



Exploring neural network surrogates for high-order mesh-free interpolants[☆]

Lucas Gerken Starepravo^{a,*,} Georgios Fourtakas^a, Steven Lind^b, Ajay B. Harish^a, Jack R.C. King^a

^a School of Engineering, The University of Manchester, Manchester, UK

^b School of Engineering, Cardiff University, Cardiff, UK

ARTICLE INFO

Dataset link: <https://github.com/uom-complexfluids/ml4meshfree.git>

Keywords:

Mesh-free
Machine learning
Discretisation

ABSTRACT

Mesh-free numerical methods offer flexibility in the discretisation of complex geometries, showing significant potential for problems where mesh-based methods struggle. Although high-order approximations can be obtained through consistency-correction linear systems, such approaches remain prohibitively expensive for Lagrangian formulations, which commonly exhibit low-order convergence. Here we investigate the use of machine learning (ML) to bridge this gap, developing two strategies to couple multilayer perceptrons (MLPs) with the Local Anisotropic Basis Function Method (LABFM) as an exemplar high-order mesh-free method. In the first strategy, neural networks are trained to directly surrogate the high-order kernel; in the second, surrogate models are developed to compute the solution of the dense, low-rank linear systems arising in high-order mesh-free discretisations. The first strategy yields qualitative agreement with validation data but only marginally outperforms inconsistent smoothed particle hydrodynamics (SPH) kernels. The second strategy produces solution vectors with mean absolute errors of $\mathcal{O}(10^{-4}-10^{-5})$, replicating LABFM second-order convergence rate up to a resolution-dependent limiting accuracy and achieving up to a 5× speedup at equivalent accuracy. A perturbation analysis across approximation orders two to eight further reveals that higher-order systems impose accuracy requirements on the predicted solution vectors orders of magnitude beyond the surrogates accuracies, representing a fundamental challenge for current neural network architectures.

1. Introduction

In the context of the numerical solution of partial differential equations (PDEs), mesh-free methods are those that require no information on connections between collocation points, allowing for collocation points to be distributed in a disordered/unstructured manner through the computational domain. This is in contrast to mesh-based methods, where the topological connections between discrete computation nodes are used in calculations. These meshless collocation points can be placed to satisfy specific criteria, such as resolution requirements and geometric boundaries, allowing efficient handling of complex geometries and dynamic problems, addressing limitations commonly encountered in traditional mesh-based methods.

Developed in the 1970s and originally formulated for astrophysical problems, smoothed particle hydrodynamics (SPH) has been adapted to a broad range of physical systems and is today perhaps the most widely used mesh-free numerical method [1,2]. In

[☆] This article is part of a Special issue entitled: ‘CPMS_SPHERIC’ published in Computational Particle Mechanics.

* Corresponding author.

E-mail address: lucas.gerkenstarepravo@postgrad.manchester.ac.uk (L.G. Starepravo).

this framework, the continuum domain is represented by a set of (typically) Lagrangian particles, with interactions governed by a smoothing kernel. In its standard formulation, SPH exhibited low-order convergence [3]. This limitation is characteristic of many Lagrangian mesh-free methods [4], which tend to operate at low-order accuracy since high-order approximations often require solution of linear systems. Although high-order approximations can be advantageous — achieving improved accuracy at the same resolution compared to low-order approximations — the computational cost becomes prohibitive in Lagrangian frameworks due to the need for frequent solution of linear systems, as the Lagrangian nodes move in time.

Nearly 50 years after the introduction of SPH, a plethora of mesh-free methods has emerged. The reproducing kernel particle method (RKPM) [5], generalised finite difference method (GFDM) [6,7], the generalised moving least squares (GMLS) [8–10], the radial basis-finite difference methods [11], and the local anisotropic basis function method (LABFM) [12–14] are examples of mesh-free numerical methods that show high-order convergence by solving linear systems to calculate kernels or inter-particle weights. For a review of high-order mesh-free methods, we refer the reader to the introduction of [12]. The above methods are typically employed in an Eulerian framework, and to the best of the authors' knowledge, no formally high-order mesh-free methods have been developed and implemented at scale for a large number of degrees of freedom in a Lagrangian framework, due to the prohibitive computational costs. This creates a gap between Lagrangian and high-order mesh-free methods, which has motivated the development of computationally cheaper high-order mesh-free methods [4].

Machine Learning (ML) has received considerable attention recently across many different fields [15–19]. In particular, neural network architectures — a central component of ML — leverage the universal approximation (UA) theorem, which allows feed-forward architecture to approximate any continuous function between two Euclidean spaces [20]. In fluid dynamics simulations, especially the ones performed in mesh-based frameworks, ML has been extensively used to accelerate simulations; for a comprehensive review, see [21]. In contrast, mesh-free methods have received comparatively less attention from the ML community. Nonetheless, recent advancements have demonstrated the potential of ML in this domain, leading to notable progress. In [22], multi-layer perceptrons (MLPs) were used to estimate the hydrodynamic forces between particles in SPH. Results showed the MLP predictions were faithful to standard SPH results in dam break, and also presented good generalisation for different geometries than the ones present in the training dataset. In [23], graph neural networks (GNNs) were implemented to surrogate the pressure Poisson's equation in incompressible SPH (ISPH). This approach produced qualitatively similar results for the tested cases, exhibited convergence behaviour comparable to ISPH, and demonstrated satisfactory generalisation to more complex systems. The methods discussed above leveraged the system's state to train data-driven models, which, despite demonstrating satisfactory generalisation, are inherently limited to similar physical regimes. Beyond these applications, ML has also been explored as a tool to mitigate the computational costs of solving linear systems in discretised PDEs, offering alternative pathways for improving numerical efficiency.

A neural network-based algorithm to solve PDEs with high precision was developed in [24], which solved large and sparse diagonal matrices that often arise from mesh-based numerical frameworks, and employed a residual network architecture together with a correction routine, obtaining errors smaller than 10^{-7} between the solution vector computed with LU factorisation and the neural network. However, it was observed that the quality of the solution deteriorated as the diagonal dominance of the linear system decreased. In [25] neural networks were trained to solve large linear systems with a primary emphasis on reducing memory requirements. In this framework, the solution vector of matrix-free linear systems is approximated with a neural network representation, containing fewer degrees of freedom than the original linear system. The method proposed allows large linear systems that have prohibitive storage requirements with traditional linear solvers to be solved. However, the solution obtained with the neural network is not as accurate as traditional linear solvers, consequently the authors do not recommend the framework for small linear systems that can be solved with traditional methods. Hamrani et al. [26] applied ML to surrogate the computationally heavy construction of the shape function in mesh-free methods. The authors employed multiple data-driven techniques, such as gated recurrent neural networks, ensemble learning and support vector machine to predict the shape functions in mesh-free weak-form Galerkin methods. The models were evaluated in one-, two- and three-dimensions, where the Gaussian process (GP) had the most accurate results marginally outperforming MLPs. A computational cost analysis demonstrated a 20- and 1000-fold speed-up when using deep neural networks and Gaussian processes, respectively. Furthermore, a convergence analysis was conducted to compare the mesh-free algorithms with the ML-predicted weights. The neural network, Gaussian process, and the original algorithms on which they were trained exhibited similar convergence rates, with the latter producing marginally more accurate results. However, the convergence analysis was performed on coarse resolutions (up to 2×10^3 nodes in the domain), which does not accurately reflect the resolutions encountered in standard numerical simulations. Furthermore, the work does not investigate the impact of the surrogate in strong-form collocation. Mesh-free methods may exhibit divergent behaviour at high resolutions due to inconsistencies in their numerical formulation. Similarly, shape function surrogates can introduce inaccuracies, as the data-driven models often fail to perfectly predict the shape functions. Therefore, to accurately verify the stability of the ML-based shape functions, it is essential to test the surrogate across the full spectrum of relative resolutions encountered in real simulations. An ongoing issue in applied ML is the utilisation of weak baselines and reporting biased results [27,28]. These biases can lead to overoptimistic assessments of model performance, which underscores the importance of rigorous and representative testing conditions to ensure the reliability and accuracy of the trained models.

To bridge the gap between Lagrangian mesh-free simulations and Eulerian high-order mesh-free numerical methods, we formulate two strategies that leverage ML to surrogate specific segments of the latter. Thus, in this manuscript, we propose a framework to compute mesh-free stencil weights that approximate differential operators more accurately than uncorrected SPH kernels, and at a lower computational cost than conventional high-order mesh-free methods. We have tested both approaches on LABFM, given its arbitrarily high-order convergence on semi-unstructured nodes and its suitability to high-fidelity simulation of non-linear PDEs [14]. However, the framework presented is not restricted to any particular high-order mesh-free method.

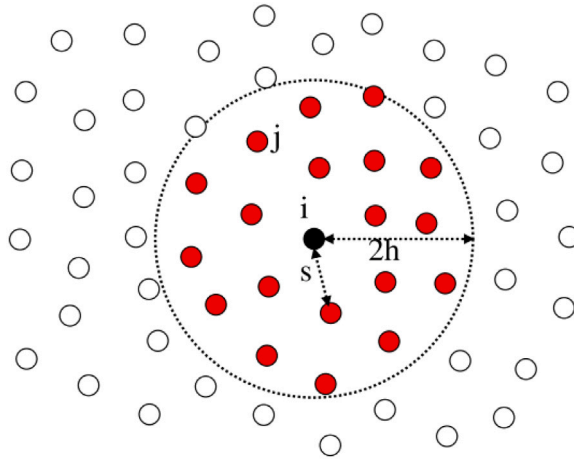


Fig. 1. Illustration of a mesh-free computational stencil [13].

This paper investigates the use of MLPs to surrogate two distinct components of high-order strong-form mesh-free methods. In the first approach, a neural network is trained to directly predict the weights of a computational stencil from the positions of the support nodes relative to the central node, effectively surrogating the entire kernel. In the second approach, a neural network is trained to solve the dense, ill-conditioned, low-rank linear systems characteristic of strong-form mesh-free discretisations. Both approaches exploit the fact that the weights are exclusively a function of nodal coordinates, making the models agnostic to the underlying physics and applicable across different physical systems and mesh-free frameworks. While the first approach reveals fundamental limitations of MLPs in achieving the accuracy required for reliable convergence, the second demonstrates that a well-chosen surrogate offers a tunable trade-off between accuracy and computational cost. Smaller models incur a higher limiting error but provide substantial speed-ups over direct linear solvers — whose cost is fixed, since the system is always solved to the precision of the method — whereas larger models reduce the limiting error at greater computational expense.

The remainder of the paper is set out as follows. Section 2 and Section 3 provide a technical background on LABFM and ML, respectively. Section 4 provides a description of the two approaches developed to accelerate high-order mesh-free methods. Section 5 presents the results obtained in the two approaches. Section 6 provides caveats on the application of neural networks for surrogates of high-order mesh-free methods. Lastly, Section 7 concludes the findings of the investigation.

2. The local anisotropic basis function methods (LABFM)

In this work, we use the LABFM as an exemplar high-order mesh-free method, noting that it shares characteristics (structure of computational stencil, form of discrete operator, and local linear system) with other high-order mesh-free methods (e.g. GFDM, GMLS, radial basis function finite difference [29,30]). For a comprehensive description and derivation of LABFM, we refer the reader to [12].

2.1. General discrete operator

In LABFM, a general discrete operator is defined to approximate differential operators as shown below:

$$L_i^d(\cdot) = \sum_{j \in \mathcal{N}} (\cdot)_{ji} w_{ji}^d \quad (1)$$

where $L_i^d(\cdot)$ is the discrete operator; d identifies the differential operator being approximated; w_{ji}^d are a set of weights; and $\mathcal{N}$ is the set of nodes within a computational stencil; the subscripts i and j denote the central and support nodes, respectively, with scalar differences expressed as $(\cdot)_{ij} = (\cdot)_i - (\cdot)_j$.

Fig. 1 illustrates the node distribution and stencil. The smoothing length, h_i , determines the size of the computational stencil, influencing the range of interactions among nodes (in [12], it was defined critical values for the smoothing length based on the order of approximation of the discrete operator). The weights w_{ji}^d are given by a weighted sum of anisotropic basis functions (ABFs),

$$w_{ji}^d = \mathbf{W}_{ji} \cdot \boldsymbol{\Psi}_i^d = W_{ji}^1 \Psi_{i,1}^d + W_{ji}^2 \Psi_{i,2}^d + W_{ji}^3 \Psi_{i,3}^d + \dots \quad (2)$$

in which the vector $\mathbf{W}_{ji}$ are the ABFs, and $\boldsymbol{\Psi}_i^d$ is a coefficient vector determined by solving the following linear system:

$$\mathbf{M}_i \boldsymbol{\Psi}_i^d = \mathbf{C}^d \quad (3)$$

The vector, C^d (also referred to as the stencil moments), ensures that the coefficient vector, Ψ_i^d , is appropriate to approximate the desired differential operator. For first-order spatial derivatives and the Laplace operator in a two-dimensional domain the are given by:

$$C^d = \begin{cases} [1, 0, 0, 0, 0, 0, \dots]^T & \text{if } d = x \\ [0, 1, 0, 0, 0, 0, \dots]^T & \text{if } d = y \\ [0, 0, 1, 0, 1, 0, \dots]^T & \text{if } d = \Delta \end{cases} \quad (4)$$

where x , y and Δ refer respectively to the spatial derivatives in each respective direction and the Laplace operator. For a given support region, M_i is independent of the differential operator being approximated, given by:

$$M_i = \sum_{j \in \mathcal{N}} X_{ji} \otimes W_{ji} \quad (5)$$

where X_{ji} is the vector containing Taylor monomials

$$X_{ji} = \left[x_{ji}, y_{ji}, \frac{x_{ji}^2}{2!}, x_{ji}y_{ji}, \frac{y_{ji}^2}{2!}, \frac{x_{ji}^3}{3!}, \frac{x_{ji}^2y_{ji}}{2!}, \dots \right]^T \quad (6)$$

note that for the monomial vector $X_{ji} \neq X_j - X_i$. The matrix M_i as defined above has infinite rank. In practice, we restrict the size of M_i by including only the first k elements of W_{ji} and X_{ji} . We highlight here the relationship between the number of elements k , and the order p of the k th element of X_{ji}

$$k = \frac{p^2 + 3p}{2} \quad (7)$$

in 2 dimensions.

2.2. Anisotropic basis functions

To construct the ABFs, bivariate Hermite polynomials are combined with a radial basis function (RBF). The q th element of X_{ji} is proportional to $x_{ji}^a y_{ji}^b$. Thus, the q th ABF is defined as

$$W_{ji}^q = \frac{\psi(|r_{ji}|/h_i)}{\sqrt{2^{a+b}}} H_a\left(\frac{x_{ji}}{h_i \sqrt{2}}\right) H_b\left(\frac{y_{ji}}{h_i \sqrt{2}}\right) \quad (8)$$

where r_{ji} is the Euclidean distance between the nodes j and i ; H_a is the a th order univariate Hermite polynomial (of the physics kind), generated from

$$H_a(z) = (-1)^a e^{z^2} \frac{d^a}{dz^a} e^{-z^2} \quad (9)$$

and ψ is an RBF. In this work, the Wendland C2 kernel (shown in [Appendix A.1](#) is used, following [\[13\]](#)).

3. Machine learning

3.1. Multi-layer perceptrons

The foundational structures of MLPs consist of an input layer, hidden layers, and an output layer. The hidden layers consist of perceptrons (or neurons) interconnected by weights across successive layers. Each neuron receives input from the preceding layer and, in the most basic architectures, performs two primary functions: calculating a weighted sum of its inputs and applying an activation function to the resulting sum. The operations in each layer can be expressed as follows:

$$c^l = \theta^l a^{l-1} + b^l \quad (10a)$$

$$a^l = \sigma(c^l) \quad (10b)$$

where superscript l refers to the layer index; c represents the pre-activation vector, which is the weighted sum of the activations of the previous layer; a refers to the vector of activations; b represents the bias vector, and θ is the weight matrix that connects the neurons in successive layers; σ is the activation function, which introduces non-linearity into the network, enabling it to exhibit the universal approximation (UA) property. Examples of activation functions are the sigmoid and rectified linear unit (ReLU). For a detailed review of MLPs, the reader is referred to [\[31\]](#).

4. Method

In the current section, we will demonstrate two approaches that surrogate segments of high-order mesh-free methods with ML. These approaches were implemented in LABFM as an exemplar, due to its high-order convergence and applicability to high-fidelity simulation of non-linear PDEs. The objective is to approximate differential operators with high-order accuracy and increased computational efficiency to bridge the existing gap between Lagrangian mesh-free methods and high-order interpolants.

4.1. High-order kernel surrogate with machine learning

In this investigation, we leveraged the UA property of feed-forward neural networks to map the distance between the support nodes and the central node to the LABFM weights, w_{ji}^d . The network inputs were the positions of each support node $j \in \mathcal{N}$ relative to the central node i , given by the sets $\{x_{ji}\}_{j \in \mathcal{N}}$ and $\{y_{ji}\}_{j \in \mathcal{N}}$, ordered by radial distance. The central node and its corresponding weight were excluded from the model. This exclusion is justified by the fact that, according to Eq. (1), the weight of the central node is irrelevant, since $(\cdot)_i - (\cdot)_i = 0$, it has no impact on the discrete operator. Consequently, the input vector consists only of the coordinate differences for the support nodes, resulting in a size of $\mathbb{R}^{d|\mathcal{N}|-d}$, and the output consists of the LABFM weights for the support nodes, with a size of $\mathbb{R}^{|\mathcal{N}|-1}$, where d is the number of dimensions.

The inflexibility of feed-forward networks with varying input and output size requires a fixed number of support nodes in the support region. In LABFM, the number of support nodes must be $|\mathcal{N}| \geq |\mathcal{N}_{crit}|$, where $|\mathcal{N}_{crit}|$ was defined in King et al. [12] for different orders of approximation. In this work, we use LABFM at second-order, which is expected to be easier to surrogate compared to higher-order approximations. Higher-order LABFM approximations introduce high-frequency variations in the weights, particularly near the stencil edges, which conflicts with the spectral bias of artificial neural networks towards learning smooth, low-frequency functions. The stencil is defined as the 30 nearest nodes to the central node — matching the approximate neighbour count of the SPH Wendland C2 kernel (Appendix A) — identified via nearest-neighbour search. Fixed-count stencils of this kind are standard in high-order mesh-free frameworks, including RBF-FD [32] and the Generalised Finite Difference Method [33], where the stencil size is a user-selected parameter. This construction guarantees a constant input size regardless of local node density. The ground-truth LABFM weights used for training are computed on the same 30-node stencils, so surrogate and reference operate on identical support sets.

Proper normalisation of inputs and outputs improves weight convergence during network trainings [34]. In the current investigation, the training dataset comprises points from domains with varying resolution, leading to significant variations in distances and weight magnitudes. These weights are inversely proportional to the smoothing length of the computational stencil, $w \propto 1/h$ and $w \propto 1/h^2$ for first and second derivatives, respectively. To mitigate resolution-dependent variability, weights and distances are normalised by the smoothing length of each computational stencil. This normalisation rescales the weights to a uniform magnitude, independent of the stencil size and the underlying resolution, ensuring consistent learning and prediction across different node resolutions, as detailed below:

$$\hat{x}_{ij} = \frac{x_{ij}}{h_i}, \quad \hat{y}_{ij} = \frac{y_{ij}}{h_i} \quad (11a)$$

$$\hat{w}_{ij} = \begin{cases} w_{ij} h_i & \text{if } d = x, y \\ w_{ij} h_i^2 & \text{if } d = \Delta \end{cases} \quad (11b)$$

4.2. Solving linear systems with machine learning

The second approach developed in this paper is a more direct way to accelerate high-order mesh-free methods with ML. The computational overhead in high-order interpolants is largely caused by the linear systems that must be solved to calculate weights. Thus, in this approach, we implement feed-forward networks to surrogate the solution of the dense, ill-conditioned, low-rank linear systems with a constant right-hand side vector (given by (3) in LABFM), which contains fewer intermediate steps compared to the approach shown in Section 4.1.

The models built in this approach have as input the flattened $\mathbf{M}_i$ matrix and as output the weights, Ψ_i^d . The network operations can be described as $f : \mathbb{R}^{k^2} \rightarrow \mathbb{R}^k$, where k is the rank of the linear system. Given the moments are constant for each differential operator being approximated, they do not provide additional information and can be omitted from the network.

During training, the moments were normalised with respect to the smoothing length, h_i . Similarly to the approach presented in Section 4.1, the normalisation depends on which operator is being approximated.

$$\hat{\Psi}_i = \begin{cases} \Psi_i h_i & \text{if } d = x, y \\ \Psi_i h_i^2 & \text{if } d = \Delta \end{cases} \quad (12)$$

This normalisation rescales the weights to a magnitude independent of node distance, ensuring learning and prediction across different node resolutions. The entries of the linear system for a second-order approximation exhibit uniform magnitudes and are relatively small. Consequently, normalisation of the input data during training was deemed unnecessary.

4.3. Training & testing details

Training data consists of local particle neighbourhoods $\mathcal{N}_i$ generated from synthetic disordered point clouds. Starting from a regular Cartesian grid with mean spacing s_u , each node is independently perturbed by a displacement drawn from $\mathcal{U}(-\frac{\epsilon s_u}{2}, \frac{\epsilon s_u}{2})$ along each coordinate, where ϵ is the non-dimensional disorder intensity. By construction, the perturbation magnitude scales with the grid spacing, allowing operator sensitivity to be studied systematically across discretisation scales. Unless stated otherwise, all models are trained at $\epsilon = 1.0$, a disorder level considerably higher than that encountered in typical mesh-free simulations, ensuring robustness under severe geometric irregularity.

The MLPs were trained using the PyTorch framework [35] with a mean squared error loss function. The training dataset comprised approximately 1.6 million computational stencils, split 80% for training and 20% for validation. Throughout Section 5, superscripts in the kernel name denote the target operator (e.g. the Laplacian) and subscripts denote the order of accuracy used to generate the LABFM weights (e.g. LABFM_{p=2} indicates second-order weights).

The networks underwent offline training, where the hyperparameters of the trained surrogates can be found in Appendix B, and the results were evaluated in three distinct phases. The first is a qualitative analysis of the weights. The second phase involved using the networks to compute the stencil moments with the predicted weights, a critical measure of the quality of the predicted weights. Lastly, a convergence study was conducted using the ML-computed weights with the designated test function:

$$\phi(x^*, y^*) = 1.0 + (x^* y^*)^4 + \sum_{i=1}^6 (x^{*i} + y^{*i}) \quad (13)$$

where $x^* = x - 0.1453$ and $y^* = y - 0.16401$. This equation matches the one used in [12]. The offset is pseudo-random, and included to ensure asymmetry in the function, to prevent the masking of errors which could cancel for a symmetric function.

5. Results & discussion

5.1. Predicted weighted anisotropic basis functions

In this section, we present results for surrogating the high-order kernel with an MLP. The analysis is divided into two parts: (i) a qualitative analysis (Section 5.1.1) and (ii) a quantitative analysis (Section 5.1.2).

5.1.1. Qualitative analysis

Fig. 2 shows results for the LABFM, MLP surrogate, quintic spline, and Wendland C2 kernels evaluated on multiple computational stencils absent from the training dataset. The stencils are overlaid, with colour representing weight magnitude. The top row shows the weights predicted for the x -derivative operator and the bottom row for the Laplacian. Visually, LABFM and the MLP surrogate are indistinguishable, exhibiting similar weight magnitudes and distributions across the stencil ensemble. In contrast, the SPH kernels produce considerably smoother weight fields. This is because SPH weights depend only on the relative position between each neighbour and the central node, making each weight assignment independent of the remaining nodes in the stencil. LABFM, on the other hand, computes weights by solving a linear system constructed from the positions of all nodes within the stencil (Eq. (5)), which is smooth for each individual stencil but appears noisy when superimposed. Despite this apparent noise, for the x -derivative, both LABFM and the MLP surrogate exhibit approximate anti-symmetry in the weight distribution about the origin, in the direction of the derivative, and rotational symmetry for the Laplacian. The stencil boundaries also appear noisier than those of the SPH kernels, which is attributed to the use of a fixed neighbour count in LABFM and the MLP surrogate, as opposed to the fixed cutoff radius employed by SPH kernels. Although LABFM does not require a fixed neighbour count, it was set to match the MLP surrogate for a fair comparison.

Fig. 3 presents the absolute and relative errors of the MLP surrogate with respect to standard LABFM for the x -derivative and Laplacian weights. For the x -derivative, the absolute errors are distributed uniformly across the overlaid stencils, with a maximum corresponding to approximately 40% of the largest weight magnitude predicted by the MLP (Fig. 2). The relative errors reach order 10^3 and concentrate near $x = 0$, orthogonal to the direction of differentiation. This is expected: nodes located orthogonally to the derivative direction carry weights close to zero, so even small absolute errors yield large relative errors there. For the Laplacian, the absolute errors are likewise distributed uniformly, with a maximum of approximately 9×10^1 , corresponding to roughly 8% of the largest weight magnitude, while the relative errors reach order 10^2 .

5.1.2. Quantitative analysis

Quantitatively, the results are evaluated through stencil moments and convergence analysis. For all stencils in the validation dataset, the predicted weights were used to compute the moments of the x -derivative and Laplacian operators. The absolute error between predicted and target moments was then computed and averaged across all stencils, with the standard deviation also reported. These residuals indicate the precision to which each term in the Taylor expansion approximates the true differential operator, up to the truncated order. The absolute error between the target and the predicted stencil moments yields the linear system residual in Eq. (3). The results are summarised in Table 1.

For the gradient operator, MLP residuals remain consistently between $\mathcal{O}(10^{-3})$ and $\mathcal{O}(10^{-2})$ across all first- and second-order monomials, compared to $\mathcal{O}(10^{-2})$ – $\mathcal{O}(10^{-1})$ for SPH. The MLP Laplacian yields slightly higher residuals than the derivative operator with residuals mostly of order 10^{-2} , whereas for the SPH kernels the gap is more pronounced, particularly in the first-order moments. This degradation in the Laplacian is consistent with the heightened sensitivity of second-order derivatives to geometric perturbations, a well-documented behaviour in high-order mesh-free and spectral discretisations [36]. Taken together, these results suggest that polynomial consistency constraints can be learned within a supervised framework, and generalise to unseen stencil configurations. However, the accuracy obtained is fairly limited, given the learned weights marginally outperform uncorrected SPH.

A convergence analysis was subsequently performed using the trained models to approximate the derivative and Laplacian operators via a test function (Eq. (13)), with results shown in Fig. 4. For the derivative operator, the MLP surrogate exhibits convergence only for $s \in [0.1, 0.5]$, marginally outperforming the SPH kernels. The limiting error in the convergence is governed by the sum of the first-order moment residual. Outside this range, the MLP surrogate fails to converge, likely because the moment

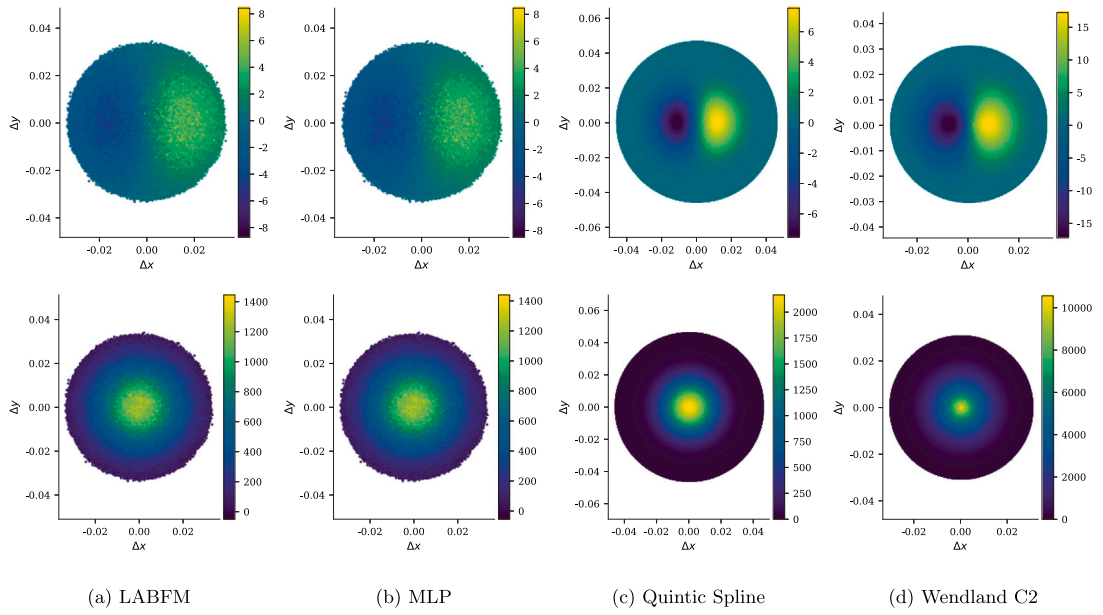


Fig. 2. Overlaid plot of multiple computational stencils with different discretisation methods for x -derivative (*top row*) and Laplacian (*bottom row*), colours indicate weight value. The MLPs were trained with particle disturbance of $\epsilon = 1.0$ and inferred with the same noise.

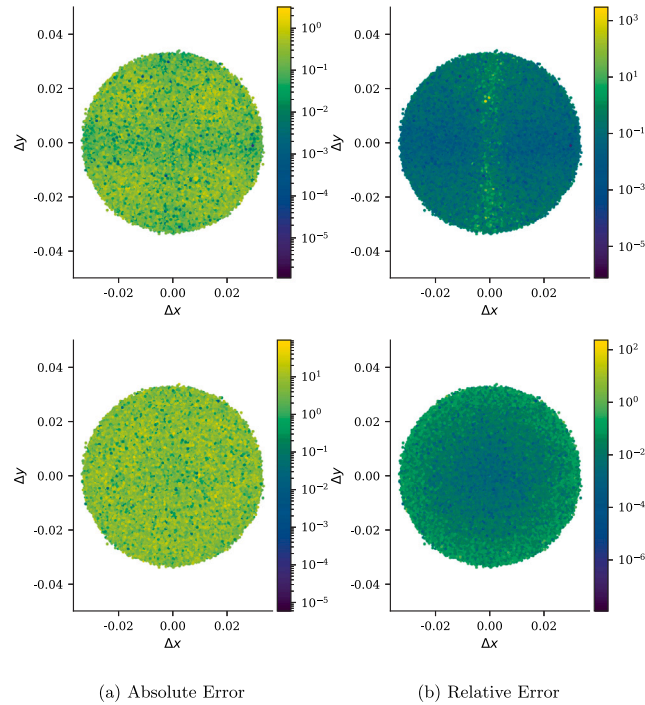


Fig. 3. Overlaid plot of multiple computational stencils with the absolute error (*left column*) and relative error (*right column*) for x -derivative (*top row*) and Laplacian (*bottom row*) for the MLP surrogate with respect to standard LABFM weight. The MLPs were trained with particle disturbance of $\epsilon = 1.0$ and inferred with the same noise.

residuals reported in Table 1 are insufficient to sustain consistency at finer resolutions. LABFM, by contrast, achieves second-order convergence across the entire resolution range tested, as consistency is enforced up to the truncation error of the linear system solve [12,32]. For the Laplacian, both the SPH and MLP operators exhibit divergent behaviour across all resolutions tested, which follows directly from the failure to satisfy the moment conditions to a sufficient degree, as evidenced by the higher residuals in

Table 1

Moment residuals for MLP and uncorrected SPH discretisations with Wendland C2 and quintic spline kernels. Mean absolute error (MAE) and standard deviation are reported for each monomial moment, averaged over independently perturbed neighbourhood realisations with $\epsilon = 1.0$. Residuals quantify the extent to which the discretisation satisfy the imposed Taylor consistency constraints. Superscripts denote the target operator and subscripts the polynomial order of approximation.

Kernel	Metric	x	y	$x^2/2$	xy	$y^2/2$
MLP ^x _{p=2}	MAE	2.87×10^{-3}	5.18×10^{-2}	1.44×10^{-3}	3.18×10^{-2}	4.93×10^{-3}
	St.d.	3.75×10^{-3}	6.52×10^{-2}	1.88×10^{-3}	4.00×10^{-2}	6.26×10^{-3}
MLP ^d _{p=2}	MAE	3.84×10^{-2}	3.79×10^{-2}	9.11×10^{-3}	1.74×10^{-2}	9.67×10^{-3}
	St.d.	5.03×10^{-2}	4.97×10^{-2}	1.19×10^{-2}	2.27×10^{-2}	1.18×10^{-2}
Wendland C2 ^x	MAE	1.01×10^{-1}	7.08×10^{-2}	4.80×10^{-2}	4.93×10^{-2}	2.55×10^{-2}
	St.d.	1.21×10^{-1}	8.84×10^{-2}	5.90×10^{-2}	6.16×10^{-2}	3.18×10^{-2}
Wendland C2 ^d	MAE	3.31×10^{-1}	3.33×10^{-1}	1.03×10^{-1}	1.50×10^{-1}	1.05×10^{-1}
	St.d.	4.13×10^{-1}	4.13×10^{-1}	1.22×10^{-1}	1.87×10^{-1}	1.24×10^{-1}
Quintic Spline ^x	MAE	6.59×10^{-2}	4.57×10^{-2}	3.77×10^{-2}	3.98×10^{-2}	2.12×10^{-2}
	St.d.	8.04×10^{-2}	5.75×10^{-2}	4.69×10^{-2}	4.95×10^{-2}	2.64×10^{-2}
Quintic Spline ^d	MAE	1.52×10^{-1}	1.49×10^{-1}	6.29×10^{-2}	9.47×10^{-2}	6.42×10^{-2}
	St.d.	1.90×10^{-1}	1.86×10^{-1}	7.70×10^{-2}	1.18×10^{-1}	7.89×10^{-2}

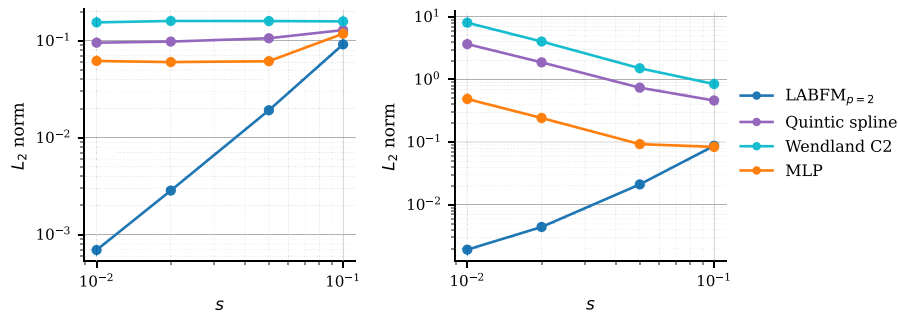


Fig. 4. Convergence of the x -derivative (*left*) and Laplacian (*right*) on a smooth test function with $\epsilon = 1.0$. Relative L_2 error versus particle spacing s for the MLP as a discretisation surrogate, a second-order LABFM kernel, and two uncorrected SPH kernels (quintic spline and Wendland C2).

Table 1. These convergence results expose a fundamental limitation of directly surrogating the high-order kernel: although the MLP learns a qualitatively accurate approximation, polynomial consistency is highly sensitive to weight errors, and the residual moment errors are large enough to preclude convergence.

The same MLP, trained on disordered node distributions with $\epsilon = 1.0$, is then applied to collocation points generated by the node-shifting algorithm of [37]. Fig. 5 shows that the quintic spline, Wendland C2, and MLP surrogate all perform better on these distributions than on the disordered Cartesian nodes of Fig. 4. For the x -derivative, both SPH kernels reduce their limiting error by approximately an order of magnitude, and the MLP surrogate by approximately half an order of magnitude; consequently, both uncorrected SPH kernels now attain a lower limiting error than the MLP surrogate. For the Laplacian, the quintic spline exhibits convergent behaviour, while the Wendland C2 and MLP surrogate show only minor improvement relative to Fig. 4. This result reinforces the limitation identified above and motivates the alternative strategy pursued in Section 5.2: rather than surrogating the kernel directly, the network is tasked with solving the consistency-enforcing linear system, where the moment conditions are embedded in the target itself.

5.2. Analysis of machine learning solution of linear systems

In this section, we present results for surrogating the ill-conditioned, low-rank, dense linear systems with constant right-hand side vectors present in high-order mesh-free methods with a MLP.

5.2.1. Qualitative analysis

The predicted kernel qualitatively matches the LABFM kernel solved by LU factorisation (Fig. 2), for both the derivative and Laplacian operators, exhibiting similar weight magnitudes, anti-symmetric distributions for the derivative, and rotational symmetry for the Laplacian. The same qualitative features discussed in Section 5.1.1 are observed here, and the same interpretation applies.

Fig. 7 shows the absolute and relative errors of the weights obtained via the linear-system surrogate with respect to those computed by LU factorisation. Qualitatively, the error distributions resemble those of the direct-weight surrogate (Fig. 3). The key difference is the magnitude: the absolute and relative errors are approximately two orders of magnitude smaller for the x -derivative,

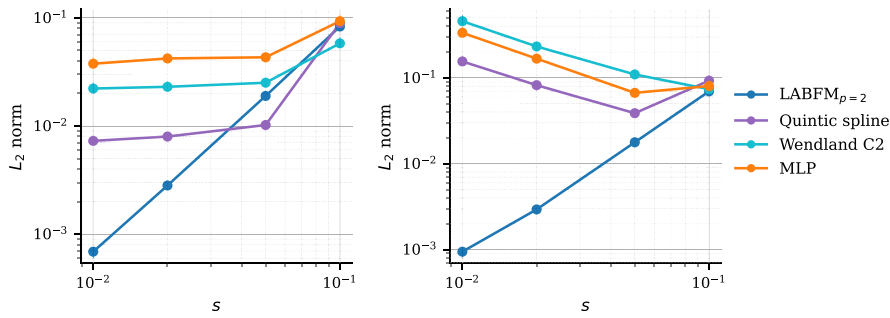


Fig. 5. Convergence of the x -derivative (left) and Laplacian (right) on a smooth test function with collocation points based on node shifting. Relative L_2 error versus particle spacing s for the MLP as a discretisation surrogate, a second-order LABFM kernel, and two uncorrected SPH kernels (quintic spline and Wendland C2).

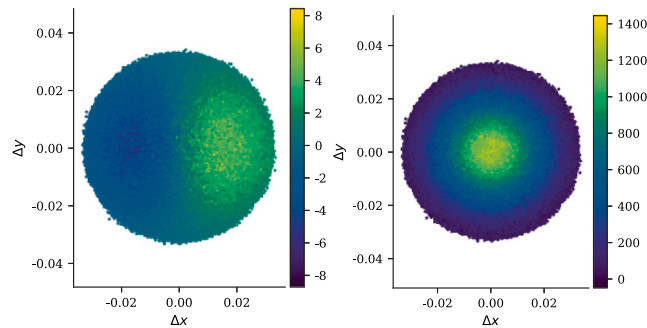


Fig. 6. Overlaid plot of multiple computational stencils with MLP surrogating the linear system for x -derivative (left) and Laplacian (right), colours indicate weight value. The MLPs were trained with particle disturbance of $\epsilon = 1.0$ and inferred with the same noise.

Table 2

Moment residuals for MLP linear system surrogate. Mean absolute error (MAE) and standard deviation are reported for each monomial moment, averaged over independently perturbed neighbourhood realisations with $\epsilon = 1.0$. Residuals quantify the extent to which the discretisation satisfy the imposed Taylor consistency constraints. Superscripts denote the target operator and subscripts the polynomial order of approximation.

Operator	Metric	x	y	$x^2/2$	xy	$y^2/2$
MLP ^x _{p=2}	MAE	1.66×10^{-4}	1.28×10^{-4}	5.40×10^{-5}	2.54×10^{-5}	5.45×10^{-5}
	St.d.	2.11×10^{-4}	2.05×10^{-4}	8.62×10^{-5}	4.22×10^{-5}	8.56×10^{-5}
MLP ^d _{p=2}	MAE	3.18×10^{-4}	3.13×10^{-4}	1.77×10^{-4}	7.03×10^{-5}	1.84×10^{-4}
	St.d.	5.03×10^{-4}	5.23×10^{-4}	2.75×10^{-4}	1.26×10^{-4}	2.79×10^{-4}

and 1.5 orders of magnitude smaller for the Laplacian, indicating substantially more accurate weights than the direct-weight approach.

5.2.2. Quantitative analysis

Following Section 5.1.2, we evaluate the consistency and convergence of the linear system surrogate, with moment residuals reported in Table 2. For the derivative operator, first-order moment residuals are of order 10^{-4} across all models, while second-order moments achieve slightly higher accuracy at 10^{-5} , for both mean absolute error and standard deviation. The Laplacian exhibits marginally larger residuals than the x -derivative, consistent with the trend observed in Table 1. Overall, surrogating the linear system yields a reduction of two to three orders of magnitude in moment residuals compared to directly surrogating the kernel, representing a substantial improvement in polynomial consistency.

A convergence analysis was subsequently performed comparing the linear system surrogate against standard LABFM with LU factorisation and the SPH kernels, with results shown in Fig. 8. For the x -derivative, the MLP surrogate exactly matches LABFM convergence up to $s = 10^{-2}$, beyond which it plateaus, indicating that the limiting error has been reached. The Laplacian exhibits similar behaviour, with the surrogate matching LABFM up to a lower resolution and at reduced accuracy compared to the derivative operator, before diverging. Overall, this approach represents a marked improvement over the full kernel surrogate. While the limiting error is necessarily larger than LABFM — since the linear system is not solved explicitly — the results demonstrate that the surrogate is viable for applications requiring moderate accuracy.

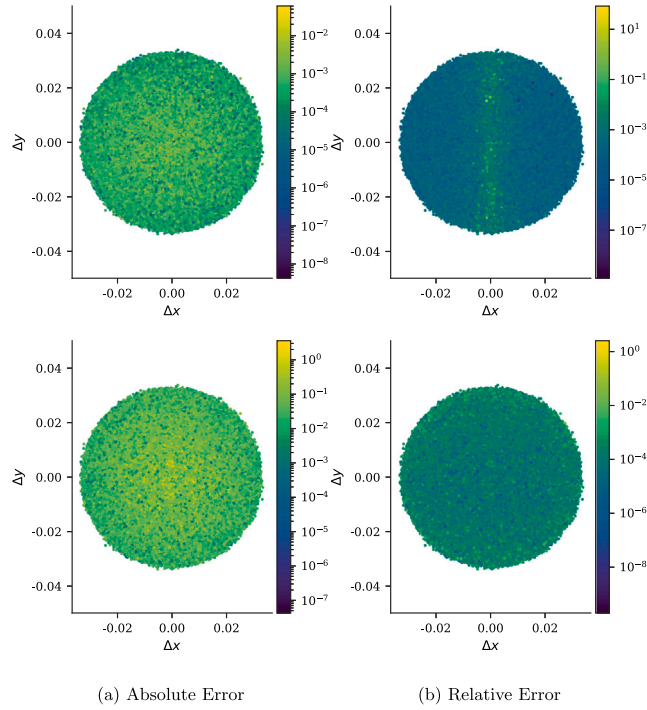


Fig. 7. Overlaid plot of multiple computational stencils with the absolute error (*left column*) and relative error (*right column*) for x -derivative (*top row*) and Laplacian (*bottom row*) for the linear system surrogate with respect to standard LABFM weight. The MLPs were trained with particle disturbance of $\epsilon = 1.0$ and inferred with the same noise.

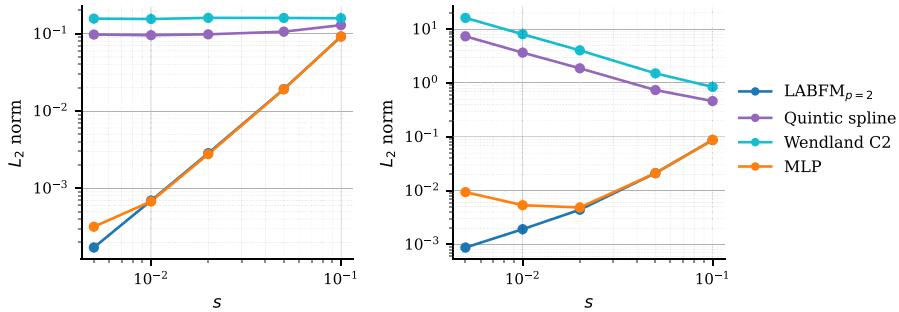


Fig. 8. Convergence of the x -derivative (*left*) and Laplacian (*right*) on a smooth test function with $\epsilon = 1.0$. Relative L_2 error versus particle spacing s for the MLP as a linear system surrogate, a second-order LABFM kernel with LU factorisation, and two uncorrected SPH kernels (quintic spline and Wendland C2).

We next evaluate the linear-system surrogate — again trained on disordered node distributions with $\epsilon = 1.0$ — on collocation points generated by the node-shifting algorithm of [37], results are shown in Fig. 9. As in Section 5.1.2, the uncorrected kernels operators benefit from the improved node distributions. For the x -derivative, the surrogate exhibits convergent behaviour across all resolutions tested; at resolutions of 5×10^{-3} – 10^{-2} its error falls marginally below that of standard LABFM, which we attribute to the stochasticity of the collocation points rather than to a genuine accuracy advantage. For the Laplacian, the limiting error decreases from approximately 5×10^{-3} to 3×10^{-3} , a modest but measurable improvement. Since the model is unchanged, these gains reflect the sensitivity of the linear systems — and hence of the difficulty of the surrogate’s task at inference — to the quality of the node distribution. An in-depth study of how node placement affects both training and inference accuracy is left for future work.

Lastly, a computational cost analysis was conducted to assess the competitiveness of the MLP surrogate relative to LU factorisation for solving the linear system. Since the output must be rescaled to its correct magnitude before use, the total forward pass time is reported. Two models are evaluated: MLP_{*l*}, a larger model with more trainable parameters targeting a lower limiting error, and MLP_{*s*}, a smaller model prioritising computational efficiency.

Fig. 10 shows the accuracy–computational cost trade-off for the linear system MLP surrogates. MLP_{*l*} is consistently more expensive than both LABFM and MLP_{*s*}, though it maintains convergent behaviour across all resolutions tested, reflecting the accuracy

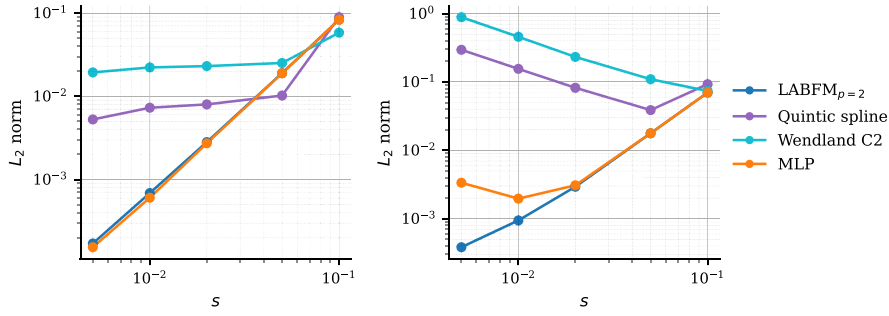


Fig. 9. Convergence of the x -derivative (left) and Laplacian (right) on a smooth test function with collocation points placed with iterative shifting. Relative L_2 error versus particle spacing s for the MLP as a linear system surrogate, a second-order LABFM kernel with LU factorisation, and two uncorrected SPH kernels (quintic spline and Wendland C2).

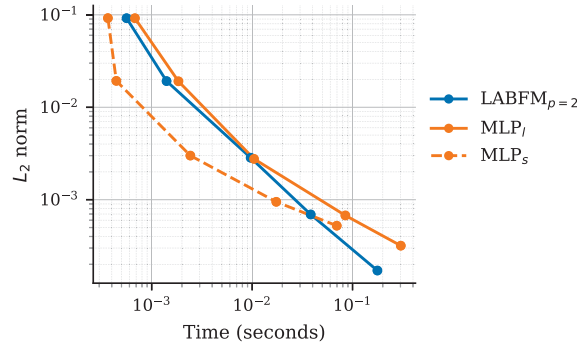


Fig. 10. Evaluation of the trade-off between parameter count, wall-clock forward time, and accuracy for the MLP linear system surrogate and baseline LABFM when computing stencil weights. The vertical axis shows the L_2 error of the x -derivative of the test function in Eq. (13) at different resolutions.

benefit of increased model capacity. MLP_s outperforms LABFM in computational cost over a moderate resolution range, achieving up to a $5\times$ speedup, but eventually plateaus at its limiting error as resolution increases. Within this range, MLP_s achieves comparable accuracy to LABFM at lower cost, demonstrating an efficiency advantage for applications requiring moderate accuracy with a second-order LABFM discretisation. These results, however, are specific to the second-order linear system; extending to higher-order approximations introduces additional challenges, as the conditioning of the linear system deteriorates with increasing polynomial order, which we discuss next.

6. Caveats regarding high-order approximation accuracy

LABFM has a theoretical leading order error of $\mathcal{O}(s_i^{p+1-\zeta})$, where ζ represents the order of the approximated differential operator, and p is the order of polynomial consistency of the differential operator. However, the conditioning of the linear system (Eq. (3)) deteriorates for higher-order approximations, amplifying the impact of numerical errors and increasing the sensitivity of the solution vector to noise; consequently, there is an increase in the limiting error for higher-order approximations even with standard linear system solvers such as LU factorisation [13].

To assess the sensitivity of the LABFM operator to errors in the solution vector at higher orders of approximation, we define a noise bound as $\eta \|\Psi_i^d\|_\infty$, where $\eta = 10^{-7}$. Independent noise values are sampled uniformly from $\mathcal{U}(-\eta \|\Psi_i^d\|_\infty, \eta \|\Psi_i^d\|_\infty)$ and added to each entry of the solution vector Ψ_i^d . This scaling ensures the injected noise is proportional to the solution magnitude while remaining well below the error levels achieved by any model in this study. A convergence analysis is then performed with the noise-perturbed solution vector, with results shown in Fig. 11.

As shown in Fig. 11, even minimal noise in the solution vector is sufficient to disrupt convergence at higher orders, consistent with the conditioning deterioration reported in [13]. The second-order approximation remains convergent across all simulated resolutions. The fourth-order approximation diverges at a resolution of 2×10^{-2} , with a limiting error of $\sim 10^{-5}$, while the sixth-order diverges earlier, at 5×10^{-2} , with a limiting error of $\sim 10^{-4}$. The eighth-order approximation exhibits divergent behaviour throughout the entire resolution range tested. The progressive degradation in both limiting error and onset resolution confirms that higher-order approximations impose increasingly stringent accuracy requirements on the predicted solution vector.

This sensitivity is a direct consequence of the higher-order moment constraints. As the approximation order increases, the solution vector must satisfy additional polynomial terms in the Taylor expansion, further degrading the conditioning of the system and

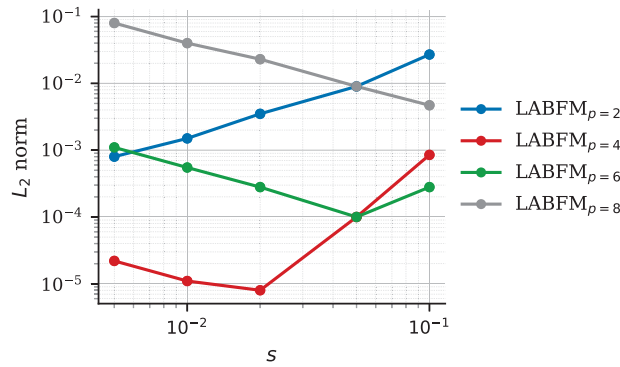


Fig. 11. Convergence study of LABFM approximation of the Laplace operator of Eq. (13) for different orders of approximation with noise introduced in coefficient vector (Ψ_i^d).

rendering the required prediction accuracy increasingly difficult to achieve with current neural network architectures. The limiting factor for the linear-system surrogate at high order is therefore the conditioning of the system itself: the tolerance on the predicted solution vector tightens faster than the network’s achievable accuracy.

The kernel surrogate is not subject to this conditioning constraint, but faces a distinct obstacle: the well-documented spectral bias of neural networks [38,39], which causes them to converge more slowly on high-frequency modes. The qualitative results in Figs. 2–6 confirm that the dominant low-frequency components are accurately learned, producing visually indistinguishable weight fields. At higher orders, the additional monomials produce weight fields with finer-scale spatial structure, and it is precisely these high-frequency components that spectral bias makes hardest for the network to resolve. Spectral bias therefore accounts for the order-dependent degradation, but not for the presence of an error floor itself. The latter reflects an intrinsic limitation of gradient-based training algorithms [40]: Colbrook et al. [41] rigorously demonstrate that even when a stable and accurate network exists in theory, training algorithms face fundamental computational barriers beyond a certain precision threshold, with further improvements requiring disproportionately more data or being unattainable in the worst case. Together, these two mechanisms provide a principled explanation for the error floors observed in both surrogate approaches.

Beyond accuracy, practical deployment introduces a competing constraint: prediction speed. Achieving lower error floors generally demands larger, more computationally expensive models, directly conflicting with the efficiency motivation for surrogating the linear system solve. This trade-off fundamentally limits the operating regime of standard multi-layer perceptrons surrogates for high-order discretisations.

7. Summary & conclusion

Mesh-free numerical methods have gained considerable popularity as a viable simulation framework for complex geometries compared to traditional mesh-based methods. Lagrangian mesh-free methods tend to operate at low-order convergence, whilst in an Eulerian frame, high-order approximations are possible, with the costs of calculating polynomially-consistent kernels incurred only in pre-processing.

In this work, we investigated the application of MLPs to surrogate segments of high-order mesh-free methods, specifically whether MLPs can obtain high-order weights with lower computational cost than current Eulerian mesh-free methods. In the first approach, we trained MLPs to predict the discretised weights in computational stencils from the distance of the support nodes to the central node in a computational stencil. The results of this approach indicate that, although MLPs can qualitatively learn the kernel present in high-order mesh-free methods with reasonable accuracy, the predicted weights fall short in eliminating undesired LABFM moments. Consequently, the ML-predicted outputs introduce amplified low-order errors, leading to relatively high limiting errors, only marginally outperforming uncorrected SPH. Hence, the architectures investigated are not suitable for high-order kernel surrogates.

The second method proposed to accelerate LABFM involved utilising MLPs to solve the low-rank, dense, ill-conditioned and constant right-hand side linear system present in high-order mesh-free methods. Quantitatively, this approach yield significantly more accurate stencil moments than the first approach. The linear system surrogates generate significantly better results than the full discretisation surrogate, with stencil residuals about two orders of magnitude more accurate. The convergence profile of the trained models showed convergence at lower resolution before reaching a higher error limit than that obtained by LABFM. The limiting error of order 10^{-4} and 10^{-3} for the derivative and Laplacian, respectively, was attributed to the residual of the predicted solution of the linear system (i.e. stencil moments) compared to traditional methods, such as LU factorisation. The computational-cost accuracy trade-off study performed indicated that the linear system surrogate can be systematically optimised for accuracy or computational efficiency. In the case of computational efficiency optimisation, it was demonstrated that the linear system surrogate gives the same accuracy as traditional LU factorisation for a second-order approximation at lower computational cost up to the limiting accuracy.

Although the linear systems arising at second order are small, the surrogate remains beneficial for two reasons. First, its cost scales with model size, exposing a cost–accuracy trade-off unavailable to direct solvers, whose cost is fixed. Second, the practical target of this framework is Lagrangian simulation: whereas Eulerian mesh-free methods incur the cost of the consistency-correcting systems only in pre-processing, node motion in a Lagrangian frame requires the stencil weights to be recomputed at every time step for every particle, so any per-stencil cost reduction is multiplied by the number of particles and time steps. Demonstration of the surrogate within a fully dynamic Lagrangian solver is left for future work.

A current limitation is regarding the application of the current framework for high-order approximations. Given the condition of the linear systems deteriorates for higher-order approximations, the linear system must be solved with extremely high precision to achieve even moderate limiting accuracy. Thus, the framework proposed is currently not suitable for higher order approximations. Future directions that may circumvent these limitations include directly approximating the differential operators rather than the discrete weights, or embedding polynomial consistency as a hard constraints.

CRediT authorship contribution statement

Lucas Gerken Starepravo: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Georgios Fourtakas:** Writing – review & editing, Supervision, Investigation, Conceptualization. **Steven Lind:** Writing – review & editing, Methodology, Conceptualization. **Ajay B. Harish:** Supervision. **Jack R.C. King:** Writing – review & editing, Supervision, Methodology, Funding acquisition, Formal analysis, Conceptualization.

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.

Acknowledgements

J.R.C.K. is funded by the Royal Society via a University Research Fellowship (URF\R1\221290). We would like to acknowledge the assistance given by Research IT and the use of the Computational Shared Facility at the University of Manchester.

Appendix A. Smoothed particle hydrodynamics operators

SPH gradient operators admit symmetric and anti-symmetric formulations, where the former guarantees exact conservation and the latter offers greater accuracy. Since the focus here is on consistency, only the anti-symmetric form is considered, which takes the form of Eq. (1) with weights

$$w_{ji}^{\nabla} = \nabla_i W(\mathbf{x}_{ji}, h) V_j, \quad (\text{A.1})$$

where $(\cdot)_{ji} = (\cdot)_j - (\cdot)_i$, so that $\mathbf{x}_{ji}$ denotes the position of node j relative to node i , W is the smoothing kernel, and V_j is the particle volume. For the Laplacian, the Morris operator [42] is adopted:

$$w_{ji}^A = \frac{-2 \mathbf{x}_{ji}}{\|\mathbf{x}_{ji}\|_2} \cdot \nabla_i W_{ji} V_j, \quad (\text{A.2})$$

where $\|\cdot\|_2$ denotes the Euclidean norm.

A.1. Smoothing kernels

Quintic spline: The quintic spline kernel is defined over the support $r \in [0, 3]$ as

$$W^{\text{QS}}(r) = \sigma \begin{cases} (1-r)_+^5 - 6\left(\frac{2}{3}-r\right)_+^5 + 15\left(\frac{1}{3}-r\right)_+^5, & 0 \leq r < 1, \\ (1-r)_+^5 - 6\left(\frac{2}{3}-r\right)_+^5, & 1 \leq r < 2, \\ (1-r)_+^5, & 2 \leq r < 3, \\ 0, & r \geq 3, \end{cases} \quad (\text{A.3})$$

where $r = \|\mathbf{x}_{ji}\|_2/h$ and σ is a normalisation constant dependent on spatial dimension; in two dimensions, $\sigma = \frac{7}{478\pi h^2}$ [43]. With $h = 1.5s$, the quintic spline yields approximately 60–65 neighbours.

Wendland C2: The Wendland C2 kernel is defined over the support $r \in [0, 2]$ as

$$W^{\text{WC2}}(r) = \sigma(1-r)^4(1+4r), \quad (\text{A.4})$$

with $\sigma = \frac{7}{\pi h^2}$ in two dimensions. With $h = 1.5s$, this kernel yields approximately 25–30 neighbours.

Table B.3
Hyperparameters of models trained in this investigation.

Hyperparameter	MLP	MLP _s
Hidden layers	10	5
Neurons/layer	128	64
Epochs	2000	1500
Batch size	512	512
Activation func.	SILU	SILU
Optimiser	Adam	Adam
Loss func.	MSE	MSE
Initial Learning rate	10 ⁻³	10 ⁻³
Optimiser Scheduler	Plateau Scheduler	Plateau Scheduler

Appendix B. Hyperparameter details

This appendix details the architectures and hyperparameters of all models used in this paper. All models were implemented using PyTorch's `nn.Module` class; any unspecified hyperparameter takes the PyTorch default value. Two distinct architectures were used, referred to as MLP and MLP_s, and are summarised in Table B.3. MLP_s is used exclusively for the computational cost analysis in Section 4.2; all other models use the MLP architecture.

Appendix C. Hardware and additional training & inference details

All models were trained on a single NVIDIA GeForce RTX 5080 (16 GB) using PyTorch 2.8.0 with CUDA 12.9. Training times ranged from approximately 1.5 h for the smallest architecture (MLP_s) to 2 h for the largest (MLP). We emphasise that training and data generation are one-off offline costs: once trained, a network is reusable across simulations and physical systems, since the stencil weights depend only on the local node configuration. The offline cost is therefore amortised over all subsequent use, and the relevant cost at simulation time is the inference time reported in Section 5.2.2. For the computational cost analysis, both surrogates and LABFM were timed on a single logical core of an AMD Ryzen 7 7800X3D processor.

In practical Lagrangian simulations, many stencils are processed per time step, and both the direct linear solves [44] and the surrogate inference [35] can efficiently be batched; we adopt the serial comparison as it isolates the per-stencil algorithmic cost from library-, hardware-, and implementation-dependent batching gains, which benefit both approaches. The trade-off identified in Fig. 10 is qualitatively unaffected by parallelisation: the cost of the direct solve is fixed by the numerical algorithm, whereas the surrogate's cost is set by the network size, allowing a smaller architecture to be selected whenever the application tolerates the corresponding limiting error. We note that the relative cost of the two approaches might change under batched CPU or GPU execution; however, since network size remains a free design parameter, the architecture can be adjusted accordingly to recover a cost advantage at the limiting error acceptable to the application. A quantitative comparison of batched direct linear solves and batched surrogate inference is left for future work.

Data availability

The data supporting the findings of this study will be submitted along with the manuscript. The code used to train and test the framework is available at <https://github.com/uom-complexfluids/ml4meshfree.git>.

References

- [1] R.A. Gingold, J.J. Monaghan, Smoothed particle hydrodynamics: theory and application to non-spherical stars, *Mon. Not. R. Astron. Soc.* 181 (3) (1977) 375–389, <http://dx.doi.org/10.1093/mnras/181.3.375>.
- [2] L.B. Lucy, A numerical approach to the testing of the fission hypothesis, *Astron. J.* 82 (1977) 1013–1024, <http://dx.doi.org/10.1086/112164>.
- [3] G.R. Liu, Y.T. Gu, *Meshfree Shape Function Construction*, Springer Netherlands, 2005, pp. 54–144, http://dx.doi.org/10.1007/1-4020-3468-7_3.
- [4] S.J. Lind, B.D. Rogers, P.K. Stansby, Review of smoothed particle hydrodynamics: towards converged Lagrangian flow modelling, *Proc. R. Soc. A: Math. Phys. Eng. Sci.* 476 (2241) (2020) 20190801, <http://dx.doi.org/10.1098/rspa.2019.0801>.
- [5] W.K. Liu, S. Jun, Y.F. Zhang, Reproducing kernel particle methods, *Internat. J. Numer. Methods Fluids* 20 (8–9) (1995) 1081–1106, <http://dx.doi.org/10.1002/fld.1650200824>.
- [6] P.S. Jensen, Finite difference techniques for variable grids, *Comput. Struct.* 2 (1) (1970) 17–29, [http://dx.doi.org/10.1016/0045-7949\(72\)90020-X](http://dx.doi.org/10.1016/0045-7949(72)90020-X).
- [7] J.J. Benito, F. Ureña, L. Gavete, Solving parabolic and hyperbolic equations by the generalized finite difference method, *J. Comput. Appl. Math.* 209 (2) (2006) 208–233, <http://dx.doi.org/10.1016/j.cam.2006.10.090>.
- [8] N. Trask, M. Perego, P. Bochev, A high-order staggered meshless method for elliptic problems, *SIAM J. Sci. Comput.* 39 (2) (2017) A479–A502, <http://dx.doi.org/10.1137/16M1055992>.
- [9] N. Trask, M. Maxey, X. Hu, A compatible high-order meshless method for the Stokes equations with applications to suspension flows, *J. Comput. Phys.* 355 (2017) 310–326, <http://dx.doi.org/10.1016/j.jcp.2017.10.039>.
- [10] B.J. Gross, N. Trask, P. Kuberry, P.J. Atzberger, Meshfree methods on manifolds for hydrodynamic flows on curved surfaces: A generalized moving least-squares (GMLS) approach, *J. Comput. Phys.* 409 (2020) 109340, <http://dx.doi.org/10.1016/j.jcp.2020.109340>.
- [11] N. Flyer, B. Fornberg, V. Bayona, G. A. Barnett, On the role of polynomials in RBF-FD approximations: I. Interpolation and accuracy, *J. Comput. Phys.* 321 (2016) 21–38, <http://dx.doi.org/10.1016/j.jcp.2016.05.026>, URL <https://www.sciencedirect.com/science/article/pii/S0021999116301632>.

- [12] J.R.C. King, S. Lind, A. Nasar, High order difference schemes using the Local Anisotropic Basis Function Method, *J. Comput. Phys.* 415 (2020) 109549, <http://dx.doi.org/10.1016/j.jcp.2020.109549>.
- [13] J.R.C. King, S.J. Lind, High-order simulations of isothermal flows using the local anisotropic basis function method (LABFM), *J. Comput. Phys.* 449 (2022) 110760, <http://dx.doi.org/10.1016/j.jcp.2021.110760>.
- [14] J.R.C. King, A mesh-free framework for high-order direct numerical simulations of combustion in complex geometries, *Comput. Methods Appl. Mech. Engrg.* 421 (2024) 116762, <http://dx.doi.org/10.1016/j.cma.2024.116762>.
- [15] M. Shehab, L. Abualigah, Q. Shambour, M.A. Abu-Hashem, M.K.Y. Shambour, A.I. Alslibi, A.H. Gandomi, Machine learning in medical applications: A review of state-of-the-art methods, *Comput. Biol. Med.* 145 (2022) 105458, <http://dx.doi.org/10.1016/j.combiomed.2022.105458>.
- [16] T. Duriez, S.L. Brunton, B.R. Noack, *Machine Learning Control (MLC)*, Springer International Publishing, Cham, 2017, pp. 11–48, http://dx.doi.org/10.1007/978-3-319-40624-4_2.
- [17] A.I. Khan, S. Al-Habsi, Machine learning in computer vision, *Procedia Comput. Sci.* 167 (2020) 1444–1451, <http://dx.doi.org/10.1016/j.procs.2020.03.355>.
- [18] J. Li, X. Du, J.R.R.A. Martins, Machine learning in aerodynamic shape optimization, *Prog. Aerosp. Sci.* 134 (2022) 100849, <http://dx.doi.org/10.1016/j.paerosci.2022.100849>.
- [19] P.P. Ray, ChatGPT: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope, *Internet Things Cyber-Physical Syst.* 3 (2023) 121–154, <http://dx.doi.org/10.1016/j.iotcps.2023.04.003>.
- [20] A. Kratsios, The Universal Approximation Property, *Ann. Math. Artif. Intell.* 89 (5) (2021) <http://dx.doi.org/10.1007/s10472-020-09723-1>.
- [21] C. Caron, P. Lauret, A. Bastide, Machine learning to speed up computational fluid dynamics engineering simulations for built environments: A review, *Build. Environ.* 267 (2025) 112229, <http://dx.doi.org/10.1016/j.buildenv.2024.112229>, URL <https://www.sciencedirect.com/science/article/pii/S0360132324010710>.
- [22] E. Amato, V. Zago, C.D. Negro, A physically consistent AI-based SPH emulator for computational fluid dynamics, *Nonlinear Eng.* 13 (1) (2024) 20220359, <http://dx.doi.org/10.1515/nleng-2022-0359>.
- [23] N. Zhang, S. Yan, Q. Ma, Q. Li, A hybrid method combining ISPH with graph neural network for simulating free-surface flows, *Comput. Phys. Comm.* 301 (2024) 109220, <http://dx.doi.org/10.1016/j.cpc.2024.109220>.
- [24] Z. Jiang, J. Jiang, Q. Yao, G. Yang, A neural network-based PDE solving algorithm with high precision, *Sci. Rep.* 13 (1) (2023) 4479, <http://dx.doi.org/10.1038/s41598-023-31236-0>.
- [25] Y. Gu, M.K. Ng, Deep neural networks for solving large linear systems arising from high-dimensional problems, *SIAM J. Sci. Comput.* 45 (5) (2023) A2356–A2381, <http://dx.doi.org/10.1137/22M1488132>.
- [26] A. Hamrani, A. Akbarzadeh, C.A. Madramootoo, F.Z. Bouarab, Machine learning surrogate modeling for meshless methods: Leveraging universal approximation, *Int. J. Comput. Methods* 20 (8) (2023) 2141022, <http://dx.doi.org/10.1142/S021987622141022X>.
- [27] N. McGreivy, A. Hakim, Weak baselines and reporting biases lead to overoptimism in machine learning for fluid-related partial differential equations, *Nat. Mach. Intell.* (6) (2024) <http://dx.doi.org/10.1038/s42256-024-00897-5>.
- [28] G. Leech, J.J. Vazquez, N. Kupper, M. Yagudin, L. Aitchison, Questionable practices in machine learning, 2024, <http://dx.doi.org/10.48550/arXiv.2407.12220>.
- [29] C. Shu, H. Ding, K.S. Yeo, Local radial basis function-based differential quadrature method and its application to solve two-dimensional incompressible Navier–Stokes equations, *Comput. Methods Appl. Mech. Engrg.* 192 (7) (2003) 941–954, [http://dx.doi.org/10.1016/S0045-7825\(02\)00618-7](http://dx.doi.org/10.1016/S0045-7825(02)00618-7).
- [30] T. Cecil, J. Qian, S. Osher, Numerical methods for high dimensional Hamilton–Jacobi equations using radial basis functions, *J. Comput. Phys.* 196 (1) (2004) 327–347, <http://dx.doi.org/10.1016/j.jcp.2003.11.010>.
- [31] I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*, MIT Press, 2016, URL <http://www.deeplearningbook.org>.
- [32] V. Shankar, G.B. Wright, R.M. Kirby, A.L. Fogelson, A radial basis function (RBF)-finite difference (FD) method for diffusion and reaction-diffusion equations on surfaces, 2014, [arXiv:1404.0812](https://arxiv.org/abs/1404.0812). URL <https://arxiv.org/abs/1404.0812>.
- [33] J.J. Benito, F. Ureña, L. Gavete, Influence of several factors in the generalized finite difference method, *Appl. Math. Model.* 25 (2001) 1039–1053, [http://dx.doi.org/10.1016/S0307-904X\(01\)00029-4](http://dx.doi.org/10.1016/S0307-904X(01)00029-4).
- [34] L. Huang, J. Qin, Y. Zhou, F. Zhu, L. Liu, L. Shao, Normalization techniques in training DNNs: Methodology, analysis and application, 2020, <http://dx.doi.org/10.48550/arXiv.2009.12836>.
- [35] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: An imperative style, high-performance deep learning library, 2019, URL <https://arxiv.org/abs/1912.01703>.
- [36] M. Lin, G. Fournakis, B.D. Rogers, A novel high-order spectral incompressible smoothed particle hydrodynamics (ISPH) scheme with an immersed boundary method (IBM), *J. Comput. Phys.* 540 (2025) 114264, <http://dx.doi.org/10.1016/j.jcp.2025.114264>.
- [37] N. Flyer, G.A. Barnett, L.J. Wicker, Enhancing finite differences with radial basis functions: Experiments on the Navier–Stokes equations, *J. Comput. Phys.* 316 (2016) 39–62, <http://dx.doi.org/10.1016/j.jcp.2016.02.078>, URL <https://www.sciencedirect.com/science/article/pii/S0021999116300195>.
- [38] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F.A. Hamprecht, Y. Bengio, A. Courville, On the spectral bias of neural networks, 2019, URL <https://arxiv.org/abs/1806.08734>.
- [39] Z.Q.J. Xu, Y. Zhang, T. Luo, Y. Xiao, Z. Ma, Frequency principle: Fourier analysis sheds light on deep neural networks, *Commun. Comput. Phys.* 28 (5) (2020) 1746–1767, <http://dx.doi.org/10.4208/cicp.0a-2020-0085>.
- [40] H. Boche, A. Fono, G. Kutyniok, Limitations of deep learning for inverse problems on digital hardware, *IEEE Trans. Inform. Theory* 69 (12) (2023) 7887–7908, <http://dx.doi.org/10.1109/tit.2023.3326879>.
- [41] M.J. Colbrook, V. Antun, A.C. Hansen, The difficulty of computing stable and accurate neural networks: On the barriers of deep learning and smale’s 18th problem, *Proc. Natl. Acad. Sci.* 119 (12) (2022) e2107151119, <http://dx.doi.org/10.1073/pnas.2107151119>, URL <https://www.pnas.org/doi/abs/10.1073/pnas.2107151119>.
- [42] J.P. Morris, P.J. Fox, Y. Zhu, Modeling low Reynolds number incompressible flows using SPH, *J. Comput. Phys.* 136 (1) (1997) 214–226, <http://dx.doi.org/10.1006/jcph.1997.5776>.
- [43] M.B. Liu, G.R. Liu, Smoothed particle hydrodynamics (SPH): An overview and recent developments, *Arch. Comput. Methods Eng.* 17 (1) (2010) 25–76, <http://dx.doi.org/10.1007/s11831-010-9040-7>.
- [44] J. King, A mesh-free framework for high-order direct numerical simulations of combustion in complex geometries, *Comput. Methods Appl. Mech. Engrg.* 421 (2024) 116762, DOI: [j.cma.2024.116762](https://doi.org/10.1016/j.cma.2024.116762). URL <https://www.sciencedirect.com/science/article/pii/S0045782524000185>.