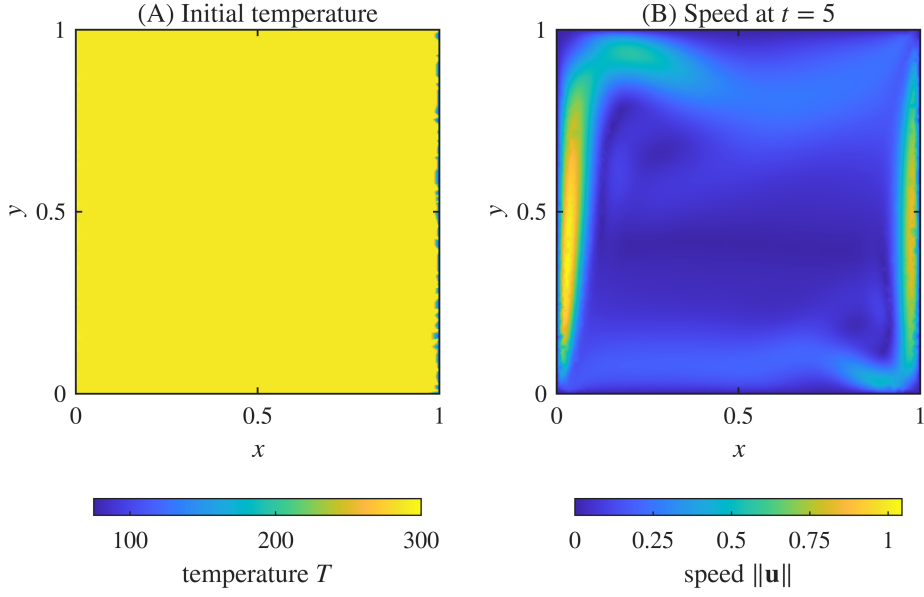


Figure 16: Buoyancy-driven cavity. (A) Initial temperature. (B) Speed at $t = 5$.

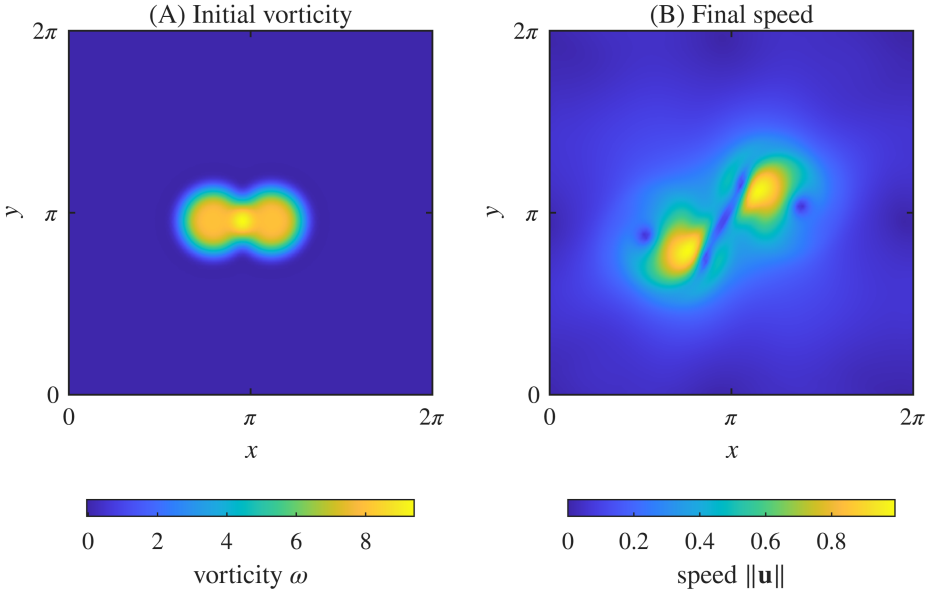


Figure 17: Merging vortices. (A) Initial vorticity. (B) Speed at $t = 0.4$.

$$u_z(x, 0) = \sin(2x_2) + \cos(2x_1) + \frac{1}{2}(\sin(3x_2) + \cos(3x_1)) + \frac{1}{4}(\sin(5x_2) + \cos(5x_1)), \quad (87)$$

and is driven by the steady divergence-free forcing

$$\mathbf{f}(x) = \begin{bmatrix} c_1(\sin(q_1 x_3) + \cos(q_2 x_2)) \\ c_2(\sin(q_1 x_1) + \cos(q_2 x_3)) \\ c_3(\sin(q_1 x_2) + \cos(q_2 x_1)) \end{bmatrix}, \quad (88)$$

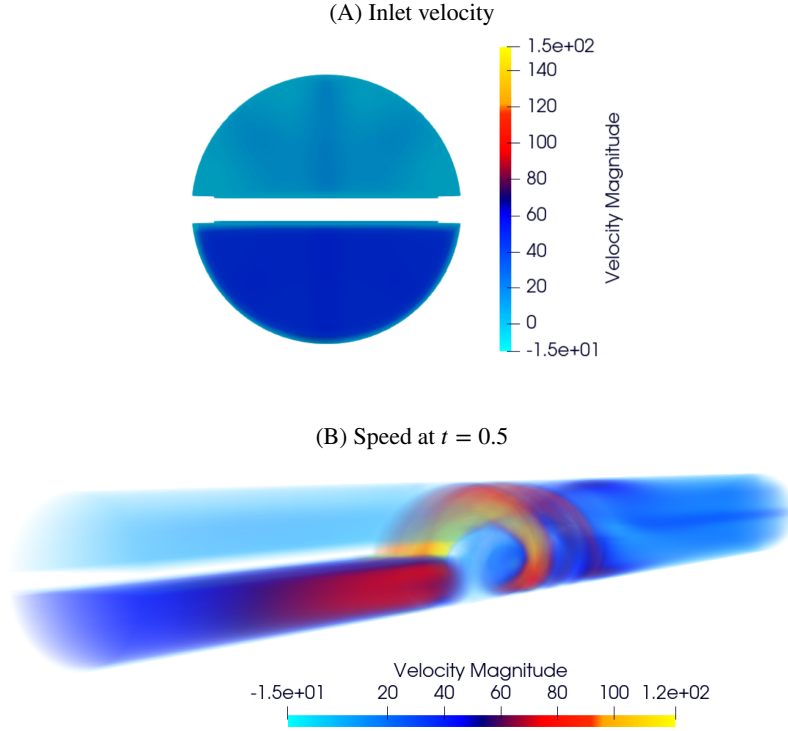


Figure 18: Turbulent species transport. (A) Inlet velocity on the plane $z = 0$. (B) Speed at $t = 0.5$ on the whole domain.

whose amplitudes c_1, c_2, c_3 are drawn from a normal distribution with standard deviation 0.15 truncated to $[-0.3, 0.3]$, and whose distinct wavenumbers q_1, q_2 are drawn from $\{1, 2, 3\}$. We learn the map from the sampled forcing to the velocity at $t = 0.5$ on 7000 Fekete points of the grid.

D.2. Manufactured tests of the output reconstruction

D.2.1. Domains and fields

The convergence tests of Section 5.1 use the domains $\Omega_d = \{F_d > 0\}$ with

$$F_2(x, y) = 1 - x^2 - 1.4y^2 - 0.15x^4 - 0.10x^2y^2 - 0.20y^4 + 0.10x^3, \quad (89)$$

$$F_3(x, y, z) = 1 - x^2 - 1.3y^2 - 1.7z^2 - 0.10(x^4 + y^4 + z^4) - 0.08x^2y^2 - 0.06y^2z^2 + 0.10xyz. \quad (90)$$

In 2D, the field is $v = (\partial_y \psi, -\partial_x \psi)$ with

$$\psi = F_2^2 [\sin(\pi x) \sin(2\pi y) + 0.2 \cos(3\pi x) \sin(\pi y)], \quad (91)$$

and in 3D, $v = \nabla \times A$ with

$$A = F_3^2 (\sin(\pi y) \sin(2\pi z), \sin(\pi z) \sin(2\pi x), \sin(\pi x) \sin(2\pi y))^T. \quad (92)$$

Both fields are divergence free and vanish on the boundary.

The node generator of KernelPack-JAX [8] produced four node sets in each domain, at spacings 0.103, 0.0675, 0.0445, and 0.0295 in 2D (172 to 1907 nodes) and 0.22, 0.16, 0.118, and 0.088 in 3D (197 to 2570 nodes). The errors

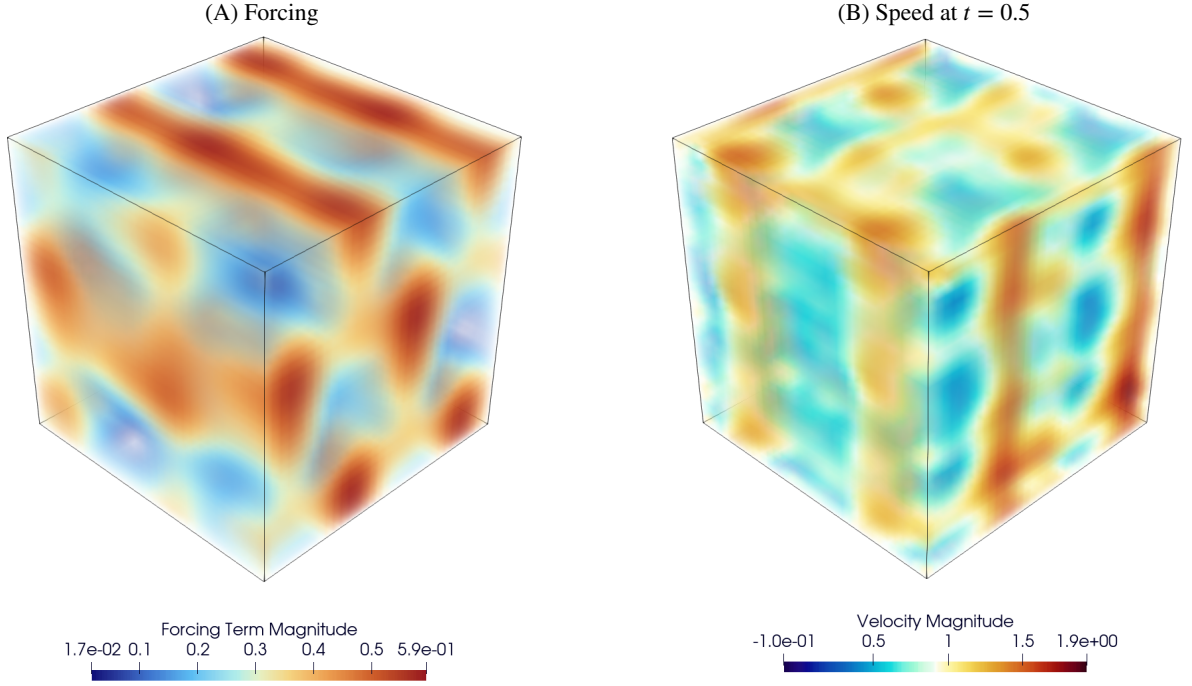


Figure 19: Forced isotropic turbulence. (A) Forcing magnitude. (B) Speed at $t = 0.5$.

use fixed sets of 4096 (2D) and 32768 (3D) uniformly random interior points. For degrees $\varpi = 1, 3, 5$, the divergence-free polynomial spaces have dimensions $n_P = 5, 14, 27$ in 2D and 11, 50, 133 in 3D, and every stencil keeps the full space. At the finest level, the sixth-order errors are about 4.5×10^{-6} in 2D and 1.6×10^{-3} in 3D for both kernels.

D.2.2. Divergence and periodicity

On the evaluation points where the output reconstruction is differentiable (i.e., away from the centers of the cubic kernel), we measure the divergence relative to the size of the velocity gradient,

$$D_{\text{rms}} = \frac{(|Q_D|^{-1} \sum_{y \in Q_D} |\nabla \cdot v_h(y)|^2)^{1/2}}{\max\{1, (|Q_D|^{-1} \sum_{y \in Q_D} \|Dv_h(y)\|_F^2)^{1/2}\}}, \quad (93)$$

with the derivative of each local expansion taken on its own stencil. Its largest value over all smooth tests is 1.5×10^{-13} . The periodic tests on $[0, 2\pi)^d$ use the potentials

$$\psi = \sin x \cos(2y) + 0.2 \cos(3x) \sin y, \quad d = 2, \quad (94)$$

$$A = (\sin y \sin(2z), \sin z \sin(2x), \sin x \sin(2y))^T, \quad d = 3, \quad (95)$$

on 657 (2D) and 1042 (3D) scattered nodes. We compare the reconstructed velocities and Jacobians at points that differ by integer multiples of the period, on the faces, edges, and corners of the box and at 128 interior points. The largest differences are 1.4×10^{-12} for the velocities and 1.6×10^{-12} for the Jacobians.

D.2.3. The turbulent ensemble

The statistical test of Section 5.1 uses the solenoidal Fourier ensemble

$$v(y) = \sum_{\mathbf{k} \in \mathcal{K}} [\mathbf{c}_{\mathbf{k}} \cos(\mathbf{k} \cdot y) + \mathbf{s}_{\mathbf{k}} \sin(\mathbf{k} \cdot y)], \quad \mathbb{E}[\mathbf{c}_{\mathbf{k}} \mathbf{c}_{\mathbf{k}}^T] = \mathbb{E}[\mathbf{s}_{\mathbf{k}} \mathbf{s}_{\mathbf{k}}^T] = w_{\mathbf{k}} \left(I - \frac{\mathbf{k} \mathbf{k}^T}{\|\mathbf{k}\|_2^2} \right), \quad (96)$$

with independent Gaussian coefficients perpendicular to their wave vectors, 1514 modes up to wavenumber 1024, and weights $w_{\mathbf{k}}$ that integrate a $\|\mathbf{k}\|_2^{-5/3}$ energy spectrum over radial shells, normalized so that $2 \sum_{\mathbf{k}} w_{\mathbf{k}} = 1$. Its covariance is divergence free, and its structure function approximately follows the Kolmogorov law over the separations $0.03 \leq r \leq 0.8$, which we fixed before any output reconstruction. At smaller separations, the finite bandwidth makes the increments smooth. We restrict the ensemble to Ω_3 and reconstruct it on six node sets with spacings 0.26, 0.19, 0.14, 0.101, 0.075, and 0.05 (128 to 13567 nodes). The structure functions (70) use 32 separations from 10^{-5} to 0.8 in eight fixed directions and are computed exactly from the covariance, not from samples. For degree five, $E_{S_2}(0.8)$ falls from 14% on the coarsest set to 4.6% on the finest, while the worst error over the whole range remains 64%.

For the comparison of local and global output reconstruction, we use the node set with 1770 nodes and one fixed five-scale Matérn kernel for both, with the same polynomials. The global system uses the covariance anchored at a point a , $K(x, y) - K(x, a) - K(a, y) + K(a, a)$, which is positive definite and gives the same interpolant, since the constant polynomials absorb the difference. For degree five, the errors $E_{S_2}(0.8)$ are 9.9% (local) and 10.6% (global), the normalized mean square errors of the increments are 0.168 and 0.166, and the worst errors over the whole range are about 87% for both.

D.3. Field figures

The field figures show the first simulation of each data set and are drawn by the MATLAB scripts in our repository. In two dimensions, the speed $\|\mathbf{u}\|_2$ is interpolated linearly from the samples. Panels of the same quantity share a color scale, and the transverse direction of the backward-facing step is expanded.

D.4. Direct and Lagrange evaluation

Table 6 evaluates seven unaugmented operator models for each data set and each $N \in \{500, 1000, 5000, 10000\}$: the fixed-shape kernel at the radii of the five prescribed densities of Section 3.6 (from 10% to 90%, with $\theta = 10^{-6}$) and at its validated radius and regularization, and the fitted-shape kernel at its validated parameters. We rebuild each model from its selected parameters and check that its Lagrange predictions at the nodes agree with those of Section 5.2 (largest relative difference 1.0×10^{-10}). Each model is then evaluated through its coefficients and through its Lagrange weights, with the same normalization, factorization (cuDSS with nested dissection), support, and regularization, and with the same spatial basis. With the local output reconstruction, direct evaluation solves the kernel system for the coefficients of all stencils, in blocks of stencils, and Lagrange evaluation applies the weights to the same coefficients. The global output reconstruction is that of Appendix A.5, on the nodes. The test errors are measured at the nodes on the 100 test fields.

Table 8

Test error at the nodes (mean relative velocity error over the 100 test inputs) of direct and Lagrange evaluation of the validated unaugmented fixed-shape model of the lid-driven cavity with global output reconstruction, of direct evaluation applied to the sampled velocities instead of the coefficients, and the Frobenius norm of the spatial coefficients of the training fields. Bold: smallest test error in each row.

N	Direct	Lagrange	Direct/Lagrange	Direct on samples	$\ B\ _F$
500	1.54×10^{-5}	1.20×10^{-5}	1.28	1.20×10^{-5}	1.58×10^9
1000	8.39×10^{-6}	4.25×10^{-6}	1.97	4.24×10^{-6}	2.24×10^9
5000	3.19×10^{-6}	9.54×10^{-7}	3.34	9.32×10^{-7}	5.00×10^9
10000	3.42×10^{-6}	6.64×10^{-7}	5.14	6.42×10^{-7}	7.08×10^9

Table 8 follows the validated fixed-shape model of the lid-driven cavity with global output reconstruction as N grows. The Frobenius norm of the spatial coefficients of the training fields grows from 1.6×10^9 to 7.1×10^9 , which shows the cancellation that Theorem 3 warns about, and the ratio of the direct to the Lagrange error grows from 1.3 to 5.1. Direct evaluation applied to the sampled velocities instead of the coefficients is as accurate as Lagrange evaluation, so the gain arises in the output reconstruction, through the large coefficients.

D.5. Software

Our implementation and the scripts of this paper are part of KernelPack-JAX (<https://github.com/VarShankar/kernelpack-jax>). The operator models of Section 5.2 were fitted with `scripts/run_fastol_`

`gpu_campaign.py`. The comparisons of Section 5.1.4, Section 5.3, and Appendices A.5 and D.4 rebuild these models with the released code, whose predictions agree with those of the original fits to 1.0×10^{-10} (Appendix D.4), and run on the same GPU with `scripts/fastol_gpu_evaluation_study.py`. The directory `FastOLPaper` lists the scripts that produce every result and holds the scripts that build the tables and figures from them.

D.6. All operator benchmarks

This section collects the results of all ten benchmarks for every training size. Figures 20 and 23 complete Figures 8 and 10, Figures 21 and 22 complete Figures 7 and 9, and Table 9 gives the rates at which the errors decrease with N . Tables 10 and 11 collect the errors, storage, and prediction times at $N = 10000$, and Tables 12–21 list all 160 fitted models. Two entries show why we report storage separately from density. The fixed-shape and fitted-shape augmented species models have densities of 9.1% and 21.7% but 26 and 35 million factor entries, and the factor of the backward-facing step, at 33% density, has 38 million entries, three quarters of a dense one. All cost columns use the nested-dissection ordering of Section 3.5, measured by a replay of the selected models (`scripts/fastol_gpu_cost_replay.py`). The test errors were computed with the approximate-minimum-degree ordering, and the predictions with the two orderings agree to 1.2×10^{-10} .

Table 9

Least-squares exponent τ of the test error $\propto N^{-\tau}$ over $N \in \{500, 1000, 5000, 10000\}$. Bold: fastest decay in each row.

Data set	Fixed	Fitted	Fixed + affine	Fitted + affine
Forced isotropic turbulence	0.68	0.69	0.76	0.66
Turbulent species transport	0.85	1.09	1.03	1.26
Merging vortices	0.77	0.73	0.57	0.53
Lid-driven cavity	0.97	1.09	0.96	1.08
Cylinder flow (no shedding)	0.91	0.87	0.85	0.84
Cylinder flow (vortex shedding)	0.54	0.53	0.44	0.47
Backward-facing step	0.39	0.42	0.14	0.15
Buoyancy-driven cavity	0.82	0.91	0.92	0.96
Taylor–Green vortices (space)	−0.56	−1.06	2.81	3.24
Taylor–Green vortices (spacetime)	−0.58	−1.03	2.79	3.22

Table 10

Test error (mean relative velocity error of the predictions for the 100 test inputs) at $N = 10000$. The two augmented models use the same affine columns. Each model is fitted and validated separately, and all four use the same local output reconstruction. Both Taylor–Green cases use the amplitude normalization (72). Bold: smallest test error in each row.

Data set	Fixed	Fitted	Fixed + affine	Fitted + affine
Forced isotropic turbulence	2.80×10^{-3}	1.62×10^{-3}	9.77×10^{-5}	7.14×10^{-5}
Turbulent species transport	4.34×10^{-4}	6.86×10^{-5}	1.89×10^{-4}	3.69×10^{-5}
Merging vortices	3.02×10^{-7}	3.07×10^{-7}	3.39×10^{-7}	3.19×10^{-7}
Lid-driven cavity	6.42×10^{-7}	4.76×10^{-7}	6.42×10^{-7}	4.72×10^{-7}
Cylinder flow (no shedding)	2.02×10^{-7}	1.94×10^{-7}	1.70×10^{-7}	1.73×10^{-7}
Cylinder flow (vortex shedding)	3.19×10^{-7}	2.95×10^{-7}	3.19×10^{-7}	2.96×10^{-7}
Taylor–Green vortices (space)	2.82×10^{-6}	2.11×10^{-6}	2.42×10^{-10}	5.22×10^{-11}
Taylor–Green vortices (spacetime)	2.46×10^{-6}	1.86×10^{-6}	2.18×10^{-10}	4.70×10^{-11}
Backward-facing step	6.33×10^{-8}	6.33×10^{-8}	5.86×10^{-8}	5.86×10^{-8}
Buoyancy-driven cavity	5.38×10^{-6}	4.47×10^{-6}	5.38×10^{-6}	4.62×10^{-6}

D.6.1. Offline cost

Table 22 gives the offline costs at $N = 10000$. Fitting and validating a fitted-shape model takes 11–27 minutes, against 2–7 minutes for a fixed-shape model. Once the parameters are chosen, rebuilding the factorization takes about one to five seconds for most benchmarks and about 43 seconds for the dense forced-turbulence models. The setup of

Local and Sparse Kernel Operator Learning

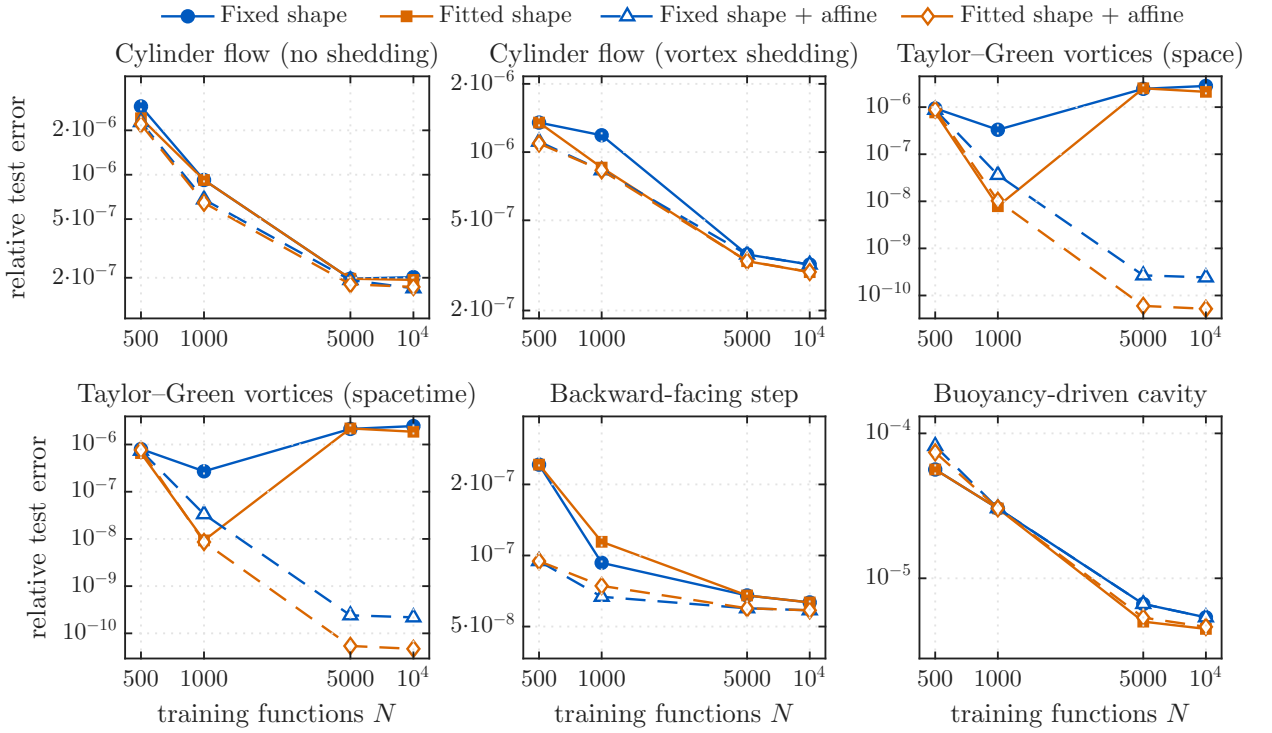


Figure 20: Test error of the other six benchmarks. Dashed: with affine augmentation.

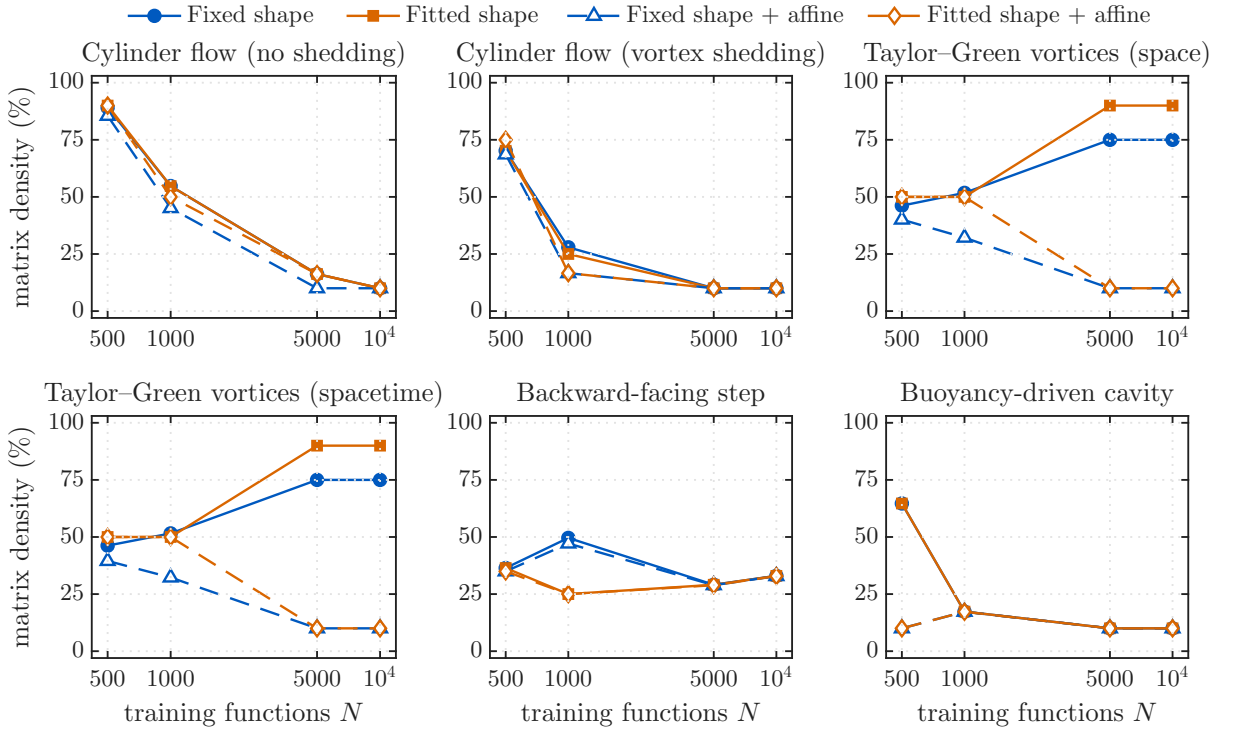


Figure 21: Selected density of the other six benchmarks. With augmentation, the Taylor-Green models select about 10% density at the two largest N , against 75% or 90% without it.

Local and Sparse Kernel Operator Learning

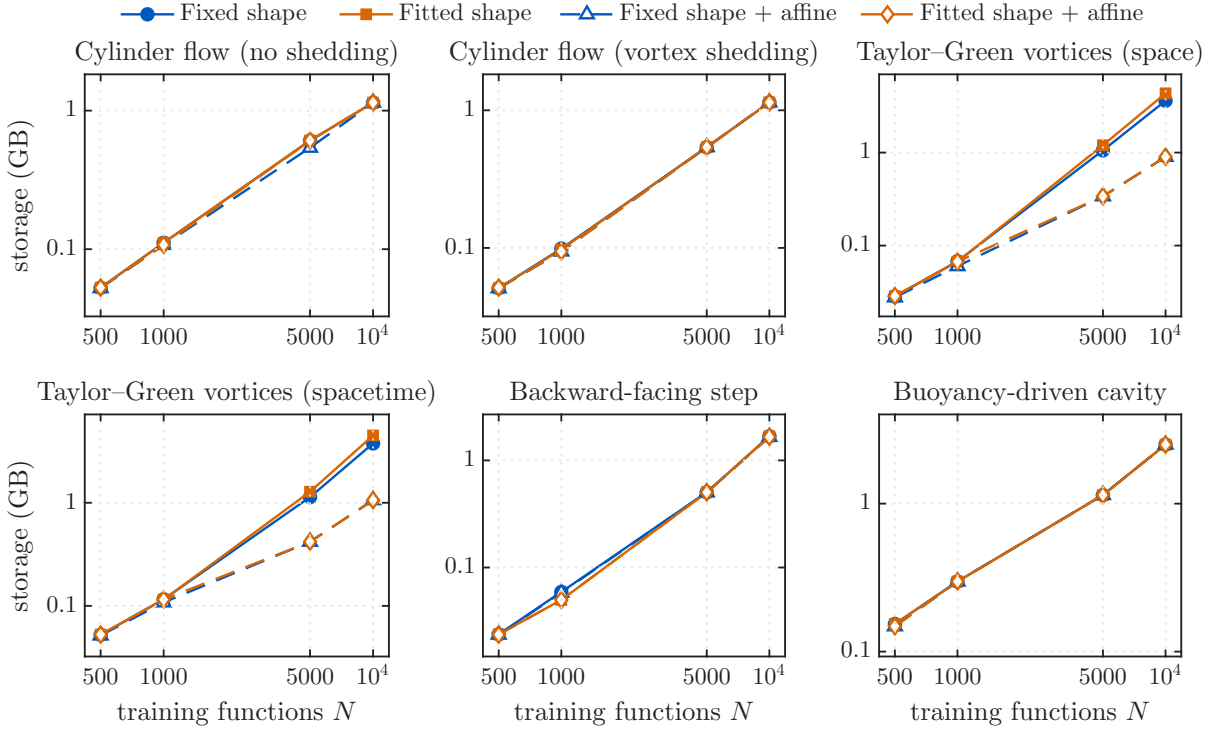


Figure 22: Accounted storage of the other six benchmarks (Appendix B.3).

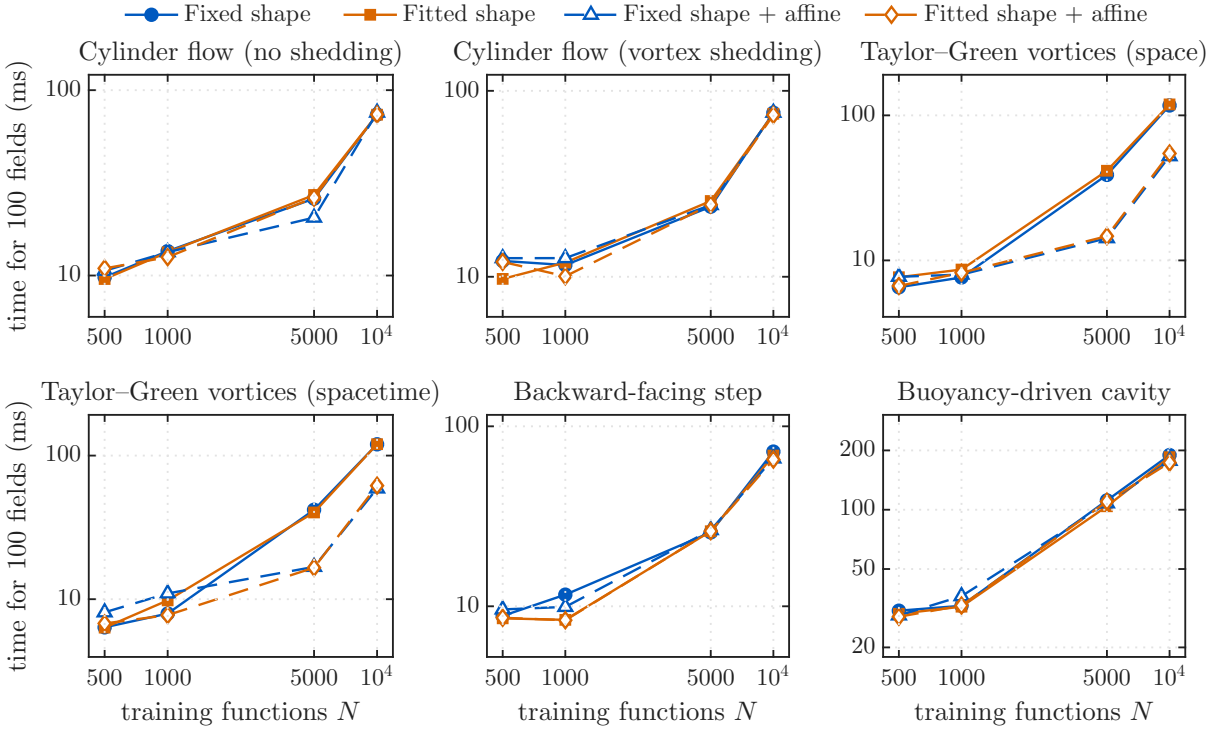


Figure 23: Median warm time to predict all 100 test fields for the other six benchmarks.

Table 11

Accounted storage in decimal GB / median warm prediction time in ms for 100 fields at $N = 10000$. Storage includes the stored model arrays, the solve buffers, and cuDSS's estimate of the memory of its factors. It is not measured peak GPU memory. Times include operator Lagrange evaluation and local output reconstruction. Bold: smallest storage and smallest time in each row.

Data set	Fixed	Fitted	Fixed + affine	Fitted + affine
Forced isotropic turbulence	10.680 / 235.2	10.680 / 234.0	10.685 / 244.1	10.682 / 233.4
Turbulent species transport	2.662 / 126.6	3.130 / 126.0	2.630 / 123.3	3.137 / 137.4
Merging vortices	1.397 / 79.6	1.029 / 62.4	0.772 / 37.7	0.767 / 39.0
Lid-driven cavity	1.135 / 75.9	1.135 / 71.5	1.136 / 70.2	1.136 / 70.3
Cylinder flow (no shedding)	1.141 / 73.8	1.141 / 73.7	1.142 / 75.7	1.142 / 74.0
Cylinder flow (vortex shedding)	1.140 / 76.2	1.140 / 75.3	1.141 / 76.2	1.141 / 74.2
Taylor–Green vortices (space)	3.606 / 117.0	4.334 / 119.0	0.899 / 52.3	0.899 / 54.6
Taylor–Green vortices (spacetime)	3.766 / 119.8	4.494 / 120.1	1.059 / 59.0	1.059 / 61.7
Backward-facing step	1.679 / 72.4	1.679 / 68.8	1.657 / 66.3	1.657 / 65.3
Buoyancy-driven cavity	2.504 / 189.4	2.504 / 184.9	2.505 / 177.0	2.505 / 173.4

Table 12

Forced isotropic turbulence: all models. Test error is the mean relative velocity error of the predictions for the 100 test inputs. Density is $d_K = \text{nnz}(K_{UU})/N^2$ in percent. Factor is the number of entries of the cuDSS Cholesky factor in millions. GB is the accounted storage and ms the median warm time for all 100 fields. Bold: smallest test error at each N .

N	Operator model	Test error	Density (%)	Factor (10^6)	GB	ms
500	Fixed shape	0.0214	90.00	0.125	0.465	60.9
500	Fitted shape	0.013	90.00	0.125	0.465	64.3
500	Fixed shape + affine	9.37×10^{-4}	90.00	0.125	0.467	58.5
500	Fitted shape + affine	4.90×10^{-4}	90.00	0.126	0.467	56.5
1000	Fixed shape	0.0157	90.00	0.990	0.750	64.8
1000	Fitted shape	0.0139	90.00	0.991	0.750	70.2
1000	Fixed shape + affine	6.05×10^{-4}	90.00	0.990	0.752	65.2
1000	Fitted shape + affine	3.43×10^{-4}	90.00	0.991	0.752	65.5
5000	Fixed shape	5.16×10^{-3}	90.00	24.722	4.390	126.8
5000	Fitted shape	4.69×10^{-3}	90.00	24.748	4.390	127.2
5000	Fixed shape + affine	1.67×10^{-4}	90.00	24.722	4.394	126.9
5000	Fitted shape + affine	1.10×10^{-4}	90.00	24.751	4.394	126.1
10000	Fixed shape	2.80×10^{-3}	90.00	98.899	10.680	235.2
10000	Fitted shape	1.62×10^{-3}	90.00	98.984	10.680	234.0
10000	Fixed shape + affine	9.77×10^{-5}	90.00	98.899	10.685	244.1
10000	Fitted shape + affine	7.14×10^{-5}	90.00	98.651	10.682	233.4

the output reconstruction, which the four models share, takes 0.03–2.9 seconds after compilation, and a first setup, which also compiles, takes 1.6–24 seconds.

Table 13

Turbulent species transport: all models. Test error is the mean relative velocity error of the predictions for the 100 test inputs. Density is $d_K = \text{nnz}(K_{UU})/N^2$ in percent. Factor is the number of entries of the cuDSS Cholesky factor in millions. GB is the accounted storage and ms the median warm time for all 100 fields. Bold: smallest test error at each N .

N	Operator model	Test error	Density (%)	Factor (10^6)	GB	ms
500	Fixed shape	4.39×10^{-3}	63.94	0.116	0.226	54.3
500	Fitted shape	2.50×10^{-3}	90.00	0.129	0.228	52.8
500	Fixed shape + affine	3.35×10^{-3}	40.98	0.097	0.224	53.1
500	Fitted shape + affine	2.48×10^{-3}	44.60	0.096	0.224	54.0
1000	Fixed shape	1.25×10^{-3}	89.71	1.000	0.278	57.1
1000	Fitted shape	9.53×10^{-4}	85.62	0.962	0.276	57.8
1000	Fixed shape + affine	1.49×10^{-3}	22.92	0.313	0.250	55.6
1000	Fitted shape + affine	8.75×10^{-4}	75.00	0.988	0.273	59.1
5000	Fixed shape	2.02×10^{-4}	69.06	24.743	1.858	98.2
5000	Fitted shape	2.95×10^{-4}	90.00	24.744	2.036	101.7
5000	Fixed shape + affine	1.96×10^{-4}	69.56	24.689	1.862	98.5
5000	Fitted shape + affine	2.55×10^{-4}	66.71	24.746	1.838	98.3
10000	Fixed shape	4.34×10^{-4}	10.40	24.431	2.662	126.6
10000	Fitted shape	6.86×10^{-5}	21.95	33.613	3.130	126.0
10000	Fixed shape + affine	1.89×10^{-4}	9.12	25.740	2.630	123.3
10000	Fitted shape + affine	3.69×10^{-5}	21.71	35.385	3.137	137.4

Table 14

Merging vortices: all models. Test error is the mean relative velocity error of the predictions for the 100 test inputs. Density is $d_K = \text{nnz}(K_{UU})/N^2$ in percent. Factor is the number of entries of the cuDSS Cholesky factor in millions. GB is the accounted storage and ms the median warm time for all 100 fields. Bold: smallest test error at each N .

N	Operator model	Test error	Density (%)	Factor (10^6)	GB	ms
500	Fixed shape	2.85×10^{-6}	75.06	0.104	0.025	6.0
500	Fitted shape	2.51×10^{-6}	75.00	0.106	0.025	6.8
500	Fixed shape + affine	1.94×10^{-6}	37.36	0.065	0.021	7.9
500	Fitted shape + affine	1.66×10^{-6}	33.64	0.060	0.021	6.9
1000	Fixed shape	1.41×10^{-6}	63.81	0.386	0.060	7.3
1000	Fitted shape	1.45×10^{-6}	60.31	0.387	0.058	8.6
1000	Fixed shape + affine	8.60×10^{-7}	29.73	0.225	0.047	9.1
1000	Fitted shape + affine	8.46×10^{-7}	29.04	0.217	0.047	7.4
5000	Fixed shape	3.54×10^{-7}	24.58	6.404	0.433	24.9
5000	Fitted shape	3.91×10^{-7}	25.00	6.356	0.436	25.2
5000	Fixed shape + affine	3.66×10^{-7}	24.58	6.404	0.434	24.9
5000	Fitted shape + affine	4.08×10^{-7}	18.33	4.892	0.369	21.7
10000	Fixed shape	3.02×10^{-7}	24.67	26.353	1.397	79.6
10000	Fitted shape	3.07×10^{-7}	15.85	18.009	1.029	62.4
10000	Fixed shape + affine	3.39×10^{-7}	10.00	10.703	0.772	37.7
10000	Fitted shape + affine	3.19×10^{-7}	10.00	10.047	0.767	39.0

Table 15

Lid-driven cavity: all models. Test error is the mean relative velocity error of the predictions for the 100 test inputs. Density is $d_K = \text{nnz}(K_{UU})/N^2$ in percent. Factor is the number of entries of the cuDSS Cholesky factor in millions. GB is the accounted storage and ms the median warm time for all 100 fields. Bold: smallest test error at each N .

N	Operator model	Test error	Density (%)	Factor (10^6)	GB	ms
500	Fixed shape	1.20×10^{-5}	22.85	0.053	0.046	9.2
500	Fitted shape	1.20×10^{-5}	22.85	0.053	0.046	10.8
500	Fixed shape + affine	1.15×10^{-5}	21.49	0.048	0.046	10.4
500	Fitted shape + affine	1.15×10^{-5}	25.00	0.055	0.047	9.2
1000	Fixed shape	4.24×10^{-6}	19.99	0.190	0.096	10.6
1000	Fitted shape	4.11×10^{-6}	25.00	0.223	0.098	11.4
1000	Fixed shape + affine	4.13×10^{-6}	20.00	0.189	0.096	10.8
1000	Fitted shape + affine	4.10×10^{-6}	25.00	0.223	0.098	11.1
5000	Fixed shape	9.32×10^{-7}	10.00	2.600	0.538	34.4
5000	Fitted shape	6.73×10^{-7}	10.00	2.600	0.538	34.6
5000	Fixed shape + affine	9.32×10^{-7}	10.00	2.600	0.539	34.2
5000	Fitted shape + affine	6.80×10^{-7}	10.00	2.600	0.539	34.7
10000	Fixed shape	6.42×10^{-7}	10.00	11.198	1.135	75.9
10000	Fitted shape	4.76×10^{-7}	10.00	11.198	1.135	71.5
10000	Fixed shape + affine	6.42×10^{-7}	10.00	11.198	1.136	70.2
10000	Fitted shape + affine	4.72×10^{-7}	10.00	11.198	1.136	70.3

Table 16

Cylinder flow (no shedding): all models. Test error is the mean relative velocity error of the predictions for the 100 test inputs. Density is $d_K = \text{nnz}(K_{UU})/N^2$ in percent. Factor is the number of entries of the cuDSS Cholesky factor in millions. GB is the accounted storage and ms the median warm time for all 100 fields. Bold: smallest test error at each N .

N	Operator model	Test error	Density (%)	Factor (10^6)	GB	ms
500	Fixed shape	2.91×10^{-6}	89.11	0.129	0.053	9.8
500	Fitted shape	2.42×10^{-6}	90.00	0.128	0.053	9.6
500	Fixed shape + affine	2.27×10^{-6}	85.52	0.128	0.052	10.6
500	Fitted shape + affine	2.20×10^{-6}	90.00	0.128	0.053	10.9
1000	Fixed shape	9.21×10^{-7}	54.74	0.674	0.111	13.5
1000	Fitted shape	9.21×10^{-7}	54.74	0.674	0.111	13.3
1000	Fixed shape + affine	6.80×10^{-7}	45.10	0.581	0.107	13.4
1000	Fitted shape + affine	6.45×10^{-7}	50.00	0.394	0.107	12.6
5000	Fixed shape	1.97×10^{-7}	16.22	4.555	0.607	26.0
5000	Fitted shape	1.97×10^{-7}	16.22	4.555	0.607	27.2
5000	Fixed shape + affine	1.93×10^{-7}	10.00	2.726	0.540	20.5
5000	Fitted shape + affine	1.81×10^{-7}	16.22	4.555	0.608	26.3
10000	Fixed shape	2.02×10^{-7}	10.00	11.981	1.141	73.8
10000	Fitted shape	1.94×10^{-7}	10.00	11.981	1.141	73.7
10000	Fixed shape + affine	1.70×10^{-7}	10.00	11.981	1.142	75.7
10000	Fitted shape + affine	1.73×10^{-7}	10.00	11.981	1.142	74.0

Table 17

Cylinder flow (vortex shedding): all models. Test error is the mean relative velocity error of the predictions for the 100 test inputs. Density is $d_K = \text{nnz}(K_{UU})/N^2$ in percent. Factor is the number of entries of the cuDSS Cholesky factor in millions. GB is the accounted storage and ms the median warm time for all 100 fields. Bold: smallest test error at each N .

N	Operator model	Test error	Density (%)	Factor (10^6)	GB	ms
500	Fixed shape	1.35×10^{-6}	70.21	0.122	0.051	12.2
500	Fitted shape	1.35×10^{-6}	70.21	0.122	0.051	9.7
500	Fixed shape + affine	1.11×10^{-6}	68.61	0.122	0.051	12.6
500	Fitted shape + affine	1.09×10^{-6}	75.00	0.125	0.051	12.0
1000	Fixed shape	1.19×10^{-6}	27.92	0.262	0.099	11.6
1000	Fitted shape	8.54×10^{-7}	25.00	0.231	0.098	11.9
1000	Fixed shape + affine	8.31×10^{-7}	16.61	0.164	0.094	12.6
1000	Fitted shape + affine	8.31×10^{-7}	16.61	0.164	0.094	10.1
5000	Fixed shape	3.53×10^{-7}	10.00	2.710	0.539	23.8
5000	Fitted shape	3.30×10^{-7}	10.00	2.710	0.539	25.6
5000	Fixed shape + affine	3.53×10^{-7}	10.00	2.710	0.540	24.3
5000	Fitted shape + affine	3.31×10^{-7}	10.00	2.710	0.540	24.4
10000	Fixed shape	3.19×10^{-7}	10.00	11.865	1.140	76.2
10000	Fitted shape	2.95×10^{-7}	10.00	11.865	1.140	75.3
10000	Fixed shape + affine	3.19×10^{-7}	10.00	11.865	1.141	76.2
10000	Fitted shape + affine	2.96×10^{-7}	10.00	11.865	1.141	74.2

Table 18

Taylor–Green vortices (space): all models. Test error is the mean relative velocity error of the predictions for the 100 test inputs. Density is $d_K = \text{nnz}(K_{UU})/N^2$ in percent. Factor is the number of entries of the cuDSS Cholesky factor in millions. GB is the accounted storage and ms the median warm time for all 100 fields. Bold: smallest test error at each N .

N	Operator model	Test error	Density (%)	Factor (10^6)	GB	ms
500	Fixed shape	9.41×10^{-7}	46.17	0.094	0.028	6.5
500	Fitted shape	7.72×10^{-7}	50.00	0.097	0.029	7.7
500	Fixed shape + affine	8.76×10^{-7}	40.07	0.083	0.028	7.7
500	Fitted shape + affine	8.83×10^{-7}	50.00	0.097	0.029	6.7
1000	Fixed shape	3.31×10^{-7}	51.76	0.422	0.068	7.6
1000	Fitted shape	7.86×10^{-9}	50.00	0.414	0.067	8.6
1000	Fixed shape + affine	3.63×10^{-8}	32.16	0.288	0.060	8.0
1000	Fitted shape + affine	1.03×10^{-8}	50.00	0.414	0.067	8.2
5000	Fixed shape	2.45×10^{-6}	75.00	22.923	1.056	38.9
5000	Fitted shape	2.51×10^{-6}	90.00	24.889	1.199	41.5
5000	Fixed shape + affine	2.67×10^{-10}	10.00	2.644	0.339	14.2
5000	Fitted shape + affine	6.00×10^{-11}	10.00	2.644	0.339	14.7
10000	Fixed shape	2.82×10^{-6}	75.00	72.677	3.606	117.0
10000	Fitted shape	2.11×10^{-6}	90.00	99.800	4.334	119.0
10000	Fixed shape + affine	2.42×10^{-10}	10.00	11.728	0.899	52.3
10000	Fitted shape + affine	5.22×10^{-11}	10.00	11.728	0.899	54.6

Table 19

Taylor–Green vortices (spacetime): all models. Test error is the mean relative velocity error of the predictions for the 100 test inputs. Density is $d_K = \text{nnz}(K_{UU})/N^2$ in percent. Factor is the number of entries of the cuDSS Cholesky factor in millions. GB is the accounted storage and ms the median warm time for all 100 fields. Bold: smallest test error at each N .

N	Operator model	Test error	Density (%)	Factor (10^6)	GB	ms
500	Fixed shape	8.01×10^{-7}	46.23	0.093	0.052	6.4
500	Fitted shape	6.58×10^{-7}	50.00	0.097	0.053	6.3
500	Fixed shape + affine	7.38×10^{-7}	39.51	0.082	0.052	8.1
500	Fitted shape + affine	7.65×10^{-7}	50.00	0.097	0.053	6.8
1000	Fixed shape	2.71×10^{-7}	51.58	0.419	0.116	7.9
1000	Fitted shape	9.30×10^{-9}	50.00	0.414	0.115	9.8
1000	Fixed shape + affine	3.34×10^{-8}	32.27	0.284	0.108	11.0
1000	Fitted shape + affine	8.61×10^{-9}	50.00	0.414	0.115	7.7
5000	Fixed shape	2.14×10^{-6}	75.00	22.923	1.136	41.9
5000	Fitted shape	2.19×10^{-6}	90.00	24.889	1.279	40.2
5000	Fixed shape + affine	2.41×10^{-10}	10.00	2.644	0.419	16.8
5000	Fitted shape + affine	5.40×10^{-11}	10.00	2.644	0.419	16.6
10000	Fixed shape	2.46×10^{-6}	75.00	72.677	3.766	119.8
10000	Fitted shape	1.86×10^{-6}	90.00	99.800	4.494	120.1
10000	Fixed shape + affine	2.18×10^{-10}	10.00	11.728	1.059	59.0
10000	Fitted shape + affine	4.70×10^{-11}	10.00	11.728	1.059	61.7

Table 20

Backward-facing step: all models. Test error is the mean relative velocity error of the predictions for the 100 test inputs. Density is $d_K = \text{nnz}(K_{UU})/N^2$ in percent. Factor is the number of entries of the cuDSS Cholesky factor in millions. GB is the accounted storage and ms the median warm time for all 100 fields. Bold: smallest test error at each N .

N	Operator model	Test error	Density (%)	Factor (10^6)	GB	ms
500	Fixed shape	2.43×10^{-7}	36.41	0.073	0.024	8.8
500	Fitted shape	2.43×10^{-7}	36.41	0.073	0.024	8.6
500	Fixed shape + affine	9.46×10^{-8}	34.94	0.068	0.024	9.6
500	Fitted shape + affine	9.46×10^{-8}	34.94	0.068	0.024	8.6
1000	Fixed shape	9.31×10^{-8}	49.59	0.350	0.059	11.6
1000	Fitted shape	1.14×10^{-7}	25.00	0.236	0.050	8.4
1000	Fixed shape + affine	6.68×10^{-8}	47.15	0.342	0.058	9.9
1000	Fitted shape + affine	7.44×10^{-8}	25.00	0.236	0.050	8.4
5000	Fixed shape	6.77×10^{-8}	29.08	7.711	0.504	25.8
5000	Fitted shape	6.77×10^{-8}	29.08	7.711	0.504	26.1
5000	Fixed shape + affine	5.98×10^{-8}	28.90	7.785	0.503	26.5
5000	Fitted shape + affine	5.98×10^{-8}	28.90	7.785	0.503	26.2
10000	Fixed shape	6.33×10^{-8}	32.98	40.506	1.679	72.4
10000	Fitted shape	6.33×10^{-8}	32.98	40.506	1.679	68.8
10000	Fixed shape + affine	5.86×10^{-8}	32.88	38.228	1.657	66.3
10000	Fitted shape + affine	5.86×10^{-8}	32.88	38.228	1.657	65.3

Table 21

Buoyancy-driven cavity: all models. Test error is the mean relative velocity error of the predictions for the 100 test inputs. Density is $d_K = \text{nnz}(K_{UU})/N^2$ in percent. Factor is the number of entries of the cuDSS Cholesky factor in millions. GB is the accounted storage and ms the median warm time for all 100 fields. Bold: smallest test error at each N .

N	Operator model	Test error	Density (%)	Factor (10^6)	GB	ms
500	Fixed shape	5.63×10^{-5}	64.68	0.118	0.154	30.7
500	Fitted shape	5.63×10^{-5}	64.68	0.118	0.154	29.6
500	Fixed shape + affine	8.18×10^{-5}	10.00	0.024	0.148	29.1
500	Fitted shape + affine	7.39×10^{-5}	10.00	0.024	0.148	28.6
1000	Fixed shape	3.03×10^{-5}	17.33	0.170	0.298	32.5
1000	Fitted shape	3.03×10^{-5}	17.33	0.170	0.298	32.2
1000	Fixed shape + affine	3.04×10^{-5}	17.27	0.166	0.298	36.4
1000	Fitted shape + affine	3.04×10^{-5}	17.27	0.166	0.298	32.6
5000	Fixed shape	6.65×10^{-6}	10.00	2.838	1.143	111.5
5000	Fitted shape	5.02×10^{-6}	10.00	2.838	1.143	104.1
5000	Fixed shape + affine	6.65×10^{-6}	10.00	2.838	1.144	107.7
5000	Fitted shape + affine	5.35×10^{-6}	10.00	2.838	1.144	109.9
10000	Fixed shape	5.38×10^{-6}	10.00	11.947	2.504	189.4
10000	Fitted shape	4.47×10^{-6}	10.00	11.947	2.504	184.9
10000	Fixed shape + affine	5.38×10^{-6}	10.00	11.947	2.505	177.0
10000	Fitted shape + affine	4.62×10^{-6}	10.00	11.947	2.505	173.4

Table 22

Offline and initialization costs at $N = 10000$. Selection includes kernel fitting, where applicable, and validation. Rebuild assembles and factors the selected operator. Initialization includes two synchronized 100-field prediction passes, compilation, and CUDA graph preparation. The four models share the setup of the output reconstruction and may reuse compiled code. The dash marks the selection time of the fixed-shape backward-facing-step model, which contains an external pause. Its raw time is in the released data. Bold: smallest time in each column for each data set.

Data set	Operator model	Selection (min)	Rebuild (s)	Initialization (s)
Forced isotropic turbulence	Fixed shape	3.83	43.27	3.71
	Fitted shape	26.32	41.63	3.28
	Fixed shape + affine	6.45	43.70	4.15
	Fitted shape + affine	27.01	42.77	3.05
Turbulent species transport	Fixed shape	3.24	1.48	4.17
	Fitted shape	25.89	4.64	3.70
	Fixed shape + affine	5.79	1.76	6.05
	Fitted shape + affine	24.01	4.93	3.79
Merging vortices	Fixed shape	4.81	0.91	3.35
	Fitted shape	15.26	2.92	3.20
	Fixed shape + affine	4.94	1.12	4.94
	Fitted shape + affine	11.70	0.81	3.26
Lid-driven cavity	Fixed shape	2.12	0.86	2.37
	Fitted shape	12.97	0.87	2.20
	Fixed shape + affine	3.06	0.91	3.54
	Fitted shape + affine	10.90	0.85	2.26
Cylinder flow (no shedding)	Fixed shape	2.32	0.87	2.58
	Fitted shape	12.77	0.86	2.54
	Fixed shape + affine	4.28	0.92	3.77
	Fitted shape + affine	14.53	0.90	2.50
Cylinder flow (vortex shedding)	Fixed shape	2.12	0.85	2.47
	Fitted shape	13.58	0.85	2.61
	Fixed shape + affine	3.07	0.91	3.43
	Fitted shape + affine	12.86	0.88	2.37
Taylor–Green vortices (space)	Fixed shape	3.41	3.18	3.61
	Fitted shape	12.80	5.54	3.43
	Fixed shape + affine	5.32	1.25	5.47
	Fitted shape + affine	15.30	0.68	3.54
Taylor–Green vortices (spacetime)	Fixed shape	3.49	3.30	3.87
	Fitted shape	16.10	3.60	3.60
	Fixed shape + affine	4.82	0.88	5.22
	Fitted shape + affine	15.47	0.69	3.64
Backward-facing step	Fixed shape	—	0.95	1.48
	Fitted shape	14.23	0.99	1.31
	Fixed shape + affine	3.25	3.27	2.53
	Fitted shape + affine	13.42	0.94	1.36
Buoyancy-driven cavity	Fixed shape	2.11	1.65	2.72
	Fitted shape	17.67	1.48	2.53
	Fixed shape + affine	3.16	1.64	4.06
	Fitted shape + affine	17.00	1.51	2.63

References

- [1] Bach, F.R., Jordan, M.I., 2005. Predictive low-rank decomposition for kernel methods, in: Proceedings of the 22nd International Conference on Machine Learning, pp. 33–40. doi:10.1145/1102351.1102356.
- [2] Batlle, P., Darcy, M., Hosseini, B., Owjadi, H., 2024. Kernel methods are competitive for operator learning. *Journal of Computational Physics* 496, 112549.
- [3] Bayona, V., Flyer, N., Fornberg, B., Barnett, G.A., 2017. On the role of polynomials in RBF-FD approximations: II. Numerical solution of elliptic PDEs. *Journal of Computational Physics* 332, 257–273. doi:10.1016/j.jcp.2016.12.008.
- [4] Belianovich, M., Fasshauer, G.E., Narayan, A., Shankar, V., 2026. A unified framework for efficient kernel and polynomial interpolation. *Journal of Approximation Software* 3. URL: <https://ojs.unito.it/index.php/JAS/article/view/12288>, doi:10.13135/3103-1935/12288.
- [5] Benner, P., Gugercin, S., Willcox, K., 2015. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Review* 57, 483–531. doi:10.1137/130932715.
- [6] Bollig, E.F., Flyer, N., Erlebacher, G., 2012. Solution to PDEs using radial basis function finite-differences (RBF-FD) on multiple GPUs. *Journal of Computational Physics* 231, 7133–7151. doi:10.1016/j.jcp.2012.06.030.
- [7] Boullé, N., Townsend, A., 2024. A mathematical guide to operator learning, in: *Handbook of Numerical Analysis*. Elsevier. volume 25, pp. 83–125.
- [8] Cattaneo, A., Ballard, M.K., Kirby, R.M., Wright, G.B., Shankar, V., 2026. KernelPack-JAX: a meshless differentiable GPU framework for node generation and forward and inverse PDE problems. Submitted to *Journal of Computational Physics*.
- [9] Davis, T.A., 2006. Direct Methods for Sparse Linear Systems. volume 2 of *Fundamentals of Algorithms*. SIAM, Philadelphia, PA. doi:10.1137/1.9780898718881.
- [10] Davydov, O., Schaback, R., 2016. Error bounds for kernel-based numerical differentiation. *Numerische Mathematik* 132, 243–269. doi:10.1007/s00211-015-0722-9.
- [11] Dodu, F., Rabut, C., 2004. Irrotational or divergence-free interpolation. *Numerische Mathematik* 98, 477–498. doi:10.1007/s00211-004-0541-x.
- [12] Drake, K.P., Fuselier, E.J., Wright, G.B., 2021. A partition of unity method for divergence-free or curl-free radial basis function approximation. *SIAM Journal on Scientific Computing* 43, A1950–A1974. doi:10.1137/20M1373505.
- [13] Drineas, P., Mahoney, M.W., 2005. Approximating a Gram matrix for improved kernel-based learning, in: *International Conference on Computational Learning Theory*, pp. 323–337.
- [14] Duchon, J., 1977. Splines minimizing rotation-invariant semi-norms in Sobolev spaces, in: *Constructive Theory of Functions of Several Variables*. Springer, Berlin, Heidelberg. volume 571 of *Lecture Notes in Mathematics*, pp. 85–100. doi:10.1007/BFb0086566.
- [15] Fasshauer, G.E., 2007. Meshfree Approximation Methods with MATLAB. World Scientific.
- [16] Fasshauer, G.E., Ye, Q., 2011. Reproducing kernels of generalized Sobolev spaces via a Green function approach with distributional operators. *Numerische Mathematik* 119, 585–611. doi:10.1007/s00211-011-0391-2.
- [17] Fine, S., Scheinberg, K., 2001. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research* 2, 243–264.
- [18] Flyer, N., Barnett, G.A., Wicker, L.J., 2016a. Enhancing finite differences with radial basis functions: Experiments on the Navier–Stokes equations. *Journal of Computational Physics* 316, 39–62. doi:10.1016/j.jcp.2016.02.078.
- [19] Flyer, N., Fornberg, B., Bayona, V., Barnett, G.A., 2016b. On the role of polynomials in rbf-fd approximations: I. interpolation and accuracy. *Journal of Computational Physics* 321, 21–38.
- [20] Fornberg, B., Flyer, N., 2015a. A Primer on Radial Basis Functions with Applications to the Geosciences. SIAM, Philadelphia, PA. doi:10.1137/1.9781611974041.
- [21] Fornberg, B., Flyer, N., 2015b. Solving PDEs with radial basis functions. *Acta Numerica* 24, 215–258. doi:10.1017/S0962492914000130.
- [22] Fornberg, B., Lehto, E., 2011. Stabilization of RBF-generated finite difference methods for convective PDEs. *Journal of Computational Physics* 230, 2270–2285. doi:10.1016/j.jcp.2010.12.014.
- [23] Furrer, R., Genton, M.G., Nychka, D., 2006. Covariance tapering for interpolation of large spatial datasets. *Journal of Computational and Graphical Statistics* 15, 502–523. doi:10.1198/106186006X132178.
- [24] Fuselier, E., 2008. Sobolev-type approximation rates for divergence-free and curl-free RBF interpolants. *Mathematics of Computation* 77, 1407–1423.
- [25] Fuselier, E.J., Shankar, V., Wright, G.B., 2016. A high-order radial basis function (RBF) leray projection method for the solution of the incompressible unsteady Stokes equations. *Computers & Fluids* 128, 41–52.
- [26] Gittens, A., Mahoney, M.W., 2013. Revisiting the Nyström method for improved large-scale machine learning, in: *International Conference on Machine Learning*, pp. 567–575.
- [27] Gneiting, T., 2002. Compactly supported correlation functions. *Journal of Multivariate Analysis* 83, 493–508. doi:10.1006/jmva.2001.2056.
- [28] Higham, N.J., 2002. Accuracy and Stability of Numerical Algorithms. 2 ed., Society for Industrial and Applied Mathematics. doi:10.1137/1.9780898718027.
- [29] Kadri, H., Duflos, E., Preux, P., Canu, S., Rakotomamonjy, A., Audiffren, J., 2016. Operator-valued kernels for learning from functional response data. *Journal of Machine Learning Research* 17, 1–54.
- [30] Khorrami, M.S., Goyal, P., Mianroodi, J.R., Svendsen, B., Benner, P., Raabe, D., 2024. A physics-encoded fourier neural operator approach for surrogate modeling of divergence-free stress fields in solids. *arXiv preprint arXiv:2408.15408*.
- [31] Kingma, D.P., Ba, J., 2015. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- [32] Li, Z., Huang, D.Z., Liu, B., Anandkumar, A., 2023. Fourier neural operator with learned deformations for PDEs on general geometries. *Journal of Machine Learning Research* 24, 1–26.

- [33] Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., Anandkumar, A., 2020a. Fourier neural operator for parametric partial differential equations. arXiv preprint arXiv:2010.08895 .
- [34] Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., Anandkumar, A., 2020b. Neural operator: Graph kernel network for partial differential equations. arXiv preprint arXiv:2003.03485 .
- [35] Lindgren, F., Rue, H., Lindström, J., 2011. An explicit link between Gaussian fields and Gaussian Markov random fields: The stochastic partial differential equation approach. Journal of the Royal Statistical Society Series B: Statistical Methodology 73, 423–498. doi:10.1111/j.1467-9868.2011.00777.x.
- [36] Lowery, M., Turnage, J., Morrow, Z., Jakeman, J.D., Narayan, A., Zhe, S., Shankar, V., 2024. Kernel neural operators (KNOs) for scalable, memory-efficient, geometrically-flexible operator learning. arXiv preprint arXiv:2407.00809 .
- [37] Lu, L., Jin, P., Pang, G., Zhang, Z., Karniadakis, G.E., 2021. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Nature Machine Intelligence 3, 218–229.
- [38] Mora, C., Yousefpour, A., Hosseinmardi, S., Owahdi, H., Bostanabad, R., 2025. Operator learning with gaussian processes. Computer Methods in Applied Mechanics and Engineering 434, 117581.
- [39] Narcowich, F.J., Ward, J.D., 1994. Generalized Hermite interpolation via matrix-valued conditionally positive definite functions. Mathematics of Computation 63, 661–687.
- [40] Narcowich, F.J., Ward, J.D., Wright, G.B., 2007. Divergence-free RBFs on surfaces. Journal of Fourier Analysis and Applications 13, 643–663.
- [41] Owahdi, H., 2023. Gaussian process hydrodynamics. Applied Mathematics and Mechanics 44, 1175–1198. doi:10.1007/s10483-023-2990-9.
- [42] Quarteroni, A., Manzoni, A., Negri, F., 2016. Reduced Basis Methods for Partial Differential Equations: An Introduction. Springer, Cham, Switzerland. doi:10.1007/978-3-319-15431-2.
- [43] Queipo, N.V., Haftka, R.T., Shyy, W., Goel, T., Vaidyanathan, R., Tucker, P.K., 2005. Surrogate-based analysis and optimization. Progress in Aerospace Sciences 41, 1–28. doi:10.1016/j.paerosci.2005.02.001.
- [44] Rahimi, A., Recht, B., 2007. Random features for large-scale kernel machines, in: Advances in Neural Information Processing Systems, pp. 1177–1184.
- [45] Richter-Powell, J., Lipman, Y., Chen, R.T., 2022. Neural conservation laws: A divergence-free perspective. Advances in Neural Information Processing Systems 35, 38075–38088.
- [46] Santin, G., Wittwar, D., Haasdonk, B., 2018. Greedy regularized kernel interpolation. arXiv preprint arXiv:1807.09575. URL: <https://arxiv.org/abs/1807.09575>, doi:10.48550/arXiv.1807.09575.
- [47] Schäfer, F., Katzfuss, M., Owahdi, H., 2021. Sparse cholesky factorization by kullback–leibler minimization. SIAM Journal on Scientific Computing 43, A2019–A2046. doi:10.1137/20M1336254.
- [48] Shankar, V., 2017. The overlapped radial basis function-finite difference (RBF-FD) method: A generalization of RBF-FD. Journal of Computational Physics 342, 211–228. doi:10.1016/j.jcp.2017.04.037.
- [49] Shankar, V., Fogelson, A.L., 2018. Hyperviscosity-based stabilization for radial basis function-finite difference (RBF-FD) discretizations of advection–diffusion equations. Journal of Computational Physics 372, 616–639. doi:10.1016/j.jcp.2018.06.036.
- [50] Shankar, V., Kirby, R.M., Fogelson, A.L., 2018a. Robust node generation for mesh-free discretizations on irregular domains and surfaces. SIAM Journal on Scientific Computing 40, A2584–A2608. doi:10.1137/17M114090X.
- [51] Shankar, V., Narayan, A., Kirby, R.M., 2018b. RBF-LOI: Augmenting radial basis functions (RBFs) with least orthogonal interpolation (LOI) for solving PDEs on surfaces. Journal of Computational Physics 373, 722–735. doi:10.1016/j.jcp.2018.07.015.
- [52] Shankar, V., Wright, G.B., Fogelson, A.L., 2021. An efficient high-order meshless method for advection-diffusion equations on time-varying irregular domains. Journal of Computational Physics 445, 110633. doi:10.1016/j.jcp.2021.110633.
- [53] Sharma, R., Lowery, M., Owahdi, H., Shankar, V., 2026. Fluids you can trust: Property-preserving operator learning for incompressible flows. arXiv preprint arXiv:2602.15472 .
- [54] Sharma, R., Shankar, V., 2026. Operator-learning datasets for incompressible fluid flow. URL: <https://doi.org/10.5281/zenodo.22907213>, doi:10.5281/zenodo.22907213.
- [55] Shu, C., Ding, H., Yeo, K.S., 2003. Local radial basis function-based differential quadrature method and its application to solve two-dimensional incompressible Navier–Stokes equations. Computer Methods in Applied Mechanics and Engineering 192, 941–954. doi:10.1016/S0045-7825(02)00618-7.
- [56] Si, S., Hsieh, C.J., Dhillon, I.S., 2014. Memory efficient kernel approximation, in: Proceedings of the 31st International Conference on Machine Learning, pp. 2507–2515.
- [57] Snelson, E., Ghahramani, Z., 2006. Sparse gaussian processes using pseudo-inputs, in: Advances in Neural Information Processing Systems, pp. 1257–1264.
- [58] Stein, M.L., 1999. Interpolation of Spatial Data: Some Theory for Kriging. Springer Series in Statistics, Springer, New York. doi:10.1007/978-1-4612-1494-6.
- [59] Stuart, A.M., 2010. Inverse problems: A Bayesian perspective. Acta Numerica 19, 451–559. doi:10.1017/S0962492910000061.
- [60] Sudret, B., Marelli, S., Wiart, J., 2017. Surrogate models for uncertainty quantification: An overview, in: 2017 11th European Conference on Antennas and Propagation (EuCAP), IEEE. pp. 793–797. doi:10.23919/EuCAP.2017.7928679.
- [61] Tezzele, M., Demo, N., Stabile, G., Mola, A., Rozza, G., 2020. Enhancing CFD predictions in shape design problems by model and parameter space reduction. Advanced Modeling and Simulation in Engineering Sciences 7, 40. doi:10.1186/s40323-020-00177-0.
- [62] Tolstykh, A.I., Shirobokov, D.A., 2003. On using radial basis functions in a “finite difference mode” with applications to elasticity problems. Computational Mechanics 33, 68–79. doi:10.1007/s00466-003-0501-9.
- [63] Tominec, I., Larsson, E., Heryudono, A., 2021. A least squares radial basis function finite difference method with improved stability properties. SIAM Journal on Scientific Computing 43, A1441–A1471. doi:10.1137/20M1320079.

- [64] Wendland, H., 1995. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics* 4, 389–396.
- [65] Wendland, H., 2005. *Scattered Data Approximation*. Cambridge University Press, Cambridge, UK.
- [66] Wendland, H., 2009. Divergence-free kernel methods for approximating the Stokes problem. *SIAM Journal on Numerical Analysis* 47, 3158–3179.
- [67] Whittle, P., 1954. On stationary processes in the plane. *Biometrika* 41, 434–449. doi:10.1093/biomet/41.3-4.434.
- [68] Wilson, A.G., Nickisch, H., 2015. Kernel interpolation for scalable structured Gaussian processes, in: *Proceedings of the 32nd International Conference on Machine Learning*, pp. 1775–1784.
- [69] Wright, G.B., Fornberg, B., 2006. Scattered node compact finite difference-type formulas generated from radial basis functions. *Journal of Computational Physics* 212, 99–123. doi:10.1016/j.jcp.2005.05.030.
- [70] Wu, H., Luo, H., Wang, H., Wang, J., Long, M., 2024. Transolver: A fast transformer solver for PDEs on general geometries. *arXiv preprint arXiv:2402.02366*.
- [71] Yelnyk, V., 2023. RBF method for solving Navier–Stokes equations. Degree project in computational science. KTH Royal Institute of Technology. URL: <https://kth.diva-portal.org/smash/get/diva2:1849751/FULLTEXT01.pdf>.
- [72] Yu, F.X., Suresh, A.T., Choromanski, K.M., Holtmann-Rice, D., Kumar, S., 2016. Orthogonal random features, in: *Advances in Neural Information Processing Systems*, pp. 1975–1983.