Proceeding Series of the Brazilian Society of Computational and Applied Mathematics, v. 12, n. 1, 2026.

Trabalho apresentado no XLIV CNMAC, Fundação Getulio Vargas - Rio de Janeiro - RJ, 2025.

Proceeding Series of the Brazilian Society of Computational and Applied Mathematics

Moving Least Square Interpolation in Matrix-Based Finite Difference Schemes on Adaptive Cartesian Grids

Catalina M. Rúa-Alvarez[1]
Universidad de Nariño, Pasto, Colombia
Claudia P. Ordóñez[2]
DMS/UPRM, Mayagüez, PR
Priscila C. Calegari[3]
INE/UFSC, Florianópolis, SC

Abstract. A variety of applications related to engineering and physics can be mathematically modeled by Partial Differential Equations, such as Poisson's equation with Dirichlet or Neumann boundary conditions. This study focuses on the application of Moving Least Squares (MLS) interpolation within matrix-based Finite Difference schemes to approximate the two-dimensional Poisson's equation on adaptive Cartesian grids or Adaptive Mesh Refinement in specific regions using Matlab. Communication between cells at different refinement levels is typically done using Lagrange interpolation methods; however, this paper investigates and discusses the benefits of MLS interpolation on the proposed approach.

Keywords. Adaptive Mesh Refinement, Cartesian grid, Finite Difference Method, Moving Least Square.

1 Introduction

Partial Differential Equations, such as Poisson's equation, model numerous physical phenomena. However, analytical solutions to these problems are often unattainable and require the use of numerical methods such as Finite Differences (FD), Finite Volumes, or Finite Elements [4, 10]. Although FD is widely used, due to its practicality, stability conditions can make the uniform domain discretization infeasible and may be inefficient for problems with localized solution constraints. Consequently, using an uniform domain discretization to approximate these solutions may not be ideal, as it could result in low accuracy or be computationally expensive. Therefore, the importance of considering Adaptive Mesh Refinement (AMR). There are various techniques for generating static or dynamic AMR, such as composite meshes formed by a set of rectangular blocks with nesting constraints as described in [1]. Also in [2], are presented meshes that subdivide rectangular cells into four finer-level cells by bisecting their horizontal and vertical dimensions, forming the Adaptive Cartesian Grids (ACG) which are used in this research. Sometimes these meshes are also called Quadtree [10]. Other approaches include hierarchical grid that also allow Cartesian discretizations, as detailed in [8]. Furthermore, [7] includes the application of FD discretization within AMR in matrix format, allowing the utilization of numerical linear algebra libraries. Recently, theoretical results for the approximation of elliptic equations with matrix-based FD schemes have been studied in [3].

[1]catalina.rua@udenar.edu.co
[2]ordoez.patricia@hotmail.com
[3]priscila.calegari@ufsc.br

Effective ACG implementation requires robust level cell communication, for which interpolation methods, including Lagrange [1, 7] and Moving Least Squares (MLS) [8] interpolations, are crucial. MLS interpolation is also used in the formulation of meshfree schemes [9]. We have illustrated the capabilities of our approach using this method by simulating numerically the approximation for Poisson's equation in a robust variety of complex meshes. Furthermore, efficient data structures, such as Hash Tables, are essential for managing AMR meshes [2, 6]. This research explores FD discretization on ACG using matrix formulations, comparing Lagrange and MLS interpolations for cell communication in a block-structured, it is a necessary requirement to Lagrange interpolation. Simulations are also performed with MLS on general ACG without losing the second order of convergence. Hash Table data structure are used for mesh management. Finally, without sacrificing accuracy, we have explored different ways to select the cloud of points to MLS, based on points and cells and we have performed a numerical simulation to explore the effectivity of the selection. We have included a numerical error analysis with the order of approximation of the method to evaluate the efficiency of MLS. Moreover, we have included a sparse matrix analysis based on the condition number. All implementations are conducted using Matlab.

2 Domain Discretization

We are interested to approximate a Poisson equation of the form

$$-\nabla^2\boldsymbol{\varphi}(x,y) = -\left(\frac{\partial^2\boldsymbol{\varphi}(x,y)}{\partial x^2} + \frac{\partial^2\boldsymbol{\varphi}(x,y)}{\partial y^2}\right) = \boldsymbol{f}(x,y), \tag{1}$$

with Dirichlet or Neumann boundary conditions and $(x,y) \in \Omega = [a_1,a_2]\times[b_1,b_2] \subset \mathbb{R}^2$. The uniform discretization divides the domain at n and m partitions on the x and y axes, where $\Delta x = (a_2-a_1)/n$ and $\Delta y = (b_2-b_1)/m$. A point (x_i,y_j) at the center of the cell has coordinates $x_i = x_0 + (i+\frac{1}{2})\Delta x$ and $y_j = y_0 + (j+\frac{1}{2})\Delta y$ with $x_0 = a_1$ and $y_0 = b_1$. Thus, the second order five points FD method to approximate (1) is given by

$$-\frac{\boldsymbol{\varphi}_{i,j-1}}{\Delta y^2} - \frac{\boldsymbol{\varphi}_{i-1,j}}{\Delta x^2} + 2\left(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}\right)\boldsymbol{\varphi}_{i,j} - \frac{\boldsymbol{\varphi}_{i+1,j}}{\Delta x^2} - \frac{\boldsymbol{\varphi}_{i,j+1}}{\Delta y^2} = \boldsymbol{f}_{i,j}. \tag{2}$$

The stencil varies according to cells positions. If the cell (i,j) is in the boundary it is necessary to approximate it, maintaining the second order approximation. For instance, a cell with index $(i,0)$ (in the inferior boundary) in the Dirichlet case, is given by, $\boldsymbol{\varphi}_{i,-1} = 2\boldsymbol{\alpha}_i - \boldsymbol{\varphi}_{i,0} + \mathcal{O}\left(\Delta y^2\right)$, where $\boldsymbol{\varphi}_{i,-1} = \boldsymbol{\varphi}\left(x_i, y_0 - \frac{\Delta y}{2}\right)$ and $\boldsymbol{\alpha}_i = \boldsymbol{\varphi}\left(x_i,y_0\right)$. The stencil using (2), including a cell $(i,-1)$ is

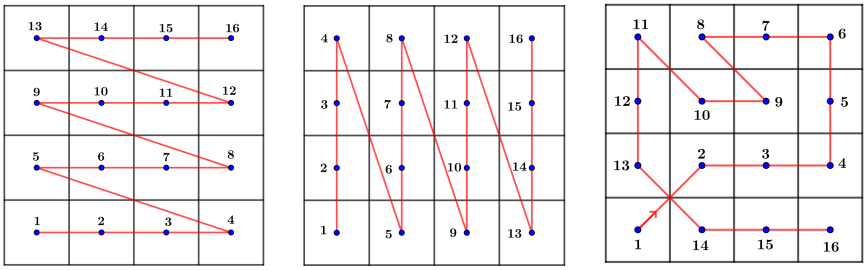
$$-\frac{\boldsymbol{\varphi}_{i-1,0}}{\Delta x^2} + \left(\frac{2}{\Delta x^2}+\frac{3}{\Delta y^2}\right)\boldsymbol{\varphi}_{i,0} - \frac{\boldsymbol{\varphi}_{i+1,0}}{\Delta x^2} - \frac{\boldsymbol{\varphi}_{i,1}}{\Delta y^2} = \boldsymbol{f}_{i,0} + \frac{2}{\Delta y^2}\boldsymbol{\alpha}_i.$$

Analogously, to Neumann boundary condition, with $\boldsymbol{\beta}_i = \boldsymbol{\varphi}_y\left(x_i + \frac{\Delta x}{2}, y_0\right)$ we have used $\boldsymbol{\varphi}_{i,-1} = \boldsymbol{\varphi}_{i,0} - \Delta y\boldsymbol{\beta}_i + \mathcal{O}\left(\Delta y^3\right)$. In this case, the stencil with the five points FD (2) change to

$$-\frac{\boldsymbol{\varphi}_{i-1,0}}{\Delta x} + \left(\frac{2}{\Delta x^2}+\frac{1}{\Delta y^2}\right)\boldsymbol{\varphi}_{i,0} - \frac{\boldsymbol{\varphi}_{i+1,0}}{\Delta x^2} - \frac{\boldsymbol{\varphi}_{i,1}}{\Delta y^2} = \boldsymbol{f}_{i,0} - \frac{1}{\Delta y}\boldsymbol{\beta}_i.$$

Then, by calculating the stencil for each mesh point and considering the boundary conditions, we get a system of equations $A\boldsymbol{\varphi} = \boldsymbol{F}$, where $\boldsymbol{F}$ is the right-hand side of known values. The sequence in which mesh points are approximated varies with the enumeration or order of routing the cells, which also changes the resulting sparsity pattern of the matrix within the FD. Figure 1 illustrates some enumeration patterns for the uniform mesh case, more details in [5].

DOI: 10.5540/03.2026.012.01.0316



(a) Lexicographical order. (b) Down-Up order. (c) Aleatory order.

Figure 1: Space filling curves. Source: own creation.

To illustrate the matrix creation with the lexicographical order and the Dirichlet boundary conditions, we define the variables $a_1 = -3\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right)$, $a_2 = -\left(\frac{2}{\Delta x^2} + \frac{3}{\Delta y^2}\right)$, $a_3 = -\left(\frac{3}{\Delta x^2} + \frac{2}{\Delta y^2}\right)$, $a_4 = -2\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right)$, $b = \frac{1}{\Delta x^2}$ and $c = \frac{1}{\Delta y^2}$. Then, are created the $n \times n$ block matrices T_i and the $m \times m$ auxiliary matrices D_i, given by

$$T_1 = \begin{pmatrix} a_1 & b & \cdots & \cdots & 0 \\ b & a_2 & \ddots & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \cdots & \ddots & a_2 & b \\ 0 & \cdots & \cdots & b & a_1 \end{pmatrix}, \quad T_2 = \begin{pmatrix} a_3 & b & \cdots & \cdots & 0 \\ b & a_4 & \ddots & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \cdots & \ddots & a_4 & b \\ 0 & \cdots & \cdots & b & a_3 \end{pmatrix}, \quad T_3 = \begin{pmatrix} c & 0 & \cdots & \cdots & 0 \\ 0 & c & \ddots & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \cdots & \ddots & c & 0 \\ 0 & \cdots & \cdots & 0 & c \end{pmatrix},$$

$$D_1 = \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ 0 & 0 & \ddots & \cdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \cdots & \ddots & 0 & 0 \\ 0 & \cdots & \cdots & 0 & 1 \end{pmatrix}, \quad D_2 = \begin{pmatrix} 0 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & \ddots & \cdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \cdots & \ddots & 1 & 0 \\ 0 & \cdots & \cdots & 0 & 0 \end{pmatrix}, \quad \text{and} \quad D_3 = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 1 & 0 & \ddots & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \cdots & \ddots & 0 & 1 \\ 0 & \cdots & 0 & 1 & 0 \end{pmatrix}.$$

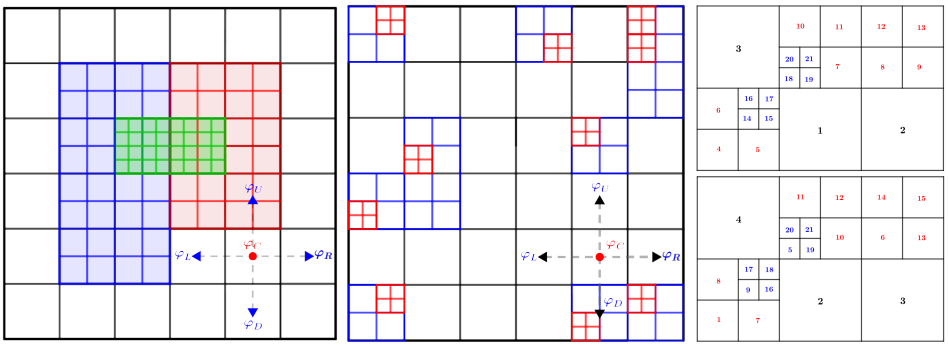
Finally, using the Kronecker product we obtain the matrix of coefficients to be solved

$$A = -\left(D_1 \otimes T_1 + D_2 \otimes T_2 + D_3 \otimes T_3\right).$$

It is important to highlight that the sparsity structure of the matrix is defined by the enumeration.

Now, we extend the methodology of the AMR approach to Cartesian domains. The refinement process starts from a uniform mesh, where each cell, selected by some refinement criteria, is divided and replaced by new cells. The process continues until the intended refinement level is reached or until the maximum level. The adaptive mesh generated has groups of cells at different levels of refinement, Figure 2. In [1], the grids are block-structured and follow certain rules to be properly nested, as shown in Figure 2(a). By the other hand, [2, 10] present ACG dividing a cell into four new fine cells, as illustrated in Figure 2(b). Additionally, to apply FD scheme (2) in these meshes, it is necessary to define an interpolation method to the communication between cells at different refinement levels. For example, in Figure 2(a), to obtain φ_U it is possible to apply the second order Lagrange interpolation [7]. In contrast, approximating φ_D in Figure 2(b), requires special techniques, such as, MLS interpolation, see [8]. Notably, MLS can also be utilized for the approximation of φ_U in Figure 2(a). This versatility enhances the utility of MLS in ACG.

In both interpolation methods, the five-point stencil alters the sparsity of the associated matrix. Figure 2(c) presents the enumeration cells in lexicographical order by level (up) and by the order of mesh creation (down) with the Hash Table data structure.

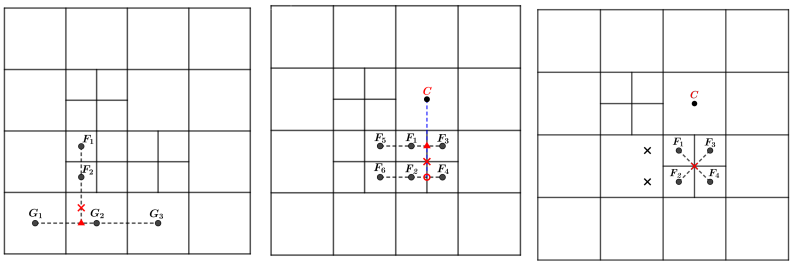


(a) Property nested AMR. (b) ACG. (c) ACG enumeration.

Figure 2: Composite meshes with three levels of refinement. Source: own creation.

3 Interpolation methods

The second order Lagrange interpolation for cell communication between two refinement levels, known as fine-coarse interpolation or covered-cell interpolation, are illustrated in Figure 3. The point, shown in red ($\times$), in both cases must be approximated. In Figure 3(a), a second order Lagrange interpolation with the coarse points G_i is first employed to obtain a value in the red triangle. Then, another Lagrange interpolation including the fine points F_i is used to obtain $\times = -\frac{1}{5}F_1+\frac{2}{3}F_2+\frac{1}{12}G_1+\frac{1}{2}G_2-\frac{1}{20}G_3$. On the other hand, in Figure 3(b), the red ($\times$) value is calculated using two second-order Lagrange interpolations, with fine values F_i, $\blacktriangle = -\frac{1}{8}F_5 + \frac{3}{4}F_1 + \frac{3}{8}F_3$ and $\circ = -\frac{1}{8}F_6 + \frac{3}{4}F_2 + \frac{3}{8}F_4$. Using Lagrange interpolation again, with the approximations at the red points ($\blacktriangle$ and $\circ$) together with the coarse point C, the approximate value at point ($\times$) is given by, $\times = \frac{1}{2}F_1 + \frac{3}{10}F_2 + \frac{1}{4}F_3 + \frac{3}{20}F_4 - \frac{1}{12}F_5 - \frac{1}{20}F_6 - \frac{1}{15}C$. The high order in the last interpolation is important for maintaining the second order approximation in the FD scheme (2), especially in T-junction cells. In the ACG, in Figure 3(c), Lagrange interpolation is limited to the first order.



(a) Fine-coarse. (b) T-junction. (c) ACG - first order.

Figure 3: Second order Lagrange interpolations. Source: own creation.

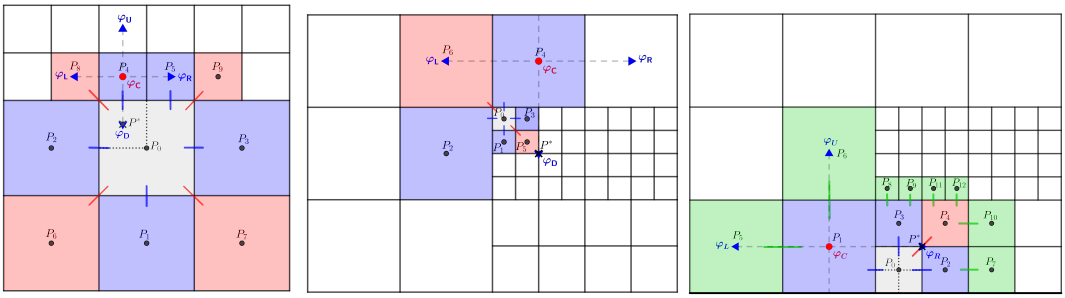
Following [8], in the MLS method, we have used the polynomial base $b = \left[1, x, y, xy, x^2, y^2\right]$ to obtain the coefficients w_j such as $\boldsymbol{\varphi}(\boldsymbol{x}) \approx \sum_{j=1}^{6} w_j b_j(\boldsymbol{x})$. Using Algorithm 1, we need to choose, at

least six neighbors points, to achieve second order of approximation. For ACG, we have selected points using the neighborhood from the edges and nodes of the cell that contains the point to be approximated.

Algorithm 1 Computation of weights w_i using MLS interpolation from [8].

1: Given a list of k points $\boldsymbol{x}_i$, a point $\boldsymbol{x} \in \mathbb{R}^2$ and a basis of l elements $b_1, b_2 \ldots, b_l$, with $k \geq l$.
2: Calculate $W = 1/\sqrt{\|\boldsymbol{x} - \boldsymbol{x}_i\|}$ and $P = [b_j(\boldsymbol{x}_i)]$, with $1 \leq i \leq k$, $1 \leq j \leq l$.
3: Do the QR decomposition to the matrix WP, such as $QR_1 = WP$.
4: Define $Q = [Q_1 \;\; Q_2]$, with $Q_1 \in \mathbb{R}^{k \times l}$ and $Q_2 \in \mathbb{R}^{k \times k-l}$.
5: Define $R_1 = [R \;\; O]^t$, such as $R \in \mathbb{R}^{l \times l}$ and $O \in \mathbb{R}^{k-l \times l}$.
6: Compute $\boldsymbol{b} = (b_1(\boldsymbol{x}), \ldots, b_l(\boldsymbol{x}))^t$.
7: Solve the triangular system $\boldsymbol{d} = R^{-t}\boldsymbol{b}$.
8: Calculate $\boldsymbol{w} = WQ_1\boldsymbol{d}$, since $\boldsymbol{w} = WQ_1R^{-t}\boldsymbol{b}$.

In Figure 4(a), to obtain the FD stencil for $\boldsymbol{\varphi}_C$ it is only necessary to find the fine-coarse interpolation using MLS for $\boldsymbol{\varphi}_D$. The other points $\boldsymbol{\varphi}_L$, $\boldsymbol{\varphi}_R$ and $\boldsymbol{\varphi}_U$ are at the same fine level as $\boldsymbol{\varphi}_C$. Since $\boldsymbol{\varphi}_D$ belongs to the gray cell with center point P_0, the known points next to this cell depend on its edges (blue lines) and nodes (red lines). These points are chosen in counterclockwise orientation for edges and in clockwise orientation for nodes. For example, in Figure 4(a) we obtained the following: left edge (P_1), down edge (P_2), right edge (P_3), above edge in a fine level share with cells (P_4 and P_5), left above node (P_6), right above node (P_7), right down node (P_8) and left down node (P_9). Finally, in this case, the list of ten points given to the MLS Algorithm 1 is $[P_0, P_1, P_2, \cdots, P_9]$.



(a) Fine-coarse. (b) Covered-cell case 1. (c) Covered-cell case 2.

Figure 4: Points selection to MLS in ACG with three levels of refinement. Source: own creation.

We consider two cases in which we select points for covered cells. Case 1 is shown in Figure 4(b), where $\boldsymbol{\varphi}_C$ is the cell approximated by the FD stencil and $\boldsymbol{\varphi}_D$ is the point to be approximated by MLS. $\boldsymbol{\varphi}_D$ is a node to four fine cells, so the next cell choose is the left above (P_0) and to this cell is done the order for edges and nodes, as described before for the Fine-coarse interpolation. So the list of points is given by: left edge in a coarse level (P_1), down edge in a fine level (P_2), right edge in a fine level (P_3), above edge in a coarse level (P_4), right down node (P_5), left down node (P_2) (already chosen), left above node (P_6) and the right above node (P_4) (already chosen). Finally, the list of seven points for Algorithm 1 is $[P_0, P_1, P_2, \cdots, P_6]$.

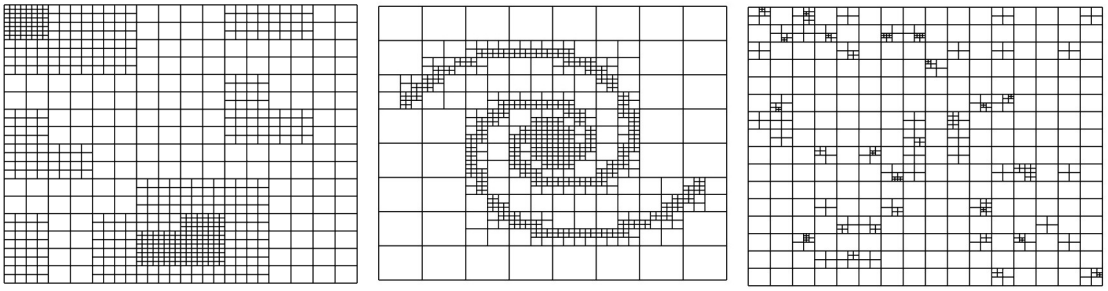
Case 2 is included in Figure 4(c), where $\boldsymbol{\varphi}_R$ is the point to be approximated for the FD stencil of $\boldsymbol{\varphi}_C$. $\boldsymbol{\varphi}_R$ is a node in the corner of four fine cells. Using case 1, the cell selected nearest is the point P_0 (in gray). After this, the selected points are: left edge in a coarse level (P_1), the down edge point needs to be approximated with the boundary condition, right edge in a fine level (P_2),

DOI: 10.5540/03.2026.012.01.0316

above edge in a fine level (P_3) and the right above node (P_4) are the only possible options to be selected. The problem in this case is that the list has only five points, and the method requires at least six. Therefore, all neighboring points from the edges in green are additionally selected to complete the list. The resulting list with thirteen points for Algorithm 1 is $[P_0, P_1, P_2, \cdots, P_{12}]$.

4 Results and Discussion

Numerical results are presented using different meshes, Figure 5, to approximate the Poisson's equation (1) with Dirichlet and Neumann boundary conditions, $\Omega = [0,1] \times [0,1]$, $\boldsymbol{f}(x,y) = -8\pi^2 \sin(2\pi x)\sin(2\pi y)$ and analytic solution $\boldsymbol{\varphi}(x,y) = \sin(2\pi x)\sin(2\pi y)$. MLS can be used in ACG meshes, Figure 5(a)-(c). Lagrange interpolation can only be used in properly nested meshes, Figure 5(a). The results obtained in tables 1 and 2 present the second-order convergence accuracy. Thus, MLS and Lagrange interpolation are working properly. The matrix information in Table 2 shows that the condition number increases with the dimension of the matrix, but the ratio between maximum and minimum singular values approaches one.



(a) Property nested AMR. (b) ACG. (c) Aleatory ACG.

Figure 5: Test meshes. Source: own creation.

Table 1: Error and convergence order for mesh in Figure 5(a) with uniform mesh base $n \times n$.

Dirichlet					**Neumann**				
	Lagrange		**MLS**			**Lagrange**		**MLS**	
n	Error	Ratio	Error	Ratio	n	Error	Ratio	Error	Ratio
16	4.5393e-3	-	3.9021e-3	-	16	7.1250e-3	-	5.7704e-03	-
32	1.1068e-3	4.101	1.1512e-3	3.390	32	1.7036e-3	4.182	2.3291e-3	2.478
64	2.7432e-4	4.035	3.1484e-4	3.656	64	4.1617e-4	4.094	7.3404e-4	3.173
128	6.8341e-5	4.014	8.2383e-5	3.822	128	1.0281e-4	4.048	2.0420e-4	3.595
256	1.7059e-5	4.006	2.1073e-5	3.909	256	2.5548e-5	4.024	5.3716e-5	3.802
512	4.2617e-6	4.003	5.3291e-6	3.954	512	6.3675e-6	4.012	1.3766e-5	3.902

Furthermore, the number of points used in Algorithm 1 for MLS interpolations was analyzed. For $n = 16$, in Figure 5(b), the minimum and maximum number of points were 6 and 22, respectively. The number of non-zero entries in the matrix was $nnz = 22237$, representing only 0.83% of the dimension of the matrix. This justifies the use of sparse matrices. Similarly, with $n = 32$ (Figure 5(c)), the point count ranged from a minimum of 6 to a maximum of 16, with $nnz = 20692$.

We present an adaptive method to solve Poisson's equation using a matrix formulation. We have compared the MLS and Lagrange interpolation to approximate data. The numerical verification performed indicates that the numerical method has second-order convergence accuracy for problems with smooth solutions. The results corroborate that MLS constitutes a versatile and valuable

DOI: 10.5540/03.2026.012.01.0316

Table 2: Error and convergence order, MLS interpolation and matrix information.

Mesh in Figure 5(b)					**Mesh in Figure 5(c)**				
n	Error	Ratio	Condest	$\frac{\lambda_{max}}{\lambda_{min}}$	n	Error	Ratio	Condest	$\frac{\lambda_{max}}{\lambda_{min}}$
16	4.0869e-3	-	1.4879e+5	1.027	32	1.1890e-3	-	1.0719e+5	1.015
32	1.0967e-3	3.727	6.3640e+5	1.007	64	3.2271e-4	3.684	4.3461e+5	1.002
64	2.8359e-4	3.867	2.6387e+6	1.002	128	8.4997e-5	3.797	1.7449e+6	1.000
128	7.2102e-5	3.933	1.0719e+7	1.001	256	2.1952e-5	3.872	6.9869e+6	1.000
256	1.8176e-5	3.967	4.3197e+7	-	512	5.5836e-6	3.932	2.7961e+7	-

methodology with potential application in various ACG contexts. Thereby obviating the need for ghost cells. The use of lexicographical enumeration clearly defines matrix bands, unlike Hash Table construction. Although the number of non-zeros is the same, structured bands improve matrix efficiency. This work was partially supported by VIIS-UDENAR.

References

[1] M. Berger and P. Colella. "Local adaptive mesh refinement for shock hydrodynamics". In: **Journal of computational Physics** 82.1 (1989), pp. 64–84.

[2] P. Calegari and A. Franco. "Generação de malhas com refinamento adaptativo uando tabelas de dispersão". In: **Proceeding Series of the Brazilian Society of Computational and Applied Mathematics**. 2021, pp. 1–7. DOI: 10.5540/03.2021.008.01.0421.

[3] A. Coco, S. Ekström, G. Russo, S. Serra-Capizzano, and S. C. Stissi. "Spectral and norm estimates for matrix-sequences arising from a finite difference approximation of elliptic operators". In: **Linear Algebra and its Applications** 667 (2023), pp. 10–43. ISSN: 0024-3795.

[4] R. Egan, A. Guittet, R. Temprano-Coleto, T. Isaac, F. J. Peaudecer, J. R. Landel, P. Luzzatto-Fegiz, C. Burstedde, and F. Gibou. "Direct numerical simulation of incompressible flows on parallel Octree grids". In: **Journal of Computational physics** 428 (2020), pp. 3–29. DOI: 10.1016/j.jcp.2020.110084.

[5] M. F. Mokbel and W. G. Aref. "Irregularity in multi-dimensional space-filling curves with applications in multimedia databases". In: **Proceedings of the tenth international conference on Information and knowledge management**. 2001, pp. 512–519.

[6] C. P. Ordoñez. "Creación y validación de estructuras de datos para aproximar EDP con diferencias finitas en mallas adaptativas". Master dissertation. UPRM, 2024.

[7] C. M. Rúa-Alvarez. "Simulação computacional adaptativa de escoamentos bifásicos viscoelásticos". PhD thesis. IME/USP, 2013.

[8] F. Sousa, C. Lages, J. Ansoni, A. Castelo, and A. Simao. "A finite difference method with meshless interpolation for incompressible flows in non-graded tree-based grids". In: **Journal of Computational physics** 396 (2019), pp. 848–866. DOI: 10.1016/j.jcp.2019.07.011.

[9] W. Y. Tey, N. A. Che S., Y. Asako, M. W. Muhieldeen, and O. Afshar. "Moving Least Squares Method and its Improvement: A Concise Review". In: **Journal of Applied and Computational Mechanics** 7.2 (2021), pp. 883–889. ISSN: 2383-4536.

[10] D. Wanta, W.T. Smolik, J. Kryszyn, P. Wróblewski, and M. Midur. "A Finite Volume Method using a Quadtree Non-Uniform Structured Mesh for Modeling in Electrical Capacitance Tomography". In: **Proc. Natl. Acad. Sci., India, Sect. A Phys** 92 (2022), pp. 443–452.

