Computers and Mathematics with Applications 164 (2024) 79–94

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

Node subsampling for multilevel meshfree elliptic PDE solvers

Andrew P. Lawrence [a,*], Morten E. Nielsen [b], Bengt Fornberg [a]

[a] Department of Applied Mathematics, University of Colorado Boulder, Boulder, CO, 80309, USA
[b] DHI, 2970, Hørsholm, Denmark

ARTICLE INFO

Dataset link: https://codeocean.com/capsule/3980307/tree

Keywords:
Node set
Coarsening
Multilevel
Meshfree
RBF-FD
PDE

ABSTRACT

Subsampling of node sets is useful in contexts such as multilevel methods, computer graphics, and machine learning. On uniform grid-based node sets, the process of subsampling is simple. However, on node sets with high density variation, the process of coarsening a node set through node elimination is more interesting. A novel method for the subsampling of variable density node sets is presented here. Additionally, two novel node set quality measures are presented to determine the ability of a subsampling method to preserve the quality of an initial node set. The new subsampling method is demonstrated on the test problems of solving the Poisson and Laplace equations by multilevel radial basis function-generated finite differences (RBF-FD) iterations. High-order solutions with robust convergence are achieved in linear time with respect to node set size.

1. Introduction

Subsampling of variable density node sets has applications in polynomial approximation, numerical integration, artificial intelligence, machine learning, multilevel methods, and computer graphics. For each of these applications, algorithms exist in 1D, 2D, and even N-D space, but their utility is often application specific.

Subsampling methods have been specifically developed to choose points optimized for global polynomial approximation and numerical integration [1–4]. Node sets have been optimized for global RBF collocation methods using multi-objective optimization [5]. However, the coarse node sets which these algorithms produce do not, in general, preserve the variable density of the initial, fine node sets.

In the context of data driven artificial intelligence and machine learning, the process of tuning data rather than tuning model parameters has driven research on subsampling [6–9]. With the exception of the generalized diversity subsampling algorithm in [8], these algorithms are either designed for uniform subsampling or statistical learning techniques such that they are not well-suited to the preservation of variable density data sets.

Research in computer graphics has led to considerable developments in the area of Poisson disk sampling which serves to subsample variable density node sets, producing resultant node sets with desirable statistical and minimum spacing properties [10,11]. The process of Poisson disk sampling is recast as a weighted sample elimination or weighted subsampling problem in [12]. Other efforts have employed Poisson disk sampling to produce heirarchical node sets for multilevel methods using RBFs [13], albeit on uniform density node sets.

Subsampling algorithms are an integral part of multilevel methods. Algebraic multilevel methods (AMM) provide robust and scalable linear solvers for a wide class of problems. They are in principle a natural choice for meshfree domain discretizations since the constituent hierarchical levels are a natural byproduct of the inter-level transfer and coarse level operators. In the context of meshfree systems, AMM has been applied to methods that don't use RBF-FD [14] [15] and those that do [16]. For solvers which use RBFs, it has been shown that geometric multilevel methods (GMMs) converge in fewer iterations [16]. Additionally, the set-up time for AMM is higher overall [17] [18]. The construction of the coarse levels themselves is higher in GMM, but that cost is reduced for a meshfree domain and motivates the need for a fast subsampling algorithm as explored in the following sections. Tests run in [18] demonstrate that AMM is sensitive to the mesh variation and resolution on the coarsest level. The

The code (and data) in this article has been certified as Reproducible by Code Ocean: https://codeocean.com/. More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.
E-mail address: anla5397@colorado.edu (A.P. Lawrence).

https://doi.org/10.1016/j.camwa.2024.03.022
Received 18 May 2023; Received in revised form 27 February 2024; Accepted 17 March 2024
Available online 8 April 2024
0898-1221/Published by Elsevier Ltd.

proper choice of parameters (the strength parameter in particular) for AMM can reduce the total computation time by 15–40%, per [18]. The GMMs have no such parameter sensitivity and have less sensitivity to mesh variation. According to [19], when using GMM and AMM as preconditioners for Krylov methods, the scheme will converge more quickly for preconditioned matrices for which the spectrum is more heavily clustered toward one. This corresponds to coefficient matrices[1] with spectra clustered at zero. In the problems considered in [16], the spectra for GMM were more clustered around one than the compared AMM method (PyAMG [20]) in all cases. For these reasons, AMMs are not considered here.

Additionally, the use of a GMM over a variable density node set requires a subsampling routine which maintains the variable density of the original node set. AMMs coarsen the operators themselves and each coarsened level beyond the original has no intuitive geometric meaning or interpretation [21–23]. As such, coarsening methods for AMMs are not useful for GMMs [24].

The combination of GMMs with meshfree solvers for partial differential equations has become increasingly popular. Meshfree methods such as radial basis function-generated finite differences (RBF-FD) discretize at scattered (quasi-uniform[2] nodes rather than with meshes. RBF-FD methods, in particular, allow for high geometric flexibility and can benefit from high density variation but require the underlying node sets to meet certain quality constraints in order to ensure stability and accuracy of the solution [25–27]. Robust algorithms for generating such node sets exist [28–30] and are utilized in this paper. The application of meshfree partial differential equation (PDE) solvers within a multilevel scheme requires a similarly robust algorithm for coarsening node sets [31,32]. When implementing a multilevel algorithm, one typically starts with the initial, fine node set. Given a desired level of refinement, the task of producing a coarse node set from a fine node set can be accomplished in one of two ways: one can either select a subset of the fine node set or generate a node set that is independent of the fine node set. Many methods exist to create coarse node sets which are visually and geometrically representative of the original node set and are not subsets of the initial, fine node set [33–35]. However, the operators to coarsen and refine between independent node sets require more memory. Alternatively, selecting a subset simplifies the coarsening and refining operators and requires less memory. The combination of a multilevel method with RBFs has been explored before, however, primarily on uniform (Cartesian grid) or uniformly distributed scattered node sets [13–15,36–39]. The use of multilevel techniques on RBF-FD meshfree solvers for PDEs over variable density node sets is explored in [40], however the subsampling routine used therein, based on [41], is not adjustable to coarse node sets of any size; it is limited to coarsening by factors of $1/n, n \in \mathbb{N}$. Due to this limitation, it is not considered in this paper. Though not applicable in its original form (as it relies on information from a mesh at the fine level), an extension of the algorithm found in [42] can be applied to meet the outlined needs for a variable density node set subsampling algorithm. However, it also suffers from inflexible coarsening factors and, as such, is not considered here. The multilevel meshfree PDE solver presented here achieves high-order solutions with robust convergence in linear time with respect to node set size.

In contrast to the process of generating coarser node set from an initial fine node set, one might consider an initial coarse node set and the generation of finer node sets. Most refining algorithms use some residual function to determine if refinement should take place [43–45]. Most refinement techniques require user-supplied criteria in the form of a residual or principle function [46] at which some set of test nodes are evaluated based on the residual to determine where an how to refine. These function evaluations are an increased computational cost. Additionally, the goodness of the refinement depends heavily on the proposed test nodes. Simple ways of determining these nodes, such as using the halfway points between existing nodes [47], may not apply well enough to variable density node sets. On the other hand, more robust methods such as determining the Voronoi nodes [48] or the centroids of a node and it's K nearest neighbors [49] may still be ill suited for variable density refinement and introduce more significant increases in computational cost. Other refinement methods are limited to uniform refinement which is not appropriate for our current applications [34,35]. Ultimately, refinement techniques will not be considered here.

Throughout this paper, the terms subsampling and coarsening will be used to refer to the process of selecting a subset from a collection of nodes or points. The subsampling algorithms considered in this paper are outlined in Section 2, boundary considerations for subsampling routines are covered in Section 4, numerical tests and comparisons between those presented earlier are presented in Section 3. Additionally, Section 5 includes two examples of a meshfree multilevel RBF-FD PDE solver utilizing the novel moving front node subsampling method from Section 2.1.

2. Subsampling algorithms

This section surveys the methodology of four subsampling algorithms. In addition to a novel moving front method presented in Section 2.1, a weighted subsampling method based on [12], a method based on Poisson disk sampling, and the generalized diversity subsampling method found in [8] are presented in Sections 2.2, 2.3, and 2.4 respectively.

2.1. Moving front

The novel subsampling algorithm presented here is a streamlined application of a 'moving front' strategy akin to those found in the node generation algorithms in [30] and [28]. Algorithm 1 begins by sorting all nodes in the fine node set according to an arbitrary direction[3]; for example, from the bottom to the top. Then, the k nearest neighbors to each node are determined.[4] For each node in the fine node set and working in the chosen direction, first check if the node has already been marked. If it has been marked, continue on to the next node. If it has not been marked, mark each of the k nearest neighbors that is both above the present node in the sort and within a distance $c * D_1$ (for example $c = 1.5$) of the present node, where D_1 is the distance to the first nearest neighbor. All marked nodes are then removed to produce the coarse node set. The value $k = H(n+1) - 1 = 3n(n+1)$ where $H(n)$ is the n^{th} centered hexagonal number, and $n = \lceil c \rceil$. This value of k represents the maximal number of nodes which can be packed into a ball of radius $n * D_1$ excluding the central node such that no more that k nodes could be within a distance of $n * D_1$ of the center node. The moving front algorithm generalizes immediately to any number of space dimensions: nodes are sorted based upon chosen directions, nearest neighbors are found, and nodes are marked based on proximity. A Python code for the moving front algorithm can be found in Appendix A.1.

[1] Those representing the application of one V-cycle of either GMM or AMM.

[2] Quasi-uniform is here used to refer to node sets which may vary in density globally yet are near-uniform locally, i.e. there exists some $c > 0$ such that for any discretization of the domain, $h^{(n)}_{max}/h^{(n)}_{min} < c$ where $h^{(n)}_{max}$ and $h^{(n)}_{min}$ are the maximum and the minimum distances, respectively, to the nearest neighbor for some discretization n.

[3] The directional sorting and progression of the algorithm enable a cost savings in that only the nodes above the present one need to be searched.

[4] Achieved for k neighbors and N nodes in $O(N \log N)$ operations by the kd-tree algorithm when $k << N$ as is true for this algorithm.

Algorithm 1 Moving Front Algorithm.

```
function MFSub(X_fine = {x_1,...,x_N}, c)
    k = H(⌈c⌉)
    Sort the nodes in X_fine
    Find the indices and distances of the k nearest neighbors for each point in X_fine
    for i = 1 : N do
        if the node x_i has not already been marked then
            Determine which nearest neighbors are within a radius of c times the distance to the nearest neighbor
            Of those, determine which are 'above' x_i in the sort order
            Mark these nodes
        end if
    end for
    Remove the marked nodes from X_fine to produce X_coarse return X_coarse
end function
```

Here it should be reiterated that intrinsic to the moving front algorithm is a directional bias. More specifically, as the algorithm proceeds across a node set, the resultant node set will differ based on the direction in which the moving front travels. The effects of this directional bias are insignificant, however, as shown in the Sections 4.2 and 5.3.

2.2. Weighted subsampling

The weighted subsampling compared with here is based on the work presented in [12], but modified for variable density node sets.[5] Each node is assigned a weight based on its distance to its nearest neighbors. The algorithm then iterates to remove the node with the highest weight, adjust the remaining weights accordingly, and repeat until the desired number of nodes remain. The code for this implementation can be found on the author's GitHub, [50].

2.3. Poisson disk subsampling

Given a radius of exclusion for each node (such that the radii of exclusion are spatially variable), a node is randomly selected from the fine node set and accepted into the coarse node set if its radius of exclusion does not overlap with that of any of the previously accepted nodes. The first node in the coarse node set is chosen randomly. The radii of exclusion are the product of the distance of the nearest neighbor in the fine node set and a hyperparameter h, $r_e(x_i) = h * r_{min}(x_i)$. While the choice of exclusion radii is the same as that of the moving front algorithm, the Poisson disk subsampling algorithm requires that for each nearest neighbor, an overlap of exclusion radii be checked. Additionally, the moving front algorithm sorts the nodes so as to only check the distance of those nodes which have not already been iterated through, thereby improving efficiency. This algorithm is similar to the thinning method presented in [29] and the Poisson thinning method presented in [13]. The use of a nearest neighbor search to support spatially variable radii of exclusion limits the computational complexity to being no better than $O(N \log N)$ in contrast to the $O(N)$ algorithm in [51]. The code for this implementation can be found on the author's GitHub, [50].

2.4. Generalized diversity subsampling

The generalized Diversity Subsampling algorithm as found in [8] selects a subsample from the fine node set according to an arbitrary, specified distribution. The distribution utilized in this paper is a function of the distance to the nearest neighbor of each node.

3. Comparisons of subsampling methods

This section compares the performance in preserving node density variation through iterative coarsening of the four subsampling algorithms found in Section 2. For each example, the primary node set has been generated from the trui image, see Fig. 1(a), using the node generation algorithm from [30], see Fig. 1(b). While this initial node set does not have immediate application to solving PDEs, its radically varying node densities make the trui image a good test problem for visually spotting any algorithmic artifacts. The trui image also contains regions of uniform density, thus illustrating subsampling capabilities on regions of both locally variable and locally uniform node densities.

3.1. Heuristic comparison

This section provides visualizations of the subsampled node sets of each algorithm. Each algorithm is applied twice to the dithered trui image as seen in Fig. 2. The original node set contains 36303 nodes, the first subsample contains 10553 nodes, and the second subsample contains 3404 nodes. Each algorithm discussed, excluding the generalized diversity subsampling algorithm,[6] requires a parameter, c, to control the level of coarsening. To reproduce the subsamples in Fig. 2, the values of c used in each algorithm are listed in Table 1. The moving front algorithm also relies on a choice of nearest neighbors which was $k = 10$ for these tests.

A heuristic comparison between the subsamplings iterations primarily demonstrates the visual goodness of the first three algorithms over the generalized diversity subsampling algorithm. Among the remaining three, the woman's nostrils are more distinct in the moving front and Poisson disk algorithms than in the weighted subsampling while the Poisson disk algorithm seems to preserve the mouth slightly more clearly by the second

[5] A sampling example presented in [12] should, in principle, serve to subsample variable density node sets. However, after repeated attempts, the example was not reproducible.

[6] The generalized diversity subsampling algorithm explicitly requires a target number of nodes as input rather than a parameter.

(a) Original trui.png image (b) Dithered trui image

Fig. 1. The original trui.png image and a dithered version with 36,303 nodes, obtained by the algorithm in [30].

Table 1
The parameters, c, for reproducing the node sets in Fig. 2 for the moving front (MF), weighted (W), and Poisson disk (PD) subsampling algorithms. The generalized diversity subsampling algorithm explicitly relies on the desired number of nodes in the coarse node set and thus has no parameter listed here. The moving front algorithm also relies on a choice of nearest neighbors which was $k = 10$ for these tests. These values are arbitrary, having been chosen only to achieve the same node count across methods.

Method	First Subsampling	Second Subsampling
MF	1.50976	1.515
W	3.44	3.1
PD	1.51	1.521

subsampling. Additionally, the moving front and Poisson disk algorithms preserve a higher level of clarity in the patterns[7] in the trui scarf than the weighted subsampling algorithm. Again, the moving front and Poisson disk subsampling algorithms each better preserve the density disparity between areas of low and high node density in the original dithering[8] than do the weighted or generalized diversity subsampling algorithms. Finally, the Poisson disk algorithm may have a tendency to subsample too aggressively in places.[9] It should be noted that no direction bias of the moving front algorithm is visible.

3.2. Node quality measures

While visually comparing the results of each algorithm is useful, ultimately a more rigorous and quantitative comparison is desirable. As such, a variety of node quality measures are commonly discussed when comparing uniform subsampling methods [8,30]. However, metrics which describe the quality of spatially variable node densities are less common [30]. One way of evaluating the quality of a variable density node set on its own is through local regularity distribution of distance to the nearest k neighbors $\delta_{i,j}$, $i = 1, 2, \ldots k$ for each node x_j.

It is here considered given that the node generation algorithm will produce an initial node set which is sufficiently good.[10] The responsibility of a subsampling algorithm is then to preserve the characteristics of the original node set. A natural way to determine how well any subsampling preserves the quality of the initial node set is to measure the coarse node set in comparison to the fine node set.

The two novel measures presented here are straightforward extensions of the commonly accepted measures of local regularity for evaluating the quality of variable density node sets and are referred to as measures of comparative local regularity (CLR). These CLRs contrast the average or standard deviation of distances to the k nearest neighbors of the fine node set from those of the coarse node set. For each node set X, the Euclidian distance between each node $x_j \in X$ and its k nearest neighbors is calculated as $\delta_{i,j} = \|x_j - x_{i,j}\|$. The average $\overline{\delta}_j$ or standard deviation σ_j of these distances is then found for each node $x_j \in X$. These are typical measures of local regularity. However, the present goal is measure how similar the initial and subsampled node sets are. To this end, the distributions of $\overline{\delta}_j^{\text{fine}}$ and $\overline{\delta}_j^{\text{coarse}}$ over each of the fine and coarse node sets, respectively, are first normalized to be between 0 and 1. Then the difference between these values is calculated at each of the nodes which is collocated in both the coarse node set and the fine node set.[11] Finally, the standard L_2 norm is taken to produce a measure of CLR, which will depend on k. The same process can be applied using the standard deviation σ_j of those distances $\delta_{i,j}$.

Given that these CLRs measure the difference between a given initial node set and a subsampling of it, it is ideal to minimize these measures across algorithms. The presented CLRs should only be compared between subsamplings of the same size and from the same initial node set. Upon

[7] The checkered pattern in the scarf on the woman's right is a clear example.
[8] The disparity is most notable between the woman's left cheek and hair and between the light stripe in the scarf on the woman's right and any of the surrounding regions.
[9] The light stripe on the scarf's right side (left side of the image) is much sparser than in the moving front or weighted subsampling algorithms.
[10] Where goodness is determined by any number of node quality measures chosen based upon context.
[11] Effectively, $\overline{\delta}_j^{\text{fine}}$ needs only be calculated at those nodes x_j in the fine node set which also belong to the coarse node set.

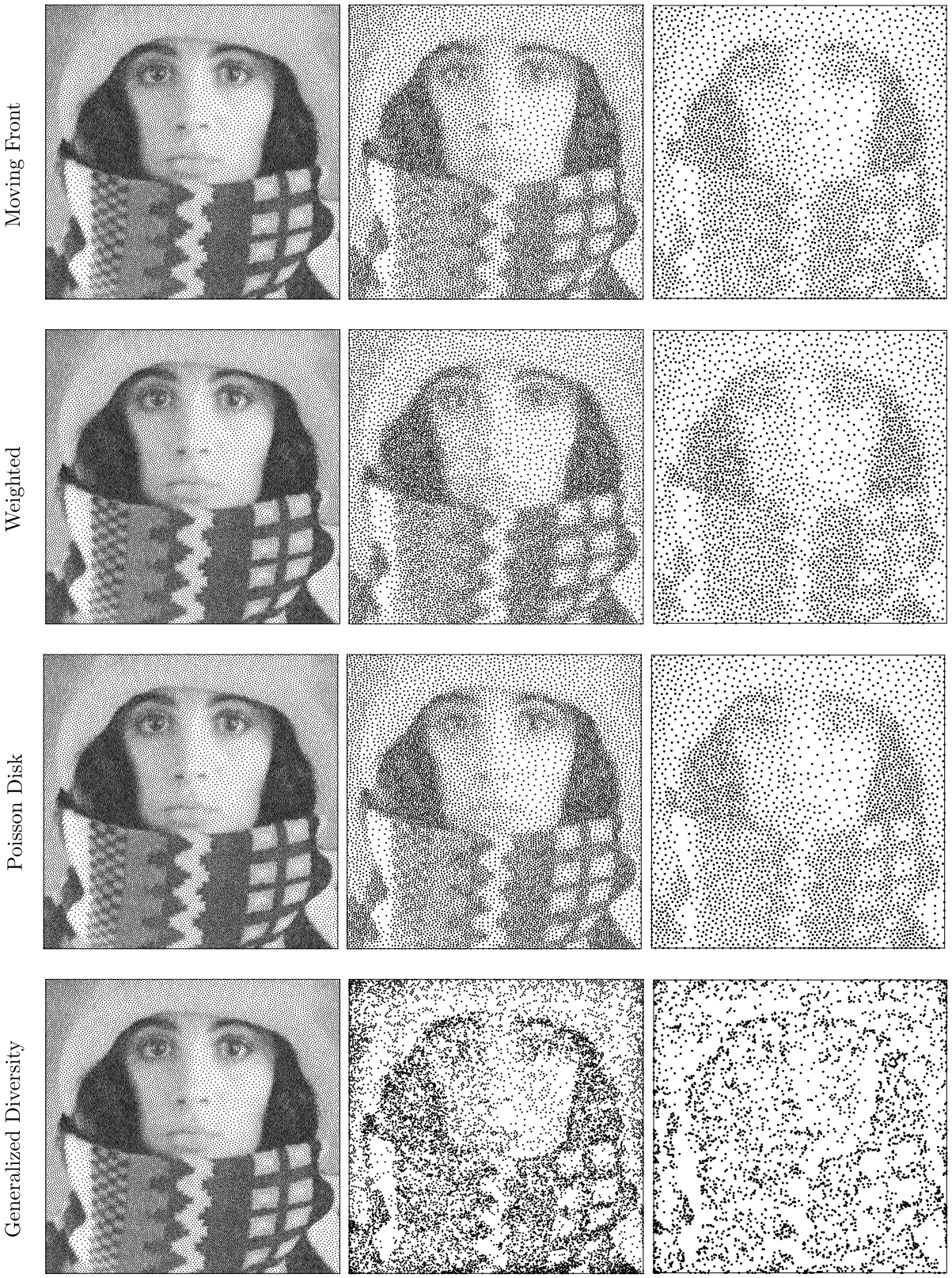


Fig. 2. A visualization of the initial dithered trui image (36303 nodes), the first subsampling (10553 nodes), and the second subsampling (3404 nodes). From top to bottom, the rows demonstrate the moving front, weighted, Poisson disk, and generalized diversity subsampling algorithms. The first column of the figure contains a redundant image of the initial dithering for convenience in comparison.

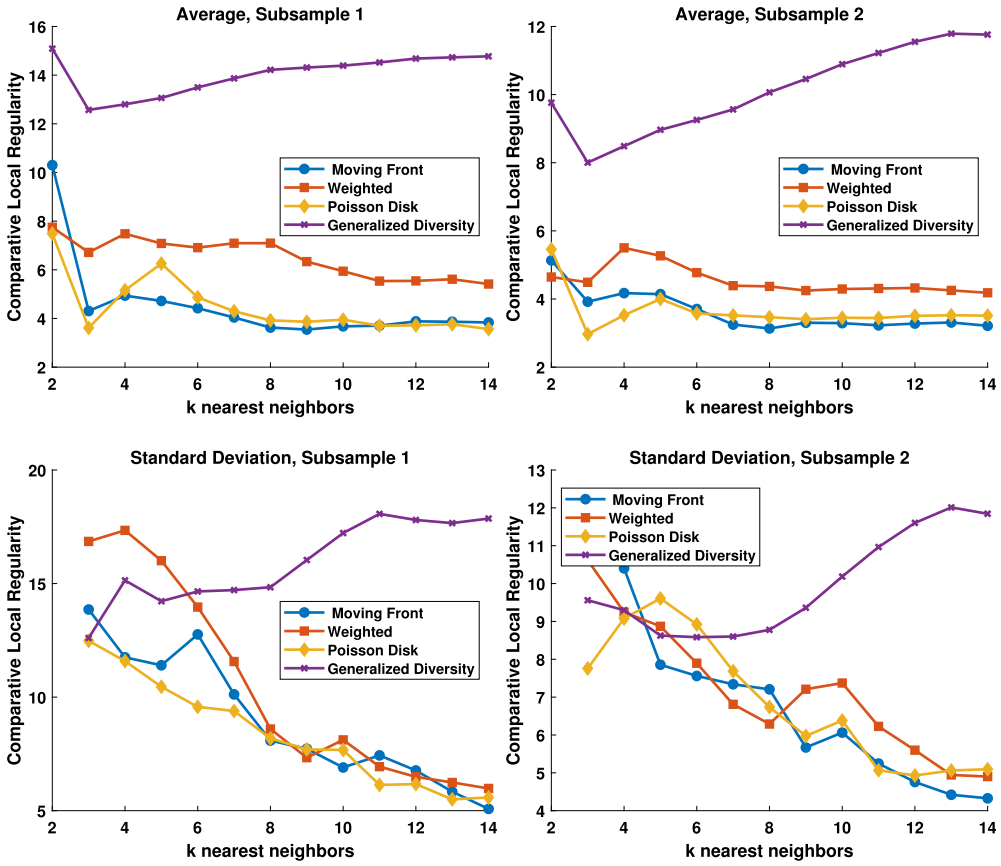


Fig. 3. The comparative local regularity (CLR) of the average distance and standard deviation of distances for $k = 2, 3, \ldots, 14$ nearest neighbors of various subsampling methods (weighted, moving front, Poisson disk, and generalized diversity subsampling) applied once and twice to the dithered trui image. For both measures of CLR, a lower value is better.

comparison of the average and standard deviation CLRs for each algorithm across various values of nearest neighbors as seen in Fig. 3, it is clear that the moving front and Poisson disk algorithms preserve the characteristics of the initial dithered trui image better than the other two algorithms. This behavior is consistent across various sizes of node sets. Between them, however, it is not clear which one is qualitatively better.

3.3. Computational cost

When evaluating any numerical scheme, computational cost must be considered as much as any other aspect of performance. The moving front, weighted, and Poisson disk subsampling algorithms presented in this paper were coded in MATLAB[12] The generalized diversity subsampling algorithm is coded in Python as presented in [8]. A comparison of the computational complexity can be found in Fig. 4. The average execution times in Fig. 4 are calculated based on ten repetitions of each algorithm subsampling from the node set size of the previous data point to the node set size for the given point using the `timeit` commands native to MATLAB and Python. The moving front algorithm is clearly the fastest algorithm of those compared here. The significant computational cost savings secures the moving front algorithm as the best overall performing algorithm. The computations were performed on an AMD Milan, 2.8 GHz processor with Linux operating system.

4. Boundary considerations

Before the moving front algorithm is applied in test problems in Section 5 this section will illustrate the potential pitfalls that can occur if boundary nodes are included in the domain node set without being handled separately and to propose methods for overcoming those pitfalls. Initial node sets are generated by [30] and nodes near the boundary have been repelled[13] prior to any subsampling.

4.1. Subsample boundary with domain

When applying the moving front algorithm naively to a set for which the boundary nodes are included in the domain, the top boundary nodes are undesirably subsampled faster than the ones at the lower boundary, as demonstrated in Fig. 5(a). One way to reduce the subsampling inconsistencies is to include nodes interior and exterior to each boundary as seen in Fig. 5(b).

[12] The code for the moving front algorithm included in Appendix A.1 is provided in Python for the readers convenience. Code is available in both MATLAB and Python on the authors GitHub [50].
[13] Following the repel methodology described in [28].

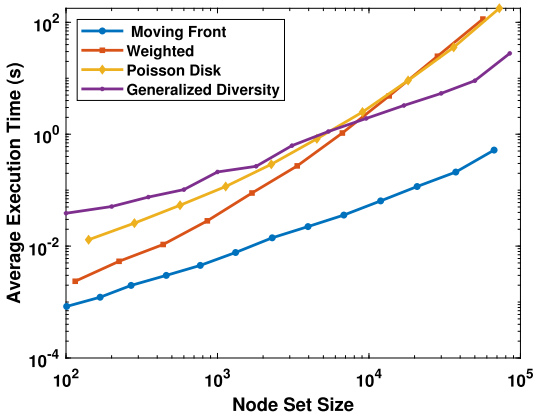
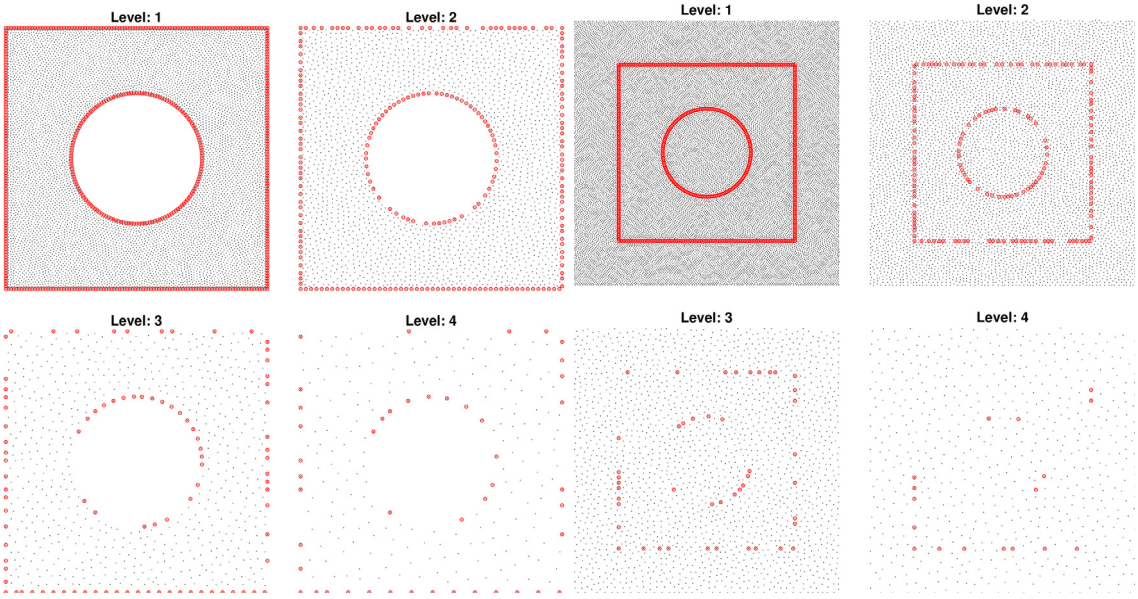


Fig. 4. The computation time of each subsampling algorithm. The node set size is that of each resultant subsample. Each subsampling is serial in nature such that the coarse node set of the previous iteration is the fine node set of the next. The execution time was averaged over ten iterations of the moving front, weighted, Poisson disk, and generalized diversity subsampling algorithms. Note the logarithmic scales. The moving front algorithm is significantly faster than any of the other algorithms.



(a) Naive algorithm (b) Nodes included interior and exterior to boundaries

Fig. 5. The moving front subsampling algorithm applied to a test node set with two boundaries. The initial node set is subsampled three times. The boundary node set is included in the domain node set such that they are subsampled collectively and simultaneously. Subsampling performance along the boundary is improved by including nodes interior and exterior to all boundaries. The method parameters for this example are $k = 10$ and $c = 1.5$.

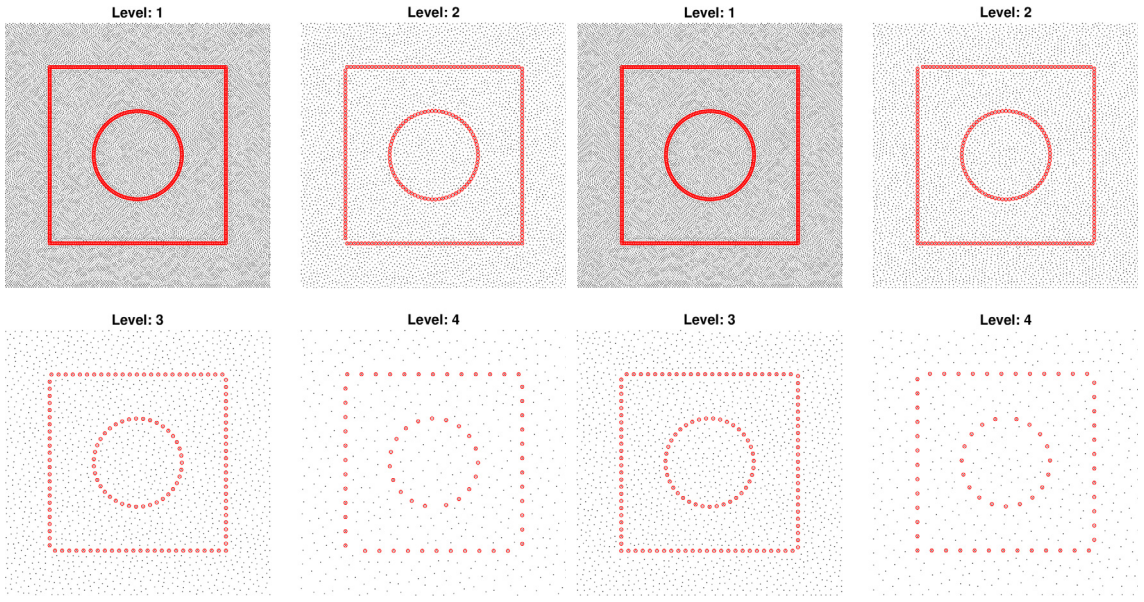
Another way to reduce the inconsistencies in subsampling which may be due to the inherent directional bias of the moving front algorithm is to alternate the direction between subsampling iterations. This technique of alternating direction is unsatisfactory, however, because an ideal method would be effective independent any inherent directional bias. To further improve robustness of the moving front method in the presence of boundaries, the following section considers subsampling boundaries separately.

4.2. Subsample boundary separately

First, the given boundary nodes are subsampled. Then, any domain nodes within a prescribed distance of the boundary nodes are removed. Finally, the domain nodes are subsampled. The effects of this two-step process can be seen in Fig. 6. Fig. 6(a) alternates direction of the moving front algorithm while Fig. 6(b) does not. A significant improvement in how consistently the algorithm behaves across the node set is apparent, independent of any direction bias in the subsampling algorithm.

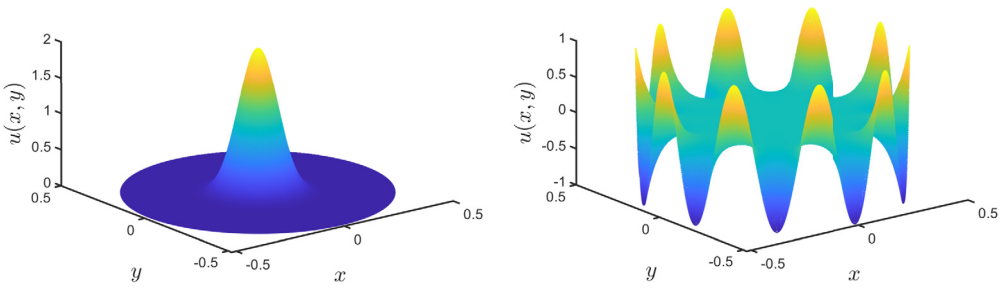
5. Meshfree multilevel RBF-FD solver

Traditionally, multigrid solvers have been used to accelerate the solution of large systems of equations [22,52]. Multigrid methods, however, require structured grids as the name indicates. In this section, it is illustrated how to utilize the geometric flexibility of RBF-FD in combination with the proposed node subsampling strategy to setup a GMM. Each example uses spatially varying node sets that seek to match the solutions. Consider the two-dimensional Poisson problem,



(a) Alternating direction (b) No alternating direction

Fig. 6. The moving front subsampling algorithm applied to a test node set with two boundaries. The initial node set is subsampled three times. In these figures, the boundary node set is subsampled separately from the domain node set. Subsampling performance along the boundary is improved by subsampling boundary nodes independently. Additionally, no directional bias is detectable even when the algorithm does not alternate direction. The method parameters for this example are $k = 10$ and $c = 1.5$.



(a) $u(r, \theta) = 2\exp(-r/0.01)$ (b) $u(r, \theta) = 1024\cos(10\theta)r^{10}$

Fig. 7. The analytical solutions used for the (a) Poisson and (b) Laplace test problems.

$$\begin{aligned} \nabla^2 u &= f, \quad \boldsymbol{x} \in \Omega \\ u &= g, \quad \boldsymbol{x} \in \partial\Omega \end{aligned} \tag{1}$$

where $u = u(\boldsymbol{x}) = u(x, y) \in \mathbb{R}$ is the exact solution in a disk with unit diameter, i.e., $\Omega = \{(x, y), x^2 + y^2 \leq 0.5\}$, $g = g(\boldsymbol{x}) \in \mathbb{R}$ specifies Dirichlet boundary conditions on $\partial\Omega$ and $f = f(\boldsymbol{x}) \in \mathbb{R}$ specifies the source term. Two different problems are solved using variable density node sets in order to test the applicability of the proposed node subsampling strategy. The first problem considered is the Poisson problem for which $g = 0$ and $f = 200e^{-100r}(100r - 1)/r$ such that the solution given in polar coordinates is $u(r, \theta) = 2\exp(-r/0.01)$, while the other is a Laplace problem (i.e. $f = 0$) for which $g = \cos(10\theta)$ such that the solution given is $u(r, \theta) = 1024\cos(10\theta)r^{10}$ (see Fig. 7).

5.1. Radial basis function-generated finite differences

To discretize the problem in (1) using RBF-FD [26,53,54] we approximate the exact solution to (1) as

$$u(\boldsymbol{x}) \approx u_h(\boldsymbol{x}) = \sum_{i=1}^{n} \kappa_i \phi\left(\|\boldsymbol{x} - \boldsymbol{x}_i\|_2\right) + \sum_{j=1}^{\ell} \gamma_j p_j(\boldsymbol{x}) \tag{2}$$

where $\phi_i(r) = \phi\left(\|\boldsymbol{x} - \boldsymbol{x}_i\|_2\right) = \|\boldsymbol{x} - \boldsymbol{x}_i\|_2^{2k+1}$ are polyharmonic spline (PHS) radial basis functions and $p_j(\boldsymbol{x})$ are bivariate monomials. The solution and approximation are required to match at n nodes, i.e., at spatial points $\{\boldsymbol{x}_i\}_{i=1}^n$,

$$u(\boldsymbol{x}_i) = u_h(\boldsymbol{x}_i), \quad \text{for} \quad i = 1, 2, \ldots, n \tag{3}$$

and the following constraints must hold:

$$\sum_{i=1}^{n} \kappa_i p_j(\boldsymbol{x}_i) = 0, \quad \text{for} \quad j = 1, 2, ..., \ell, \tag{4}$$

where $\ell = (m+1)(m+2)/2$ is the number of monomial terms in a bivariate polynomial of degree m. The above equations can be arranged in a linear system of equations,

$$\tilde{A} \begin{bmatrix} \boldsymbol{\kappa} \\ \boldsymbol{\gamma} \end{bmatrix} = \begin{bmatrix} A & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\kappa} \\ \boldsymbol{\gamma} \end{bmatrix} = \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{0} \end{bmatrix}, \tag{5}$$

where $\boldsymbol{\kappa}, \boldsymbol{u} \in \mathbb{R}^n$, $\boldsymbol{\gamma} \in \mathbb{R}^\ell$, $A_{ij} = \phi\left(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2\right)$ is an entry of the RBF collocation matrix $A \in \mathbb{R}^{n \times n}$ and $P_{ij} = p_j(\boldsymbol{x}_i)$ is an entry of the supplementary polynomial matrix $P \in \mathbb{R}^{n \times \ell}$. Now, the linear operation $\mathcal{L}$ can be approximated at an evaluation point $\boldsymbol{x}_e$ as,

$$\mathcal{L}u|_{\boldsymbol{x}_e} \approx \sum_{i=1}^{n} \kappa_i \mathcal{L}\phi\left(\|\boldsymbol{x} - \boldsymbol{x}_i\|_2\right)|_{\boldsymbol{x}_e} + \sum_{j=1}^{\ell} \gamma_j \mathcal{L}p_j(\boldsymbol{x})|_{\boldsymbol{x}_e}, \tag{6}$$

which again can be arranged in matrix-vector format,

$$\mathcal{L}u|_{\boldsymbol{x}_e} \approx \begin{bmatrix} \boldsymbol{a}^T & \boldsymbol{b}^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\kappa} \\ \boldsymbol{\gamma} \end{bmatrix} = \begin{bmatrix} \boldsymbol{a}^T & \boldsymbol{b}^T \end{bmatrix} \tilde{A}^{-1} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{0} \end{bmatrix} = \begin{bmatrix} \boldsymbol{w}^T & \boldsymbol{v}^T \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{0} \end{bmatrix} = \boldsymbol{w}^T \boldsymbol{u} \tag{7}$$

where $a_i = \mathcal{L}\phi\left(\|\boldsymbol{x} - \boldsymbol{x}_i\|_2\right)|_{\boldsymbol{x}_e}$ corresponds to the ith entry of $\boldsymbol{a} \in \mathbb{R}^n$ and $b_j = \mathcal{L}p_j(\boldsymbol{x})|_{\boldsymbol{x}_e}$ corresponds to the jth entry of $\boldsymbol{b} \in \mathbb{R}^\ell$. The weights necessary for the multilevel solver, i.e. $\boldsymbol{w} \in \mathbb{R}^n$, can equivalently be computed by solving the linear system,

$$\begin{bmatrix} A & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{v} \end{bmatrix} = \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix}. \tag{8}$$

The weights, which approximate the operator $\mathcal{L}$ at point x_e, can then be found for each point in the domain. These weight vectors $\boldsymbol{w}_i$ can then be assembled into an approximation of the operator $\mathcal{L}$ over the whole domain which we shall call L:

$$L = \begin{bmatrix} \boldsymbol{w}_1^T \\ \boldsymbol{w}_2^T \\ \vdots \\ \boldsymbol{w}_n^T \end{bmatrix}. \tag{9}$$

The linear system can then be represented as

$$L\boldsymbol{u} = \boldsymbol{f} \tag{10}$$

where $\boldsymbol{u}$ is a vector of function values and $\boldsymbol{f}$ is a vector of the right hand side of the PDE for all spatial points, including boundary points. In the following two examples, the weights are adjusted for all boundary nodes to reflect the Dirichlet conditions. Dirichlet boundary conditions are expressed by setting all of the weights on row i_b corresponding to boundary node x_{i_b} to zero except for that weight on the diagonal which is one. Then the system can be solved using a GMM method.

5.2. Geometric multilevel elliptic solver

The meshfree GMM introduced here is based on ideas similar to those of the meshfree GMM found in [16] which is used for solving PDEs on surfaces, although some parts differ. In this study, no Krylov subspace methods will be used to increase the rate of convergence [55]. Furthermore, the coarse grid difference operators are computed explicitly on each node set level. Finally, all restriction operations are performed as injection (i.e. directly using values from the fine node set). For further details on multilevel methods, the reader is referred to literature on the topics of multilevel approximation [36,37] and multilevel solvers [16,40].

A pseudocode for the proposed GMM is given in Algorithm 2, where u_1 is the solution at the finest node set level, $L = \{L_j\}_{j=1}^{p}$ are the difference operators computed for each level, $I = \{I_{j+1}^{j}\}_{j=1}^{p-1}$ are the interpolation operators for each level and $R = \{R_j^{j+1}\}_{j=1}^{p-1}$ are the restriction (injection) operators for each level. The multilevel solver performs up to i_{max} iterations (V-cycles) unless the relative residual $||r_1^i||_2/||r_1^0||_2 = ||f_1 - L_1 u_1^i||_2/||f_1 - L_1 u_1^0||_2$ of the ith iteration becomes less than a predefined tolerance tol.

The basis of the GMM is the geometric multilevel V-cycle, which is described in Algorithm 3. During the V-cycles, pre- and post smoothing operations are performed using (ν_1, ν_2) Gauss-Seidel relaxations, respectively. At the coarsest node set level, a sparse LU solver is used.

The pseudocode for performing the geometric multilevel preprocessing, i.e., establishing all the different subsets of nodes, $X = \{X_j\}_{j=1}^{p}$, and the discrete operators L, I, R, are given in Algorithm 4. The two input node sets for this algorithm are $\{X_{bg}$ and $X_b\}$ which describe the scattered node set covering Ω and boundary nodes on $\partial\Omega$ at the finest node set level, respectively. Finally, the parameter $N_{\min}$ is used to control the minimum number of boundary nodes at the coarsest node set level.

The multilevel solver is tested on node sets that have been generated with variable node densities as illustrated in Fig. 8, where $N_{\min} = 60$ for the Poisson problem and $N_{\min} = 120$ for the Laplace problem. The node density function used in this study is defined by a linear transition between two prescribed node densities as,

$$\rho(d) = \begin{cases} \rho_1, & d < d_{\text{lim}} \\ \rho_1 + (\rho_2 - \rho_1)(d - d_{\text{lim}})/d_{\text{bl}}, & d_{\text{lim}} \le d \le d_{\text{lim}} + d_{\text{bl}} \\ \rho_2, & \text{otherwise} \end{cases} \tag{11}$$

Algorithm 2 Geometric multilevel solver.

function MLSOLVER($u_1^0, f_1, L, I, R, \nu_1, \nu_2, tol_1, tol_2, i_{max}, i_{conv}$)
  $i \leftarrow 0$
  $r_1^0 \leftarrow ||f_1 - L_1 u_1^0||_2$
  $counter \leftarrow 0$
  while $i < i_{max}$ and $||r_1^i||_2 > ||r_1^0||_2 \cdot tol_1$ and $counter < i_{conv}$ **do**
    $i \leftarrow i + 1$
    $u_1^i \leftarrow$ MLVCYC($u_1^{i-1}, f_1, L, I, R, \nu_1, \nu_2$)
    $r_1^i \leftarrow ||f_1 - L_1 u_1^i||_2$
    if $|\ln r_1^i - \ln r_1^{i-1}| < tol_2$ **then**
      $counter \leftarrow counter + 1$
    else
      $counter \leftarrow 0$
    end if
  end while
  return u_1^i
end function

Algorithm 3 Geometric multilevel V-cycle.

function MLVCYC($u_1, f_1, L, I, R, \nu_1, \nu_2$)
  $u_1 \leftarrow$ RELAX(u_1, f_1, L_1, ν_1)
  $r_2 \leftarrow R_1^2(f_1 - L_1 u_1)$
  for $j = 2$ to $p - 1$ **do**
    $e_j \leftarrow$ RELAX($0, r_j, L_j, \nu_1$)
    $r_{j+1} \leftarrow R_j^{j+1}(r_j - L_j e_j)$
  end for
  $e_p =$ LUSOLVE(L_p, r_p)
  for $j = p - 1$ to 2 **do**
    $e_j \leftarrow e_j + I_{j+1}^j e_{j+1}$
    $e_j \leftarrow$ RELAX(e_j, r_j, L_j, ν_2)
  end for
  $u_1 \leftarrow u_1 + I_2^1 e_2$
  $u_1 \leftarrow$ RELAX(u_1, f_1, L_1, ν_2)
  return u_1
end function

Algorithm 4 Geometric multilevel preprocessing.

function MLPRE($X_{bg}, X_b, N_{\min}$)
  $[X, R, p] \leftarrow$ MLMFSUB($X_{bg}, X_b, N_{\min}$)
  $L_1 \leftarrow$ RBFFD(X_1, X_1)
  for $j = 1$ to $p - 1$ **do**
    $I_{j+1}^j \leftarrow$ RBFFD(X_{j+1}, X_j)
    $L_{j+1} \leftarrow$ RBFFD(X_{j+1}, X_{j+1})
  end for
  return X, L, I, R, p
end function

where $d = ||\boldsymbol{x}||_2$ is the distance to the origin according to Fig. 7, ρ_1 is the node density in region 1, d_{lim} is a distance within which ρ_1 is kept constant, whereas d_{bl} is the distance over which ρ_1 linearly blends into ρ_2. It should be noted that the nodes in the vicinity of boundary have been adjusted by means of repulsion only at the finest node set level [28]. It should be noted that the node density function, $\rho(d)$, is chosen such that node density can be matched with the characteristics of the solutions.

The numerical test setups have been chosen to showcase the applicability of the node subsampling strategy for multilevel solvers using RBF-FD and to test whether the high-order accuracy will still be dictated by the degree of the augmented polynomials as shown, e.g., in [53]. Thus, the parameters used for computing the difference operators, L, of polynomial degree m_L are chosen to be $(k, n) = (1, 2\ell)$, while the parameters $(m_I, k, n) = (0, 0, 5)$ are used for computing the interpolation operators, I. The parameters for the interpolation operators are kept fixed for all choices of m_L. Finally, the multilevel solver settings are defined as $(\nu_1, \nu_2, tol_1, tol_2, i_{max}, i_{conv}) = (2, 1, 10^{-16}, 10^{-1}, 50, 5)$ for both test problems, whereas the polynomial degree m_L ranges from 2 to 8. All reported wall clock times include only the execution of the multilevel schemes. They do not include those actions which are considered pre-processing, including the generation of the coarse levels and coarse level operators.[14]

5.3. Poisson equation test problem

First, it can be seen from Fig. 9 that the wall clock time[15] scales linearly with the number of nodes, which is in accordance with expectations for any multigrid or multilevel solver [16,22,52]. Furthermore, the maximum relative error ($||u - u_h||_\infty / ||u||_\infty$) decreases as function of node set

[14] The calculation of all RBF-FD operators is considered a preprocessing step which can be fully parallelized.

[15] For these tests, the algorithm achieved a relative residual within the prescribed tolerance ($tol = 10^{-16}$) before the maximum number of iterations was reached. The latter would produce a trivially linear plot of wall clock time versus number of nodes.

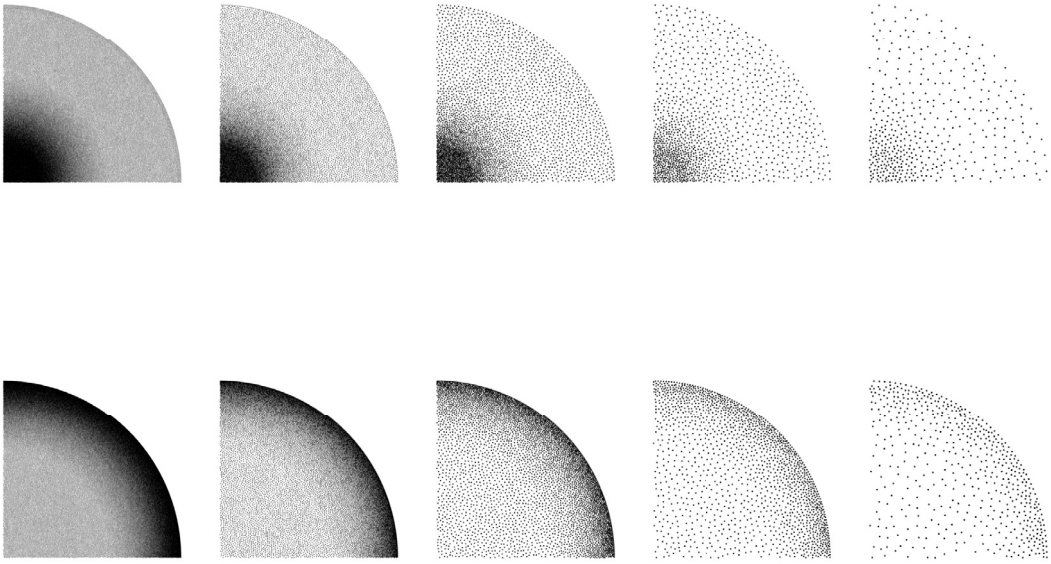


Fig. 8. Example of the multilevel node subsampling process for the Poisson problem node set (top) and the Laplace problem node set (bottom). Only nodes within the first quadrant of the Cartesian coordinate system are shown.

resolution ($\rho_{mean} = 1/\sqrt{N}$) and the slope is dictated by the polynomial degree of the difference operator, m_L. This is in agreement with previous RBF-FD studies [53].

In this study, the low-order interpolation operators are chosen in order to accelerate the convergence of the multilevel solver. However, this choice is not aligned with the rule of thumb for the transfer operators, i.e. $m_I + m_R > 2$, which is used in traditional multigrid methods for the Poisson problem [22]. In this work, the order of the restriction operators is $m_R = 0$ because injection is used as the restriction operation between all node set levels. Nevertheless, no detrimental effects have been noticed in any of the numerical tests conducted.

Finally, the proposed multilevel solver should provide solutions without any directional bias. Hence, to identify whether any directional bias is present, the relative errors for all node set resolutions have been normalized and depicted in Fig. 10. Thus, the scale factors used in Fig. 10 refer to the plateau of the maximum relative error plots in Fig. 9. As no particular directional pattern is noticed in Fig. 10, it can be concluded that the subsampling process used for setting up the multilevel solver does not introduce any directional bias.

5.4. Laplace equation test problem

The performance indicators of the multilevel solver for the Laplace problem are illustrated in Fig. 11. The same overall conclusions that were made for the Poisson problem can be made for the Laplace problem, i.e. high-order accuracy and linear scaling of the computation time.[16] For $m_L = 8$, note that the convergence factor ($||r_1^i||_2/||r_1^{i-1}||_2$) of the multilevel solver is less for $N = 14419$ and $N = 28279$ compared with the other values of N. The decrease in convergence factors is most likely caused by a stencil size that is too large as compared to the relatively low node density near the boundary at the most coarse level since localized error peaks are present for both $N = 14419$ and $N = 28279$ ($m_L = 8$) in Fig. 12. This decrease in convergence factor does not occur if the node resolution is increased to $N = 55966$ or above. Furthermore, if the multilevel V-cycle is used as a preconditioner for a Krylov subspace method, e.g. the generalized minimal residual method or biconjugate gradient stabilized method, fewer iterations will be needed for the solution to converge and the solver will be more robust compared to the standalone multilevel solver. However, each iteration will become more computational expensive. Thus, whether the multilevel V-cycle should be used as a standalone solver or a preconditioner is a trade-off between computational cost and robustness. With respect to the wall clock times for $N = 14419$ and $N = 28279$ given $m_L = 8$, the times are similar to those for $m_L = 6$ despite the lower rate of convergence. This similarity is due to fact that for $N = 14419$ and $N = 28279$ and $m_L = 6$ the relative residuals achieve the prescribed tolerance just before the maximum number of iterations is reached. Alternately, for $N = 14419$ and $N = 28279$ and $m_L = 8$, the maximum number of iterations is reached.[17]

The normalized relative error distributions in Fig. 12 illustrate that no directional bias seems to be introduced by the subsampling process, which is the same conclusion as for the Poisson problem.

6. Conclusion

A novel method for subsampling quasi-uniform node sets of highly variable density with sharp gradients is presented along with boundary preservation techniques and two novel measures for evaluating node quality of subsamplings. The moving front subsampling algorithm demonstrates the capability to coarsen a node set with high contrast and detail. Additionally, the moving front algorithm maintains the characteristics of the original node set as outlined by the comparative local regularity of the average distance and standard deviation of distances to the k nearest nodes. It is also faster, both by a constant and in the limit as node set size increases, than any other algorithm considered in this paper for subsampling variable density node sets.

[16] For these tests, the algorithm achieved a relative residual within the prescribed tolerance ($tol = 10^{-16}$) before the maximum number of iterations was reached except in the case of $m_L = 8$. For $m_L = 8$ and $N = 14419$ or $N = 28279$, the maximum number of iterations was reached before the relative residuals reached the prescribed tolerance.

[17] In these examples, the maximum number of iterations was 50.

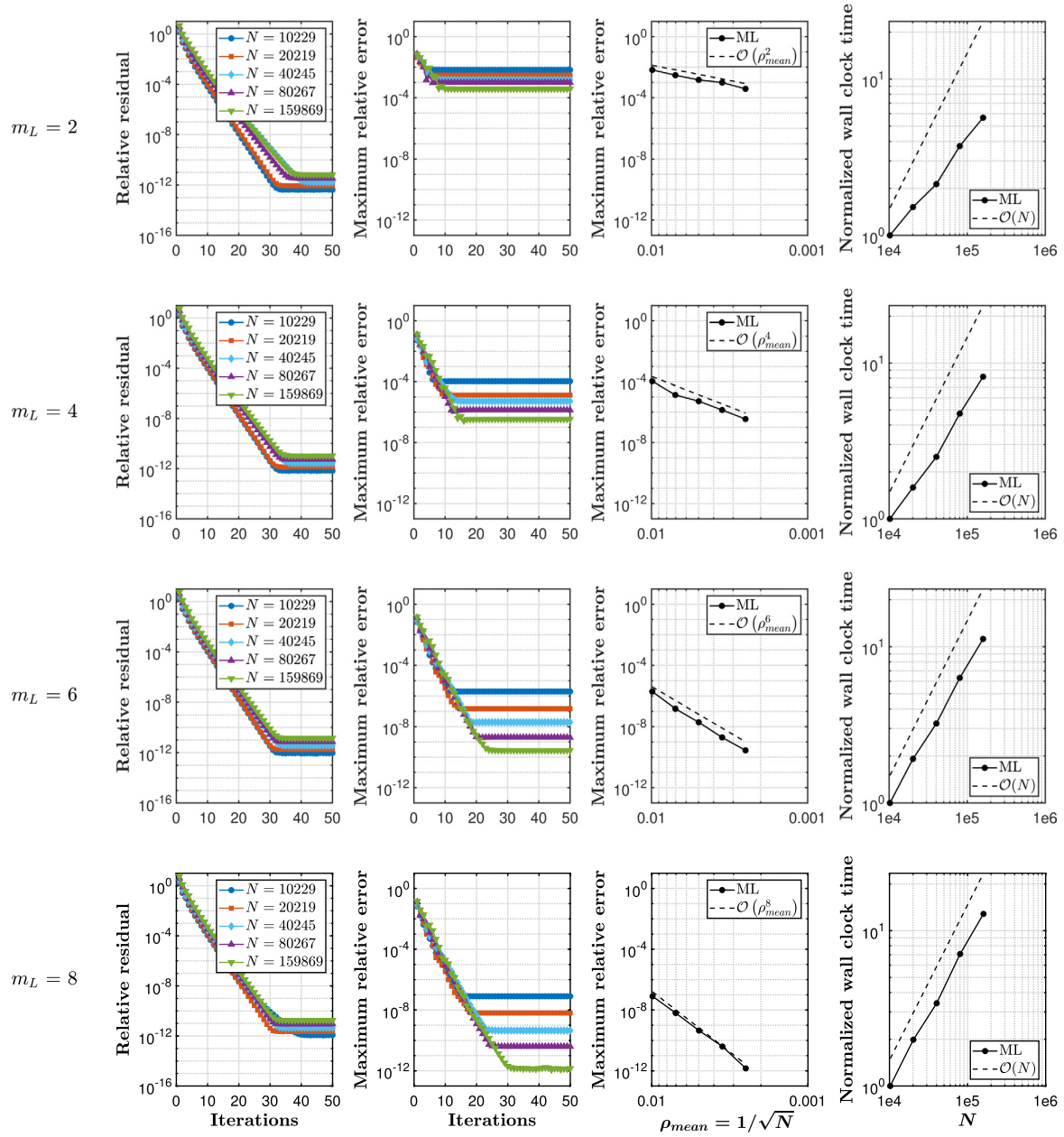


Fig. 9. Poisson problem performance indicators of the implemented GMM for polynomial degrees of $m_L = \{2, 4, 6, 8\}$ from top to bottom. The mean node density is defined as $\rho_{mean} = 1/\sqrt{N}$.

The utility of the moving front algorithm for the purpose of subsampling node sets in a meshfree multilevel PDE solver is demonstrated by solving both the Poisson and Laplace problems on variable density node sets. In both test cases, the meshfree PDE solver with the multilevel method and the proposed subsampling algorithm achieves the fast linear scaling of computational cost with node set size expected from a multilevel scheme. At the same time, this combination has no adverse impact on the expected high-order accuracy of the RBF-FD method when the node set is large enough relative to the variation in the node set. The meshfree multilevel PDE solver has been tested up through eighth order convergence and also demonstrates robust performance with minimal limitations based on node set size and density variation as seen in Section 5.4.

Data availability

No data was used for the research described in the article.

Link to the Reproducible Capsule

https://codeocean.com/capsule/3980307/tree

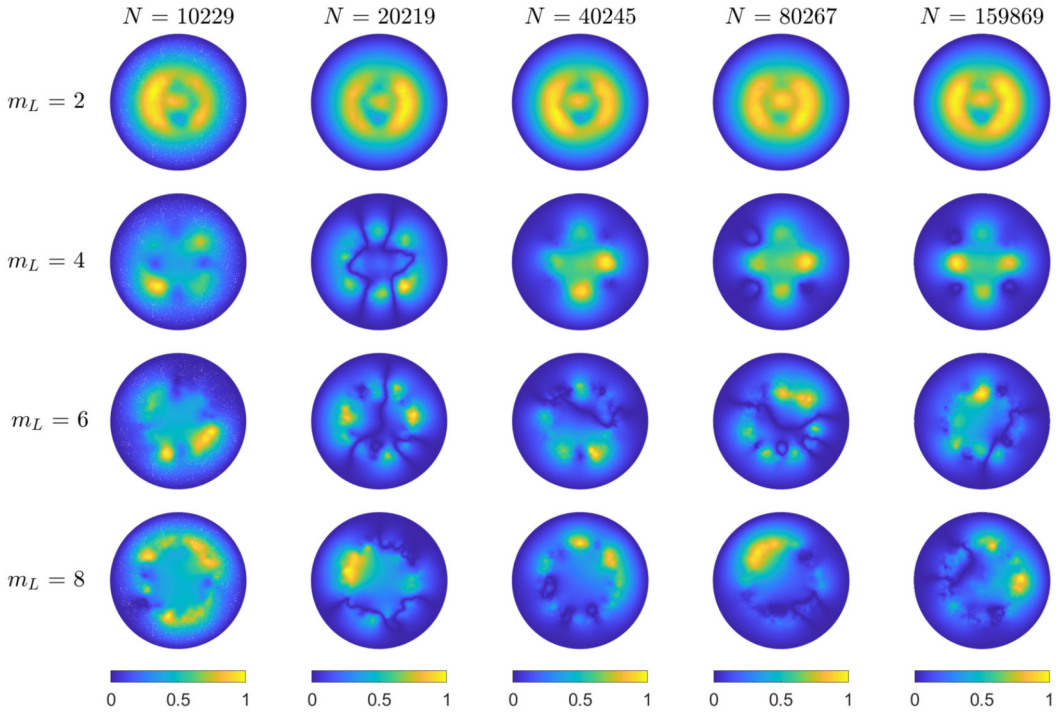


Fig. 10. Normalized relative error distributions for various orders of the difference operators and node set resolutions for the Poisson problem. The color scale factors refer to the plateau of the maximum relative error plots in Fig. 9.

Acknowledgements

Andrew Lawrence acknowledges support from the US Air Force.[18]

Appendix A. Subsampling algorithms

A.1. Moving front subsampling

The Python code for the moving front subsampling algorithm. The code can also be found on the author's GitHub repository in both MATLAB and Python along with examples of implementation [50].

```
import numpy as np
import copy
from sklearn.neighbors import NearestNeighbors

def MFNUS(xy, fc=1.5):
    """
    Moving Front Non-Uniform Subsampling

    Args:
        xy (array): initial node set to be subsample
        fc (float): coarsening factor algorithm

    Returns:
        xy_sub (array): subsampled node set
    """

    xy = np.array(xy)
    if xy.shape[0] < xy.shape[1]:
        xy = xy.T

    def hexRing(n):
```

[18] The views expressed in this article are those of the authors and do not reflect the official policy or position of the Air Force, the Department of Defense or the U.S. Government.

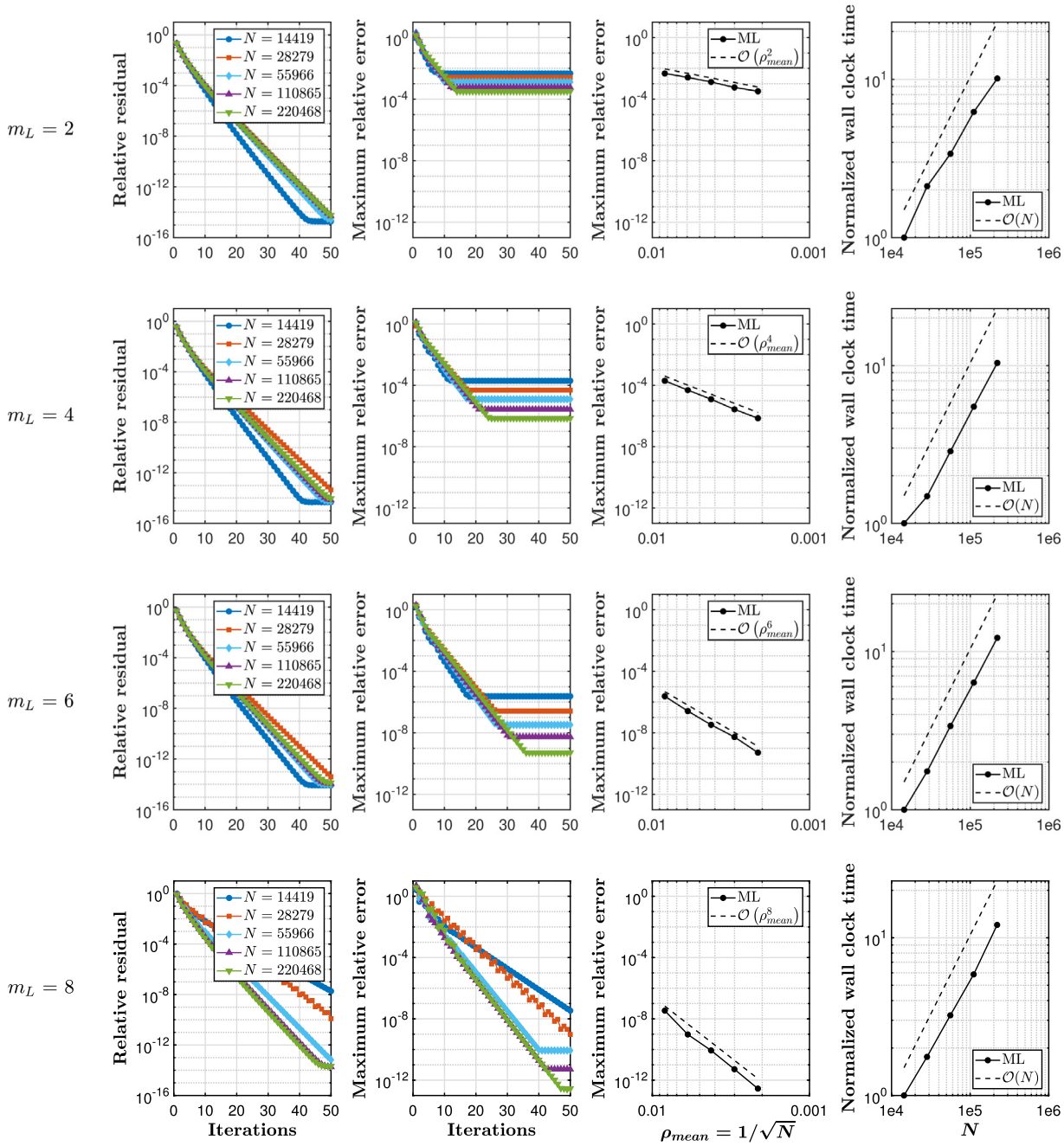


Fig. 11. Laplace problem performance indicators of the implemented GMM for polynomial degrees of $m_L = \{2, 4, 6, 8\}$ from top to bottom. The mean node density is defined as $\rho_{mean} = 1/\sqrt{N}$.

```
    return (n + 1) ** 3 - n ** 3 - 1

K = hexRing(int(np.ceil(fc)))

if xy.shape[0] < K:
    return []

N = xy.shape[0]  # Get the number of dots

# xy = xy[np.random.permutation(len(xy))]   # Randomize for robustness
xy = xy[np.lexsort(xy.T,axis=0),:]  # Sort dots from bottom and up

# Create nearest neighbor pointers and distances
tree = KDTree(xy)
```

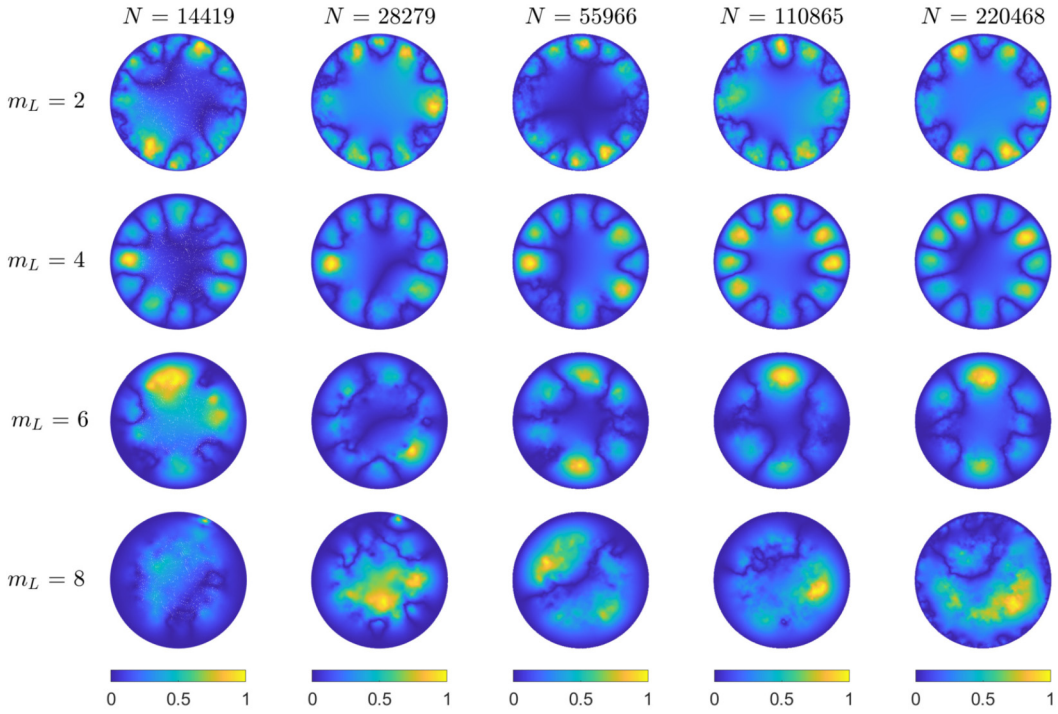


Fig. 12. Normalized relative error distributions for various orders of the difference operators and node set resolutions for the Laplace problem. The color scale factors refer to the plateau of the maximum relative error plots in Fig. 11.

```
    D, Idx = tree.query(xy, k=K)
    D1 = copy.deepcopy(D[:, 1])

    D[Idx<np.tile(Idx[:,0],(K,1)).T] = np.max(D)+1
    filt = D < fc * np.tile(D1,(K,1)).T

    for k in range(N):  # Loop over nodes
        if Idx[k, 0] != N+1:  # Check if node already eliminated
            ind = Idx[k, np.concatenate(([False], filt[k, 1:]))]
            Idx[ind, 0] = 0  # are within the factor fc of the closest one
                             # in the original node set

    xy_sub = xy[Idx[:, 0] != N+1]  # Eliminate these marked nodes

    return xy_sub

rep
```

References

[1] S. De Marchi, F. Piazzon, A. Sommariva, M. Vianello, Polynomial meshes: computation and approximation, in: Proceedings of the 15th International Conference on Computational and Mathematical Methods in Science and Engineering, 2015.

[2] A. Narayan, D. Xiu, Constructing nested nodal sets for multivariate polynomial interpolation, SIAM J. Sci. Comput. 35 (5) (2013) A2293–A2315, https://doi.org/10.1137/12089613X.

[3] F. Piazzon, A. Sommariva, M. Vianello, Caratheodory-Tchakaloff subsampling, Dolomites Res. Notes Approx. 10 (11 2016).

[4] A. Sommariva, M. Vianello, Computing approximate Fekete points by QR factorizations of Vandermonde matrices, Comput. Math. Appl. 57 (8) (2009) 1324–1336, https://doi.org/10.1016/j.camwa.2008.11.011.

[5] C. Roque, J. Madeira, A. Ferreira, Node adaptation for global collocation with radial basis functions using direct multisearch for multiobjective optimization, Eng. Anal. Bound. Elem. 39 (2014) 5–14, https://doi.org/10.1016/j.enganabound.2013.10.012.

[6] R.W. Kennard, L.A. Stone, Computer aided design of experiments, Technometrics 11 (1) (1969) 137–148, https://doi.org/10.1080/00401706.1969.10490666.

[7] X. Meng, Y. Li, D. Shi, S. Hu, F. Zhao, A method of power flow database generation base on weighted sample elimination algorithm, Front. Energy Res. 10 (2022).

[8] B. Shang, D.W. Apley, S. Mehrotra, Diversity subsampling: custom subsamples from large data sets, INFORMS J. Data Sci. 2 (2) (2023) 161–182, https://doi.org/10.1287/ijds.2022.00017.

[9] A.L. Silveira, P.J.S. Barbeira, A fast and low-cost approach for the discrimination of commercial aged cachaças using synchronous fluorescence spectroscopy and multivariate classification, J. Sci. Food Agric. 102 (11) (2022) 4918–4926, https://doi.org/10.1002/jsfa.11857.

[10] R.L. Cook, Stochastic sampling in computer graphics, ACM Trans. Graph. 5 (1) (1986) 51–72, https://doi.org/10.1145/7529.8927.

[11] M.A.Z. Dippé, E.H. Wold, Antialiasing through stochastic sampling, SIGGRAPH Comput. Graph. 19 (3) (1985) 69–78, https://doi.org/10.1145/325165.325182.

[12] C. Yuksel, Sample elimination for generating Poisson disk sample sets, Comput. Graph. Forum 34 (2015) 25–32, https://doi.org/10.1111/CGF.12538.

[13] S. Le Borne, M. Wende, Multilevel interpolation of scattered data using H-matrices, Numer. Algorithms 85 (4) (2020) 1175–1193, https://doi.org/10.1007/s11075-019-00860-1.

[14] F.P. Nick, Algebraic multigrid for meshfree methods, Ph.D. thesis, Universitäts-und Landesbibliothek, Bonn, 2020.

[15] B. Metsch, F. Nick, J. Kuhnert, Algebraic multigrid for the finite pointset method, Comput. Vis. Sci. 23 (2020) 1–14.
[16] G.B. Wright, A. Jones, V. Shankar, MGM: a meshfree geometric multilevel method for systems arising from elliptic equations on point cloud surfaces, SIAM J. Sci. Comput. 45 (2) (2023) A312–A337.
[17] K. Watanabe, H. Igarashi, T. Honma, Comparison of geometric and algebraic multigrid methods in edge-based finite-element analysis, IEEE Trans. Magn. 41 (5) (2005) 1672–1675.
[18] K.N. Volkov, V.N. Emel'yanov, I.V. Teterina, Geometric and algebraic multigrid techniques for fluid dynamics problems on unstructured grids, Comput. Math. Math. Phys. 56 (2016) 286–302.
[19] C.W. Oosterlee, T. Washio, An evaluation of parallel multigrid as a solver and a preconditioner for singularly perturbed problems, SIAM J. Sci. Comput. 19 (1) (1998) 87–110, https://doi.org/10.1137/S1064827596302825.
[20] N. Bell, L. Olson, J. Schroder, PyAMG: algebraic multigrid solvers in Python, https://github.com/pyamg/pyamg.
[21] Y. Shapira, Matrix-Based Multigrid: Theory and Applications, Springer, 2008.
[22] U. Trottenberg, C.W. Oosterlee, A. Schuller, Multigrid, Elsevier, 2000.
[23] R.D. Falgout, An introduction to algebraic multigrid, Comput. Sci. Eng. 8 (6) (2006) 24–33.
[24] B. Seibold, Performance of algebraic multigrid methods for non-symmetric matrices arising in particle methods, Numer. Linear Algebra Appl. 17 (2–3) (2010) 433–451.
[25] B. Fornberg, N. Flyer, Solving PDEs with radial basis functions, Acta Numer. 24 (2015) 215–258, https://doi.org/10.1017/S0962492914000130.
[26] B. Fornberg, N. Flyer, A Primer on Radial Basis Functions with Applications to the Geosciences, SIAM, Philadelphia, PA, 2015.
[27] T. Liu, R.B. Platte, Node generation for RBF-FD methods by QR factorization, Mathematics 9 (16) (2021), https://doi.org/10.3390/math9161845.
[28] B. Fornberg, N. Flyer, Fast generation of 2-D node distributions for mesh-free PDE discretizations, Comput. Math. Appl. 69 (7) (2015) 531–544, https://doi.org/10.1016/j.camwa.2015.01.009.
[29] P. Suchde, T. Jacquemin, O. Davydov, Point cloud generation for meshfree methods: an overview, Arch. Comput. Methods Eng. (2022) 1–27.
[30] K. van der Sande, B. Fornberg, Fast variable density 3-D node generation, SIAM J. Sci. Comput. 43 (1) (2021) A242–A257.
[31] M.S. Floater, A. Iske, Multistep scattered data interpolation using compactly supported radial basis functions, J. Comput. Appl. Math. 73 (1–2) (1996) 65–78.
[32] M.S. Floater, A. Iske, Thinning algorithms for scattered data interpolation, BIT Numer. Math. 38 (1998) 705–720.
[33] C. Drumm, S. Tiwari, J. Kuhnert, H.-J. Bart, Finite pointset method for simulation of the liquid–liquid flow field in an extractor, Comput. Chem. Eng. 32 (12) (2008) 2946–2957.
[34] P. Suchde, J. Kuhnert, A fully Lagrangian meshfree framework for PDEs on evolving surfaces, J. Comput. Phys. 395 (2019) 38–59.
[35] P. Suchde, A meshfree Lagrangian method for flow on manifolds, Int. J. Numer. Methods Fluids 93 (6) (2021) 1871–1894.
[36] G.E. Fasshauer, Meshfree Approximation Methods with MATLAB, vol. 6, World Scientific, 2007.
[37] H. Wendland, Scattered Data Approximation, vol. 17, Cambridge University Press, 2004.
[38] H. Wendland, Multiscale analysis in Sobolev spaces on bounded domains, Numer. Math. 116 (2010) 493–517.
[39] H. Wendland, Multiscale radial basis functions, in: Frames and Other Bases in Abstract and Function Spaces, Springer, 2017.
[40] R. Zamolo, E. Nobile, B. Šarler, Novel multilevel techniques for convergence acceleration in the solution of systems of equations arising from RBF-FD meshless discretizations, J. Comput. Phys. 392 (2019) 311–334.
[41] A. Katz, A. Jameson, Multicloud: multigrid convergence with a meshless operator, J. Comput. Phys. 228 (14) (2009) 5237–5250, https://doi.org/10.1016/j.jcp.2009.04.023.
[42] S.T. Ha, H.G. Choi, A meshless geometric multigrid method based on a node-coarsening algorithm for the linear finite element discretization, Comput. Math. Appl. 96 (2021) 31–43.
[43] R. Cavoretto, A. De Rossi, Adaptive meshless refinement schemes for RBF-PUM collocation, Appl. Math. Lett. 90 (2019) 131–138.
[44] R. Cavoretto, A. De Rossi, A two-stage adaptive scheme based on RBF collocation for solving elliptic PDEs, Comput. Math. Appl. 79 (11) (2020) 3206–3222.
[45] Q. Zhang, Y. Zhao, J. Levesley, Adaptive radial basis function interpolation using an error indicator, Numer. Algorithms 76 (2017) 441–471.
[46] L. Ling, An adaptive-hybrid meshfree approximation method, Int. J. Numer. Methods Eng. 89 (5) (2012) 637–657.
[47] R. Cavoretto, A. De Rossi, An adaptive residual sub-sampling algorithm for kernel interpolation based on maximum likelihood estimations, J. Comput. Appl. Math. 418 (2023) 114658.
[48] J. Behrens, A. Iske, S. Pöhn, Effective node adaption for grid-free semi-Lagrangian advection, in: Discrete Modelling and Discrete Algorithms in Continuum Mechanics, 2001, pp. 110–119.
[49] S. Kaennakham, N. Chuathong, An automatic node-adaptive scheme applied with a RBF-collocation meshless method, Appl. Math. Comput. 348 (2019) 102–125.
[50] A.P. Lawrence, Subsampling algorithms, https://github.com/andplaw/Subsampling.
[51] R. Bridson, Fast Poisson disk sampling in arbitrary dimensions, SIGGRAPH sketches 10 (1) (2007) 1.
[52] W.L. Briggs, V.E. Henson, S.F. McCormick, A Multigrid Tutorial, 2nd edition, SIAM, 2000.
[53] V. Bayona, N. Flyer, B. Fornberg, G.A. Barnett, On the role of polynomials in RBF-FD approximations: II. Numerical solution of elliptic PDEs, J. Comput. Phys. 332 (2017) 257–273.
[54] 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, https://doi.org/10.1016/j.jcp.2016.02.078.
[55] A. Ghai, C. Lu, X. Jiao, A comparison of preconditioned Krylov subspace methods for large-scale nonsymmetric linear systems, Numer. Linear Algebra Appl. 26 (1) (2019) e2215, https://doi.org/10.1002/nla.2215.