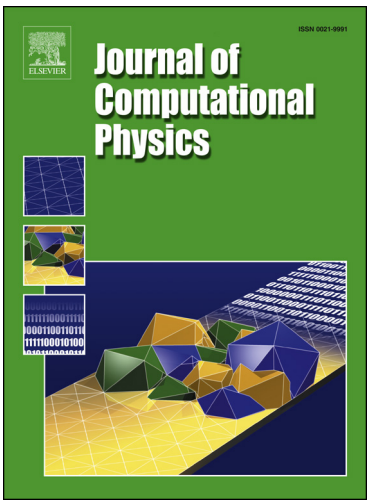
Accepted Manuscript



A finite difference method with meshless interpolation for incompressible flows in non-graded tree-based grids

F.S. Sousa, C.F.A. Lages, J.L. Ansoni, A. Castelo, A. Simao

PII: S0021-9991(19)30496-6
DOI: https://doi.org/10.1016/j.jcp.2019.07.011
Reference: YJCPH 8812

To appear in: *Journal of Computational Physics*

Received date: 21 November 2018
Revised date: 3 July 2019
Accepted date: 4 July 2019

Please cite this article in press as: F.S. Sousa et al., A finite difference method with meshless interpolation for incompressible flows in non-graded tree-based grids, *J. Comput. Phys.* (2019), https://doi.org/10.1016/j.jcp.2019.07.011

Highlights

- A new finite difference method with meshless interpolations is presented
- Meshless interpolations are used to compute weights for the finite difference stencil
- Good alternative for geometric interpolations on non-graded quadtree/octree meshes
- 2D and 3D results show 2nd order convergence for both the solution and its gradient
- The method is extended for the Navier-Stokes model with good convergence results

A finite difference method with meshless interpolation for incompressible flows in non-graded tree-based grids

F. S. Sousa[a,*], C. F. A. Lages[a], J. L. Ansoni[a], A. Castelo[a], A. Simao[b]

[a]*Department of Applied Mathematics and Statistics, Institute of Mathematical and Computer Sciences, University of São Paulo at São Carlos, Brazil*
[b]*Department of Computer Systems, Institute of Mathematical and Computer Sciences, University of São Paulo at São Carlos, Brazil*

Abstract

Tree-based mesh grids bring the advantage of using fast cartesian discretizations, such as finite differences, and the flexibility and accuracy of local mesh refinement. The main challenge is how to adapt the discretization stencil near the interfaces between grid elements of different sizes, which is usually solved by local high-order geometrical interpolations. These interpolations depend on the distribution of cells in the vicinity of the point of interest, hence they are site-specific and can become quite complex in three-dimensional simulations, specially when dealing with staggered unknown arrangements. Most methods usually avoid this by limiting the mesh configuration (usually to graded quadtree/octree grids), reducing the number of cases to be treated locally. In this work, we propose a robust method based on a moving least squares meshless interpolation technique, which is employed to compute the weights of the finite difference approximation in a given hierarchical grid, allowing for complex mesh configurations, still keeping the overall order of accuracy of the resulting method. Numerical convergence tests and application to fluid flow simulations are performed to illustrate the flexibility, robustness and accuracy of this new approach.

Keywords: Finite difference methods, Meshless interpolation, Moving least

*Corresponding author

squares, Tree-based grids, Navier-Stokes equations, Projection method

1. Introduction

The numerical solution of partial differential equations in general grids have been sought by many researchers in last decades. Many schemes try to combine efficiency and simplicity with the flexibility of unstructured mesh grids. A major advantage is the ability to locally refine the mesh, improving the accuracy in specific regions without drastically increasing the number of unknowns.

Among all possible ways of discretizing the spatial domain (simplicial meshes, curvilinear meshes, chimera, and even meshless, among others), cartesian tree-based hierarchical grids are a common choice. They allow for the development of finite difference or finite volume methods, without the hassle of mapping and transforming distorted elements or dealing with general and complicated stencils, as happens in non-cartesian grids. Since fluxes are usually computed in facets aligned with the cartesian axis, numerical schemes are usually simpler to derive. Still, these facets are usually shared by different number of elements in each side, which is the main difficulty for numerical methods for PDEs in such grids. Different techniques to deal with this problem have been developed in literature, most of them restricted to quadtree (in 2D) or octree (in 3D) meshes, that are special cases of hierarchical grids represented by quadtree/octree data structures. Despite this restriction, these tree-based data structures are general enough and still a suitable choice for adaptive grids and moving fronts [1, 2, 3, 4, 5, 6, 7].

Another type of hierarchical grids that share most of the difficulties above are the block-structured meshes. They are composed of the superposition of patches of structured grids of different resolutions, "glued" together by interpolation schemes [8, 9, 10]. Such methods take advantage of the induced matrix structure

of the grid, not only demanding a careful mapping between all blocks, but also adding a restriction on the adaptability of the mesh geometry.

Finite difference and finite volume methods in quadtree and octree meshes for fluid flow simulations were first developed in late 1990's and early 2000's ([1, 2, 11]). The main difficulty is to modify the FD/FV stencils near the coarse/fine interfaces accordingly, which is usually performed by high-order interpolations. Since these interpolations are mostly geometrical, they heavily rely on the number of cells shared by each facet in the grid. Therefore, many of these earlier schemes were developed in graded quadtree/octree meshes (see [2]), which means that the characteristic mesh size reduction factor is restricted to 2:1 in coarse-to-fine interfaces. Thus, the number of possible configurations of neighboring cells is reduced, allowing for hardcoding the interpolation formula in each specific case. As well stated by Batty in [12], this restriction leads to an unnecessary increase in the overall number of cells, motivating the recent development of numerical methods for non-graded quadtree/octree meshes, such as seen in [5, 13] for fluid flow simulations. However, while these numerical approaches issue 2nd order convergence of the solved variables, their gradient does not converge as such. Improved methods can be found in [12, 14], presenting more sophisticated geometrical interpolations, however still only for the numerical solution of Poisson equations.

Another way to overcome the limitations of geometrical interpolation is to take advantage of meshless interpolations. Fully meshless methods are thoroughly available in the literature, however they still pose issues regarding convergence and consistency, that are not completely solved (see for instance [15, 16, 17, 18, 19, 20, 21]). Conversely, much less references are found in hybrid methods, that try to combine the simplicity of finite difference schemes with meshless interpolations. Still, some remarkable work was developed in this di-

rection, mostly applied to the discretization of complex objects immersed in a cartesian grid and in joining structured grids together (as seen in [22, 23, 24, 25]).

The present work brings a new development in the same direction. We employ meshless interpolations computed via moving least squares (MLS) in order to modify finite difference stencils near the coarse-fine grid interfaces, as happens in hierarchical meshes. The developed methodology is further investigated and compared with non-graded octree-based methods, showing the advantages of our new methodology in terms of accuracy and robustness, from simple Poisson problems, to fluid flows in complex geometries.

2. Finite difference approximation in hierarchical grids

2.1. Hierarchical grids (HiG)

Our goal here is to present a general finite difference method that can be applied to cartesian hierarchical grids represented by tree data structures. One example of the generality of such hierarchical grid (HiG) can be seen in Fig. 1, as well as the tree data structure representing it. In this data structure, each leaf in the tree can be geometrically partitioned in any matrix arrangement of cells. Particular cases are the quadtree in 2D and octree in 3D, in which this division is restricted to four cells in 2D, and eight in 3D. Such general grids impose difficulties for numerical schemes, specially those based on cartesian finite difference (FD) approximations. They usually require the computation of spacial interpolations in the unknown points of the FD stencil, that heavily relies on the geometrical characteristics of the grid.

In order to avoid this geometrical dependence, we introduce a method based on interpolations in a neighboring point cloud, requiring no geometry or topology information, that is performed by an efficient MLS interpolation scheme.

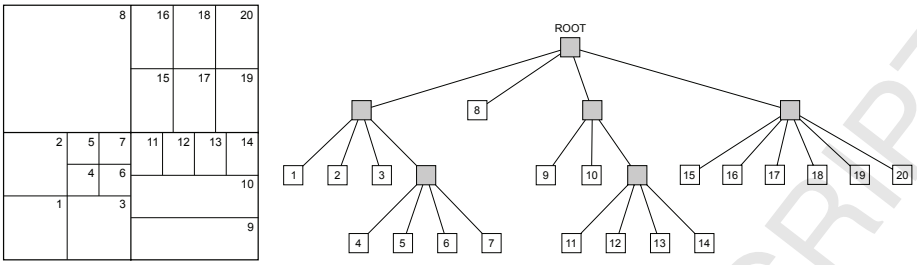


Figure 1: HiG tree data structure.

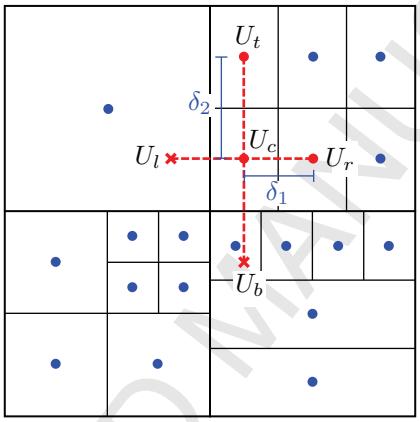


Figure 2: Finite difference 2nd order stencil discretization of a Poisson equation, where unknowns are located at cell centers. In this figure, δ_1 and δ_2 are the sizes of the cell containing the center point of the stencil.

2.2. Approximating derivatives in HiGs

To introduce the procedure to approximate derivatives with point cloud interpolation, suppose we need to solve a Poisson equation of the form

$$\nabla^2 u(x, y) = f(x, y) \tag{1}$$

in a rectangular domain $\Omega \in \mathbb{R}^2$ with suitable boundary conditions. Suppose we want to apply a standard 2nd order finite difference method to approximate u at the centers of the cells defined in Fig. 2, resulting in the numerical

approximation

$$\frac{1}{\delta_1^2}\left(U_l - 2U_c + U_r\right) + \frac{1}{\delta_2^2}\left(U_t - 2U_c + U_b\right) = f_c \ , \tag{2}$$

where U_l, U_r, U_t, U_b, U_c are the approximation values of u at left, right, top, bottom and center position of the 5-point stencil, with sizes δ_1 in the x-direction and δ_2 in the y-direction. These sizes are defined as the corresponding sizes of the cell were the center point of the stencil lies. If the neighboring cells have the same size, the unknowns are aligned and no special treatment is required, resulting in the standard 5-point finite difference stencil. In a grid as general as the one in Fig. 2, we see that the approximation of u does not coincide with some of the mesh grid points (such as U_l and U_b), and those have to be approximated by some kind of interpolation of the grid unknowns in the vicinity of U_c. Such interpolation can be performed as

$$U_l = \sum_{k \in \mathcal{I}_l} w_k^l U_k \quad \text{and} \quad U_b = \sum_{k \in \mathcal{I}_b} w_k^b U_k \tag{3}$$

where $\mathcal{I}_l = \{i_k, k = 1, \ldots N_l\}$ and $\mathcal{I}_b = \{j_k, k = 1, \ldots, N_b\}$ are the sets of indexes for the unknowns that are in the vicinity of U_c for each approximation. The coefficients w_k^l and w_k^b are computed by the interpolation procedure to be described later. The number of neighbors N_l and N_b depend on how many points are needed to keep the order of accuracy of the overall approximation. In a 2nd order approximation such as this, these interpolations should be at least 3rd order accurate, which means that, in 2D, we will need at least 6 points to completely determine all coefficients of a 2nd degree interpolating polynomial, which will have a 3rd-order-accurate error. The final approximation will finally

read

$$\frac{1}{\delta_1^2}\left(\sum_{k\in\mathcal{I}_l} w_k^l U_k - 2U_c + U_r\right) + \frac{1}{\delta_2^2}\left(U_t - 2U_c + \sum_{k\in\mathcal{I}_b} w_k^b U_k\right) = f_c \qquad (4)$$

or, in terms of the total number N_u of unknowns,

$$\sum_{k=1}^{N_u} A_{ck} U_k = f_c \ , \qquad (5)$$

$c = 1, \ldots, N_u$, that will result in a linear system of the form $A\mathbf{u} = \mathbf{f}$ where each line of A comes from approximations similar to Eq. (4).

This procedure is easily generalized for any finite difference stencil in hierarchical meshes. The number of neighbors of U_c is computed based on the expected order of accuracy. If there is not enough unknowns to keep the overall order of the finite difference approximation (that can occur for example in coarse grids), an adaptive stencil can be implemented, or a lower order interpolation can be computed, for the sake of robustness. In the next section we show how to efficiently compute the weights involved in the meshless interpolation.

2.3. Discrete Moving Least Squares

The procedure to complete the finite difference approximation in arbitrary grids depends heavily on high order interpolations performed in the vicinity of the unknown of interest. In our method, we perform these interpolations by a Moving Least Squares (MLS) procedure. Although computation of MLS basis functions is described in many works [15, 16, 18, 21], we introduce here an efficient way of computing them for the high order interpolation needed by the finite difference scheme.

Let us consider a set of n linearly independent smooth interpolation functions $\Phi_i : \mathbb{R}^d \to \mathbb{R}$. Our goal is to interpolate the value of a function $u : \mathbb{R}^d \to \mathbb{R}$ such

that the approximation is given by

$$U(\mathbf{x}) = \sum_{j=1}^{n} c_j \, \Phi_j(\mathbf{x}) \; . \tag{6}$$

In our implementations, polynomial functions of several variables $\Phi_i(\mathbf{x}) = x_1^{\beta_1^i} \cdot x_2^{\beta_2^i} \cdots x_d^{\beta_d^i}$ are used, where $\beta_j^i \in \mathbb{N}$, $j = 1, \ldots, d$, such that polynomials will be of maximum degree $K = \max \sum_{j=1}^{d} \beta_j^i$.

Given a set of m known points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m \in \mathbb{R}^d$, $m > n$, and m associated function values $u_1 = u(\mathbf{x}_1), u_2 = u(\mathbf{x}_2), \ldots, u_m = u(\mathbf{x}_m)$, the procedure to interpolate u at $\mathbf{x}$ using MLS is to minimize the error function

$$E(\mathbf{c}) = \|U - u\|_2^2 = \langle U - u, U - u \rangle_\lambda = \sum_{i=1}^{m} \left(U(\mathbf{x}_i) - u_i\right)^2 \lambda_i(\mathbf{x}) \tag{7}$$

that is a norm induced by a weighted inner product defined over the weight functions

$$\lambda_i(\mathbf{x}) = \frac{1}{\|\mathbf{x} - \mathbf{x}_i\|_2} \tag{8}$$

that depends on the position $\mathbf{x}$ (explaining the term *moving* in MLS). Therefore, the minimizer $\mathbf{c} = (c_1, \ldots, c_n)$ depends on the position $\mathbf{x}$ as well. In practice, a threshold value $\varepsilon > 0$ is used to avoid the weight functions to become indefinite. Let us define matrices $W(\mathbf{x}) \in \mathbb{R}^{m \times m}$, which is the diagonal weight matrix, defined by $W_{ij}(\mathbf{x}) = \delta_{ij} \sqrt{\lambda_i(\mathbf{x})}$, and $P \in \mathbb{R}^{m \times n}$ the evaluation of the polynomial basis over the points $\mathbf{x}_i$, with components $P_{ij} = \Phi_j(\mathbf{x}_i)$. Furthermore, defining the vector $\mathbf{u} = (u_1, \ldots, u_m)$, the error becomes

$$E(\mathbf{c}) = \|WP\mathbf{c} - W\mathbf{u}\|_2^2 \; . \tag{9}$$

The solution that minimizes this error is then given by

$$\mathbf{c}(\mathbf{x}) = (WP)^{\dagger} W\mathbf{u} \tag{10}$$

where $(\cdot)^{\dagger}$ denotes the pseudo-inverse of Moore-Penrose of a matrix [26]. In fact, since matrix WP is non-square, we can perform its QR decomposition, resulting in

$$WP = Q \begin{bmatrix} R \\ 0 \end{bmatrix} = \begin{bmatrix} Q_{\|} & Q_{\perp} \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix}, \tag{11}$$

where $Q \in \mathbb{R}^{m\times m}$ is an orthogonal matrix, $Q_{\|} \in \mathbb{R}^{m\times n}$, $Q_{\perp} \in \mathbb{R}^{m\times(m-n)}$ are submatrices of Q, and $R \in \mathbb{R}^{n\times n}$ is an upper triangular matrix. The error in Eq. (9) is then rewritten as

$$E(\mathbf{c}) = \left\| R\mathbf{c} - Q_{\|}^{t} W\mathbf{u} \right\|_2^2 + \left\| Q_{\perp}^{t} W\mathbf{u} \right\|_2^2 \tag{12}$$

such that the solution is given by

$$\mathbf{c}(\mathbf{x}) = R^{-1} Q_{\|}^{t} W\mathbf{u} \ , \tag{13}$$

and the minimum error value is $E(\mathbf{c}) = \|Q_{\perp}^{t} W\mathbf{u}\|_2^2$.

This procedure however results in the coefficients $c_j(\mathbf{x})$ in the linear combination of the polynomial basis functions $\Phi_j(\mathbf{x})$ in Eq. (6). Since we intend to couple these approximations to the finite difference approximations of derivatives, we should compute the interpolation coefficients w_i that form the linear combination of the values u_i. From Eqs. (6) and (13), we have

$$U(\mathbf{x}) = \mathbf{c}^t \mathbf{\Phi} = \mathbf{u}^t \underbrace{W Q_{\|} R^{-t} \mathbf{\Phi}}_{\mathbf{w}} = \sum_{i=1}^{m} w_i(\mathbf{x}) u(\mathbf{x}_i) \ . \tag{14}$$

The approximation $U(\mathbf{x})$ should also interpolate $u_i = u(\mathbf{x}_i)$ at the points $\mathbf{x}_i$, such that Eq. (6) yields

$$u_i = U(\mathbf{x}_i) = \sum_{j=1}^{n} c_j \Phi_j(\mathbf{x}_i) \ , \tag{15}$$

and therefore

$$U(\mathbf{x}) = \sum_{i=1}^{m} w_i u(\mathbf{x}_i) = \sum_{i=1}^{m} w_i \sum_{j=1}^{n} c_j \Phi_j(\mathbf{x}_i) = \sum_{j=1}^{n} c_j \sum_{i=1}^{m} w_i \Phi_j(\mathbf{x}_i) \tag{16}$$

where the dependency of c_j and w_i on $\mathbf{x}$ was dropped by the sake of notation. Comparing to the definition of $U(\mathbf{x})$ in Eq. (6) results

$$\sum_{j=1}^{n} c_j \Phi_j(\mathbf{x}) = \sum_{j=1}^{n} c_j \sum_{i=1}^{m} w_i \Phi_j(\mathbf{x}_i) \tag{17}$$

which is true for every $\mathbf{c}$ that is a solution of the least squares problem in Eq. (9), allowing us to conclude that

$$\Phi_j(\mathbf{x}) = \sum_{i=1}^{m} w_i \Phi_j(\mathbf{x}_i) \ . \tag{18}$$

The main consequence of Eq. (18) is that interpolation by the MLS preserve important properties such as the exact recovery of the basis functions. Particularly, since Eq. (18) is true for constant polynomial basis functions (say, $\Phi_1(\mathbf{x}) = 1$), the weights w_i satisfy the partition of the unity property, i.e.,

$$\sum_{i=1}^{m} w_i = 1 \ . \tag{19}$$

The coefficients w_i computed in that way are then used to assemble the finite difference approximation matrix A as in Eq. (5). The procedure to compute

the coefficients w_i is summarized in Algorithm 1. These values are then used in the discretization stencil when needed, like in the example of Eq. (4). Clearly, $w_i = w_i(\mathbf{x})$, which means that this algorithm should be executed for each approximation of $U(\mathbf{x})$ needed. The most computational demanding procedures in this algorithm are the QR decomposition and the solution of a triangular system. They are however local and therefore can be easily computed in parallel, and since the problems are relatively small, they can take advantage of GPU architecture as well. As long as the mesh is static, this procedure is computed only once as a pre-processing step, and the interpolation weights can be cached.

Algorithm 1 Computation of coefficients w_i using MLS interpolation.

1: Given $\mathbf{x}_1, \ldots, \mathbf{x}_m, \mathbf{x} \in \mathbb{R}^d$, $\Phi_1, \ldots, \Phi_n \in \mathcal{P}_K(\mathbb{R}^d)$, $m > n$
2: Compute $W = \left[\delta_{ij}\sqrt{\lambda_i(\mathbf{x})}\right]$, $P = [\Phi_j(\mathbf{x}_i)]$, $\lambda_i(\mathbf{x})$ given by Eq. (8)
3: Perform the QR decomposition of matrix WP, resulting in R and $Q_\parallel$
4: Compute $\mathbf{\Phi} = (\Phi_1(\mathbf{x}), \ldots, \Phi_n(\mathbf{x}))^t$
5: Solve triangular system $R^t\mathbf{d} = \mathbf{\Phi} \Rightarrow \mathbf{d} = R^{-t}\mathbf{\Phi}$
6: Compute $\mathbf{w} = WQ_\parallel\mathbf{d}$

Remark 1. If $U(\mathbf{x})$ is a complete polynomial of degree $K = \max\sum_{j=1}^{d}\beta_j^i$, i.e., when it contains all monomials of degree s, for $s = 0, \ldots, K$, then the coefficients w_i will be invariant under rigid transformations (rotations, reflections and translations). The proof can be easily derived by induction on the polynomial degree K. This result can be further used to build a hash table for the cached interpolations, reducing even further the overall computing cost. This feature is even more efficient in graded quadtree/octree meshes, since there are just few different configurations to be considered. Additionally, since all basis functions are fully recovered by the MLS interpolation, PDEs with zero truncation errors are solved exactly, as in standard finite difference techniques.

Remark 2. The points $\mathbf{x}_1, \ldots, \mathbf{x}_m$ are taken in a vicinity of point $\mathbf{x}$, that is defined as a circle centered in $\mathbf{x}$ with a given interpolation radius. The number of

points inside this vicinity should be enough to at least recover the basis functions used in the interpolation. If this is not satisfied, the result is reflected as zeroes in the diagonal of matrix R (from QR decomposition in Eq. (11)), which turns out to be singular. In that case, this vicinity can be adaptively increased to overcome this drawback, adding robustness to the computed approximation. In our results, the initial radius is taken as $3 \times \max\{\delta_1, \ldots, \delta_d\}$, where δ_i are the corresponding sizes of the cell containing $\mathbf{x}$, and every time we detect zeroes on the diagonal of R, the interpolation radius is increased by $\min\{\delta_1, \ldots, \delta_d\}$.

Remark 3. It is clear that the matrices produced by such procedure are sparse, with sparsity depending on the size of the neighborhoods used in the MLS, and are not symmetric. Therefore iterative linear solvers such as the Generalized Minimum Residual Method (GMRES) or the Bi-Conjugate Gradient Method (Bi-CG) and its variants are good alternatives to solve the resulting linear systems.

Remark 4. Although the weights w_i computed by the MLS procedure satisfy the partition of the unity property, there can be negative values, hence it is not possible to ensure the diagonal dominance of the resulting finite difference linear system. It is possible however to extend the degrees of freedom of the MLS problem over the kernel of $(WP)^t$ such that solutions with the restriction $w_i \geq 0$ can be computed, ensuring the diagonal dominance of the resulting matrix. Our numerical experience with many different computations with this method indicates that this workaround in somewhat costly and completely unnecessary.

2.4. Dirichlet and Neumann boundary conditions

Dirichlet and Neumann type boundary conditions are handled much like standard techniques in finite difference context. Each finite difference stencil, when hitting the boundaries, end up in requiring points outside the computational domain. In the case of Dirichlet-type boundary conditions, the known

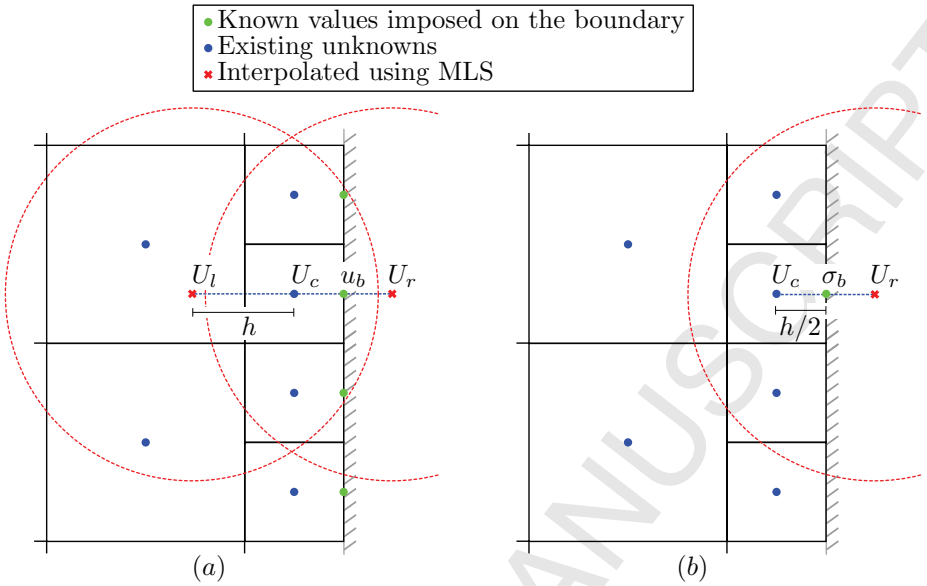
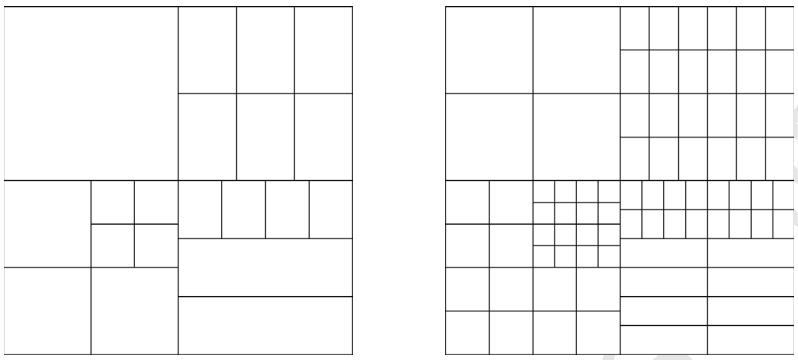


Figure 3: Discretization of the (a) Dirichlet and (b) Neumann boundary conditions using the meshless MLS interpolation.

values are added on the boundary, so that they are taken into account on every interpolation in the vicinity of the boundaries. An example is illustrated in Fig. 3(a), where the value u_b is imposed on all green dots (boundary values) such that the required MLS interpolations of U_l and U_r take those into account. This is very similar to standard finite difference techniques, where U_r would be computed such that the linear interpolation between U_c and U_r is exactly u_b.

In case of Neumann-type boundary conditions, say $\frac{\partial u}{\partial x} = \sigma_b$ on the right boundary, as depicted in Fig. 3(b), we discretize this boundary condition by standard finite differences, arriving at $U_r = U_c + h\sigma_b$, h being the size of the smaller cells near the boundary. Therefore, the stencil centered at U_c is modified accordingly. Variations of these two techniques are straightforwardly applied to different cell configurations and different unknowns positions with minor adaptations.

The interpolations are also affected by the lack of available points when



(a) $h = 1/4$ (b) $h = 1/8$

Figure 4: Two mesh refinements considered in the solution of the Poisson equation in a general HiG.

interpolating near boundaries, turning R singular and lowering the order of approximation. In such cases, the interpolation radius is increased to complete the minimum requirements to keep the order of MLS interpolation. As seen in results later on, these simple techniques produce accurate and robust representation of boundary conditions.

2.5. Verification: Poisson equation in a general HiG

In summary, the described method works as follows: whenever a finite difference stencil is used to compute terms of the linear system matrix A, if the required point belonging to the stencil is available, it is used straight away. If it is missing, as occurring in the interface between different mesh sizes, MLS interpolations are used to compute the final weights entering matrix A.

To test this methodology and show the robustness of the method for a sufficiently general grid, we started with grid shown in Fig. 1 to discretize a square $\Omega = [-1, 1]^2$, which we attribute a reference mesh size of $h = 1/4$, the length of the smallest edge.

The described numerical method was then applied to this grid to solve a

Table 1: Errors and estimated order of convergence (EOC) for the Poisson equation in a general HiG.

h	$\|u-u_h\|_{L_2}$	EOC	$\|u-u_h\|_{L_\infty}$	EOC
1/4	1.29e-01	-	1.21e-01	-
1/8	7.06e-03	4.1	8.66e-03	3.8
1/16	1.30e-03	3.3	1.66e-03	3.1
1/32	2.32e-04	3.0	3.14e-04	2.9

Poisson equation (1) with analytical solution given by

$$u(x,y) = \sin(x)\cos(y) \tag{20}$$

with source term $f(x,y)$ and Dirichlet type boundary conditions manually computed as to result in this analytical solution. The MLS interpolations are performed with second degree polynomials. One can note that there are several interfaces with non-coincident cells (with the occurrence of many points, which are mostly known in finite element literature as "hanging nodes"). Refinement is carried out, subdividing uniformly each cell into four cells, to assess the numerical convergence for this case, with results reported in Table 1. The refined grids with reference sizes $h = 1/4$ and $h = 1/8$ are depicted in Fig. 4.

These results confirm the robustness of this method even when the mesh is divided in mostly non-coincident cells, with intensive use of interpolations all across interfaces. Evidently, this method can be further applied to connect domains discretized by different (non-coincident) meshes, however this type of application is not the main concern of the current work. We then move on to present more detailed results for non-graded quadtree/octree grids, in order to further extend this methodology to the solution of Navier-Stokes equations.

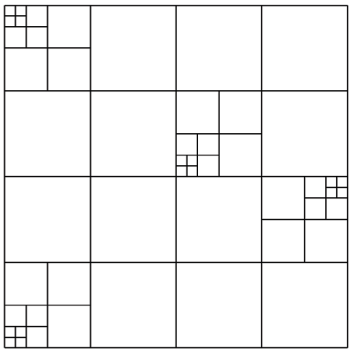


Figure 5: Non-graded mesh used in the Poisson convergence test.

2.6. Verification: convergence in non-graded quadtree and octree meshes

One of the main advantage of our methodology is the ability to preserve expected order of convergence without any special treatment or geometrical interpolations. This is specially advantageous when dealing with non-graded meshes, i.e., meshes that vary abruptly, with large cells being neighbors of very small ones. Any geometrical high-order interpolation would be site-specific, and difficult to program in higher dimensions.

To show that our method is capable of solving non-graded meshes, we performed tests by solving the Poisson equation (1) in non-graded quadtrees, as performed by other authors, such as Min et al. [14] and Batty [12]. The non-graded quadtree mesh is depicted in Fig. 5, discretizing a domain $[0, \pi]^2$ where (1) is solved with source term $f(x, y)$ computed as to yield an exact solution $u(x, y) = e^{-x-y}$, using Dirichlet-type boundary conditions. We computed numerical results for this problem using polynomial meshless interpolation described earlier, with polynomials of degree 2 and 3, that are reported in Tables 2 and 3 respectively. The 2nd order accuracy is evident for u_h and ∇u_h when using 3rd degree polynomials, with a slight drop in the order of convergence for ∇u_h to around 1.6 when using 2nd degree polynomials.

To assess how the resulting linear systems are affected by our interpolation

Table 2: Errors and estimated order of convergence (EOC) for the numerical solution of the Poisson equation and its gradients, computed with 2nd degree interpolations in a non-graded quadtree with Dirichlet boundary conditions.

h	$\|u - u_h\|_{L_\infty}$	EOC	$\|\nabla u - \nabla u_h\|_{L_\infty}$	EOC
$\pi/32$	2.68e-2	-	2.57e-2	-
$\pi/64$	6.08e-3	2.1	1.07e-2	1.3
$\pi/128$	1.15e-3	2.3	3.18e-3	1.5
$\pi/256$	2.39e-4	2.3	1.03e-3	1.5
$\pi/512$	5.59e-5	2.2	3.24e-4	1.6
$\pi/1024$	1.43e-5	2.3	9.85e-5	1.6
$\pi/2048$	3.73e-6	2.1	3.03e-5	1.6
$\pi/4096$	8.72e-7	2.1	8.76e-6	1.6

Table 3: Errors and estimated order of convergence (EOC) for the numerical solution of the Poisson equation and its gradients, computed with 3rd degree interpolations in a non-graded quadtree with Dirichlet boundary conditions.

h	$\|u - u_h\|_{L_\infty}$	EOC	$\|\nabla u - \nabla u_h\|_{L_\infty}$	EOC
$\pi/32$	1.08e-2	-	7.01e-3	-
$\pi/64$	1.80e-3	2.6	2.36e-3	1.6
$\pi/128$	3.18e-4	2.5	6.96e-4	1.7
$\pi/256$	7.65e-5	2.4	1.70e-4	1.8
$\pi/512$	1.90e-5	2.3	4.40e-5	1.8
$\pi/1024$	5.19e-6	2.2	1.08e-5	1.9
$\pi/2048$	1.33e-6	2.2	2.81e-6	1.9
$\pi/4096$	3.36e-7	2.1	7.08e-7	1.9

Table 4: Observed condition numbers for the algebraic linear systems obtained with our method, for the Poisson convergence test using mesh from Figure 5.

h	Uniform grid estimate	Condition numbers from [12]	Observed condition numbers
$\pi/32$	4.205e+01	7.171e+01	1.116e+03
$\pi/64$	1.682e+02	2.815e+02	6.318e+03
$\pi/128$	6.728e+02	1.193e+03	3.297e+04
$\pi/256$	2.691e+03	4.713e+03	1.432e+05
$\pi/512$	1.076e+04	1.887e+04	5.960e+05
$\pi/1024$	4.306e+04	7.550e+04	2.419e+06
$\pi/2048$	1.722e+05	3.020e+05	9.754e+06
$\pi/4096$	6.889e+05	1.208e+06	3.921e+07

method, we compare the condition numbers of the linear system matrices to those observed in [12] for this exactly same problem, which can be seen in Table 4. Our condition numbers are consistently 32 times greater than those obtained by the geometric interpolation in [12], and roughly 55 times greater than a linear system obtained by a uniform mesh discretization with the same resolution as the finest grid cell in the corresponding non-graded quadtree mesh. This increase in the condition number is expected, since all the unknowns in the neighborhood of the MLS interpolation are connected, appearing as non-zero entries in the corresponding line of the final matrix. For this case, the condition number appears to be scaling with $\kappa(A) \approx 23/h^2$, as compared to the classical result for uniform grids $\kappa(A) \approx 4/(\pi^2 h^2)$, being both $\mathcal{O}(1/h^2)$. This difference however has little impact on the number of iterations required for convergence of iterative linear solvers.

One can also look at the fill-in effect on the matrix due to the MLS interpolations, which can be measured in terms of the number of non-zero entries for the matrices produced by our method as compared to a standard finite difference discretization in uniform grids with the same number of unknowns. For the results reported in Table 3 using 3rd degree polynomials, the coarsest mesh $h = \pi/32$ yields a fill-in factor of about 6.2 times greater than the standard 5-point stencil, while for $h = \pi/1024$ this factor drops to about 1.6, and for the finest grid with $h = \pi/4096$, the number of non-zeros in the MLS matrix is only about 1.2 times greater than a reference uniform finite difference discretization.

These results are similar to the ones found in [12], which employs a geometrical diagonal interpolation scheme, being difficult to extend in 3D. Our method is general and can easily be programmed for any number of spatial dimensions. To give an evidence of convergence in higher dimensions, we solved a 3D Poisson problem in a cubical domain $[-\frac{\pi}{2}, \frac{\pi}{2}]^3$ with non-homogeneous Neumann

Table 5: Errors and estimated order of convergence (EOC) for the numerical solution of the 3D Poisson equation and its gradient, computed with 3rd degree interpolations in a non-graded octree with non-homogeneous Neumann boundary conditions.

h	$\|u - u_h\|_{L_\infty}$	EOC	$\|\nabla u - \nabla u_h\|_{L_\infty}$	EOC
$\pi/32$	2.14e-1	-	1.14e-1	-
$\pi/64$	5.22e-2	2.0	3.25e-2	1.8
$\pi/128$	1.23e-2	2.1	8.10e-3	2.0
$\pi/256$	2.12e-3	2.5	2.32e-3	1.8
$\pi/512$	4.62e-4	2.2	5.93e-4	2.0

boundary conditions, i.e.,

$$\nabla^2 u = f, \quad \check{\mathbf{n}} \cdot \nabla u = g \ , \tag{21}$$

where functions $f = f(x, y, z)$ and $g = g(x, y, z)$ are prescribed so that the manufactured exact solution for this problem is

$$u(x, y, z) = \cos(x)\cos(y)\cos(z) \ . \tag{22}$$

The convergence test is performed using the same non-graded mesh found in [4], originally employed for a Navier-Stokes problem. This initial mesh is depicted in Fig. 6 in which we associate a mesh size of $h = \pi/32$, the size of the shortest edge. Results are shown in Table 5, where one can observe a 2nd order convergence for both solution and its gradient.

The results presented in this section demonstrates the robustness and accuracy of our method for 2D and 3D Poisson problems with both Dirichlet and Neumann boundary conditions, in non-graded quadtree and octree meshes. Although not shown here, the extension of this method for even higher dimension Poisson problems is straightforward, with similar results. In the next sections, we extend this method for the incompressible Navier-Stokes equations using standard projection methods, followed by verification and convergence results.

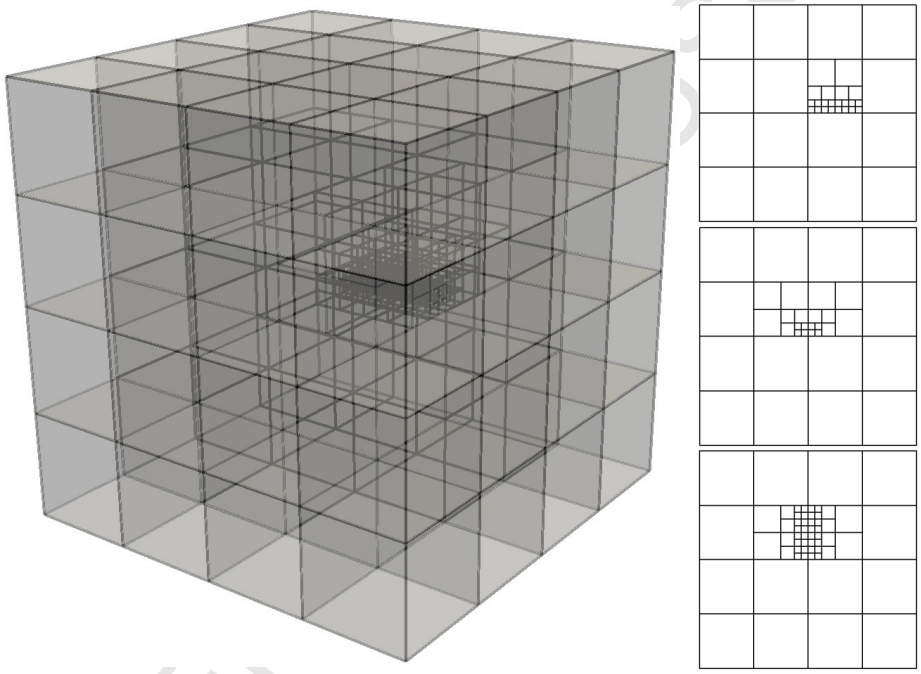


Figure 6: Non-graded mesh used for the 3D convergence test case, same as in [4]. On the left a 3D view of the mesh, and on the right three cut-planes corresponding to $x = 0$, $y = 0$ and $z = 0$ respectively, from top to bottom. They are oriented such that positive axis point always to the right and upwards.

3. Discretization of the Navier-Stokes equations

We are interested in the simulation of incompressible fluid flows using general hierarchical tree-based meshes, through our general meshless interpolation scheme, as discussed in last section. We begin with the well known model, the Navier-Stokes equations for newtonian fluid and incompressible flow, that in non-dimensional form reads

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \frac{1}{Re} \nabla^2 \mathbf{u} + \mathbf{f} \tag{23}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{24}$$

where $\mathbf{u} = \mathbf{u}(\mathbf{x}, t)$ denotes the velocity field and $p = p(\mathbf{x}, t)$ the pressure, both defined in a d-dimensional domain $\Omega \subset \mathbb{R}^d$, $d = 2, 3$. The flow depends on the source term $\mathbf{f} = \mathbf{f}(\mathbf{x}, t)$ that could be used to model gravity or any other body force, and on the parameter Re, known as *Reynolds number*, that represents the ratio between inertial and viscous forces acting on the flow. Usually, $Re = U_\infty L/\nu$, with U_∞ and L being characteristic velocity and length, and $\nu = \mu/\rho$ the kinematic viscosity, ratio between dynamic viscosity and density.

Many methods were developed since late 60's, resulting and many different approaches to deal with the strong coupling between velocity and pressure in the incompressible Navier-Stokes equations. A large number of such works implement the so called *Projection Method* or *Fractional Step Method* [27, 28, 29, 30, 31], which employs the well-known *Helmholtz-Hodge* decomposition to split the Navier-Stokes operator allowing the sequential computation of velocity and pressure.

3.1. Staggered grid arrangement

As performed in many different works, we employ a staggered grid arrangement of the unknowns, which yields a stable discretization for uniform grids.

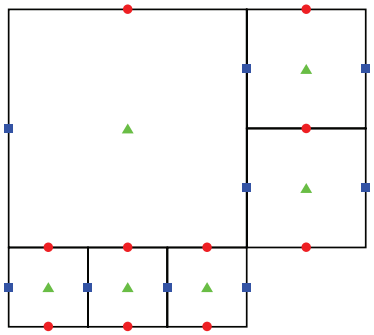


Figure 7: Illustration of the staggered unknown distribution in an hierarchical grid: ambiguities are solved with smaller-cell priority rule.

In this arrangement, pressures are evaluated in the center of the grid cells, while normal components of the velocity are evaluated at cell facets.

When storing the normal components at cell facets and numbering the unknowns, ambiguities can appear, as illustrated in Fig. 7. Besides being a sketch in 2D, the 3D case is easily derived as a trivial extension of the proposed ideas. All ambiguities are solved based on a smaller-cell priority, i.e., cells with smaller sizes keep their unknowns on shared facets with bigger cells. On the other hand, bigger cells loose their unknowns on shared facets with smaller cells, unless they coincide. Both situations are present in Fig. 7: the bigger cell lost its right (blue square) unknown, but kept its bottom (red circle) unknown, that coincides with central smaller cell. Although not impossible, this arrangement makes it difficult to extend the staggered grid to a general mesh such as in Fig. 1, therefore, for the Navier-Stokes solver, we will limit ourselves to meshes with the restriction that all shared facets divide exactly the whole facet of the larger cell, which is the case displayed in Fig. 7. The set of allowed meshes includes, but is not restricted to, quadtree and octree type meshes.

3.2. The Helmholtz-Hodge decomposition

The Helmholtz-Hodge decomposition theorem states that any vector field $\mathbf{f}$ such that $\int_{\partial\Omega} \mathbf{f} \cdot \mathbf{n} = 0$ can be decomposed as a sum of a divergence-free vector field $\mathbf{u}$ and a curl-free vector field $\mathbf{v}$, i.e., $\mathbf{f} = \mathbf{u} + \mathbf{v}$. Since $\mathbf{v}$ is curl-free, there exists a scalar function p in such a way that $\mathbf{v} = \nabla p$, yielding the famous form

$$\mathbf{f} = \mathbf{u} + \nabla p \tag{25}$$

with $\nabla \cdot \mathbf{u} = 0$ and the normal components of $\mathbf{f}$ and $\mathbf{u}$ coinciding at the boundary of Ω.

This decomposition is often used to design numerical methods to solve the Navier-Stokes equations by means of segregating the coupled system of equations, creating a sequential and therefore more efficient way to compute velocity and pressure. This is well known in the literature, and it is referred by as many names as there are variations. The most common terminology is "projection methods", referring to the projection induced by the Helmholtz-Hodge decomposition theorem, of one vector field into the direct sum of divergence-free and curl-free spaces (other terminologies are known as Chorin's method, MAC-type methods, fractional step methods, among others).

One of the main issues when employing the projection method in graded and non-graded tree-based meshes is the stability of the projection method by itself. For grids represented by quad/octrees, a sketch of a proof of stability, showing that the projection step does not introduce any spurious kinetic energy into the flow by itself, is shown in [5]. It relies on showing that there exist metrics (matrices) L_F and L_C such that the mimetic finite difference operators G and D satisfy the identity $L_F G = -\left(L_C D\right)^t$, which is further used to show that $\|\mathbf{u}^{n+1}\|_{L_F} \leq \|\mathbf{u}^*\|_{L_F}$, concluding the stability of the projection method. Although there is a small error in their definition of the metrics L_F and L_C

by assuming they are diagonal, which is not the case, the rest of the proof is correct (see [5]). Numerical observations support their theoretical result.

In a recent development, Olshanskii and co-workers [13] found that their discretization, which is based on the work of Lossasso et al. [2], presents numerical instabilities in the computation of the discrete Helmholtz-Hodge decomposition in a staggered arrangement. They end up creating and implementing an artificial filter to avoid those instabilities coming from their discretization (more details in [13]). More recently, Batty [12] presented numerical evidence of stability of its central scheme with diagonal interpolations by successively applying the Helmholtz-Hodge decomposition.

Since our work intends to discretize the Navier-Stokes equations in general tree-based grids, we are motivated to perform the same tests as in [4, 5, 12, 13] to evaluate the stability of our discretization method.

The tests are based on the decomposition of a known vector field $\mathbf{f}$ as in Eq. (25), by computing $\mathbf{u}$ and p. To achieve this, we have to first solve the Poisson equation

$$\nabla \cdot \nabla p = \nabla \cdot \mathbf{f} \ , \quad \nabla p \cdot \mathbf{n}|_{\partial\Omega} = 0 \tag{26}$$

to find p, and finally compute $\mathbf{u}$ from $\mathbf{u} = \mathbf{f} - \nabla p$. The test is performed in a square domain $\Omega = [0,1]^2 \subset \mathbb{R}^2$, discretized by a two-level refinement mesh, where the characteristic mesh size is set to $h = h_{\min}$ for $x > \frac{1}{2}$ and $h = 2h_{\min}$ for $x < \frac{1}{2}$, configuring the same setup as in [13]. In their work, they provide analytical expressions for $\mathbf{u}$ and p that are used to compute known field $\mathbf{f}$ and

$\nabla \cdot \mathbf{f}$, that are

$$
\begin{aligned}
u(x,y) &= \sin\left(\frac{2\pi(e^{ay}-1)}{e^a-1}\right)\left[1-\cos\left(\frac{2\pi(e^x-1)}{e-1}\right)\right]\frac{ae^{ay}}{2\pi(e^a-1)} \\
v(x,y) &= -\sin\left(\frac{2\pi(e^x-1)}{e-1}\right)\left[1-\cos\left(\frac{2\pi(e^{ay}-1)}{e^a-1}\right)\right]\frac{e^x}{2\pi(e-1)} \\
p(x,y) &= a\cos\left(\frac{2\pi(e^x-1)}{e-1}\right)\cos\left(\frac{2\pi(e^{ay}-1)}{e^a-1}\right)\frac{e^{a+1}}{(e-1)(e^a-1)}
\end{aligned}
\tag{27}
$$

where $\mathbf{u} = (u, v)^t$ and $a = 0.1$.

Discretizations of the Poisson equation (26) are performed in a staggered grid (p located at cell centers and normal components of $\mathbf{u}$ and $\mathbf{f}$ at cell facets, as described earlier). A standard 2nd order 5-point finite difference stencil is employed, with meshless interpolations using 2nd degree polynomials whenever needed. We present a comparison of our results with those published in [13], which are displayed in Figs. 8 and 9, in terms of the L_∞ and L_2 error norms for $\mathbf{u}$ and p, respectively. One can see that our method yields clean 2nd order convergence for all error norms considered. Despite the fact that L_∞ error norm for p is not shown in [13], we also computed errors with this norm to verify 2nd order convergence as well. One can clearly see that Olshanskii's method and both its variants Pressure Enrichment (PE) and Differential Filter (DF) did not perform well, at least for the reported results from [13]. Results display only first order convergence for the $\mathbf{u}$ error in L_∞ norm, and less-than-second order for the L_2 norm.

These results demonstrate the superiority of our method as compared to [13], specially in terms of stability of the calculations around the interfaces between cell refinements, even after the fixes proposed by the author.

The error for the x-component of the velocity vector can be seen in Fig. 10, where small oscillations can be detected in our method, coming from the unbalancing between fluxes across the coarse-to-fine interface. However they are no

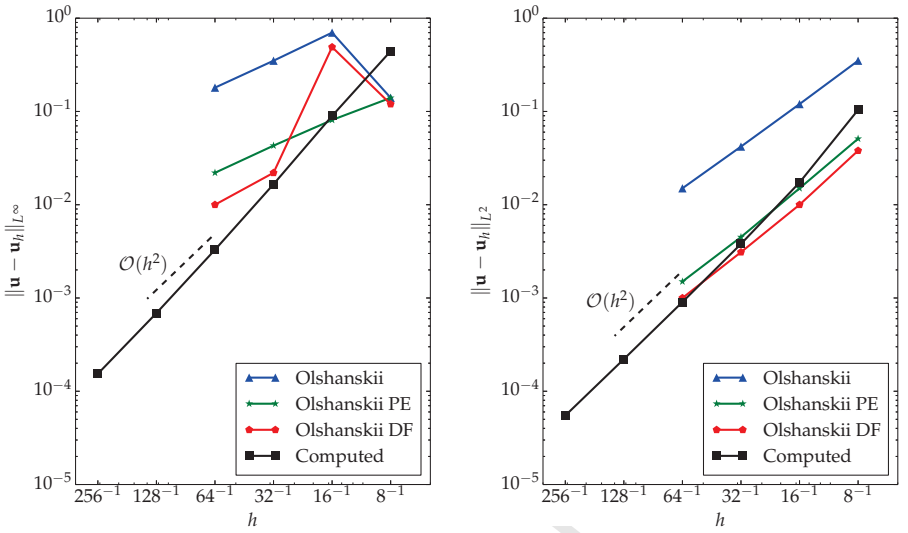


Figure 8: Error norms for the computed velocity field, as compared to Olshanskii's method and its variants.

bigger than the error elsewhere, and they do not increase (in fact they decrease) as the mesh is refined, which is a clear sign of stability for our discretization method.

Another numerical test for the stability of the projection via Helmholtz-Hodge decomposition can be found in the work of Min and Gibou [4], later reproduced by Guittet et al. [5] and more recently by Batty [12]. It comprises successive applications of the Helmholtz-Hodge decomposition, beginning with the initial vector field

$$\mathbf{u}^{(0)}(x,y) = \begin{bmatrix} \sin(x)\cos(y) + x(\pi - x)y^2\left(\frac{y}{3} - \frac{\pi}{2}\right) \\ -\cos(x)\sin(y) + y(\pi - y)x^2\left(\frac{x}{3} - \frac{\pi}{2}\right) \end{bmatrix}, \tag{28}$$

defined in $\Omega = [0,\pi]^2$, which is discretized by a non-graded mesh illustrated in Fig. 11. The procedure consists of solving a sequence of projections of the form

$$\mathbf{u}^{(k)} = \mathbf{u}^{(k+1)} + \nabla p^{(k+1)} \tag{29}$$

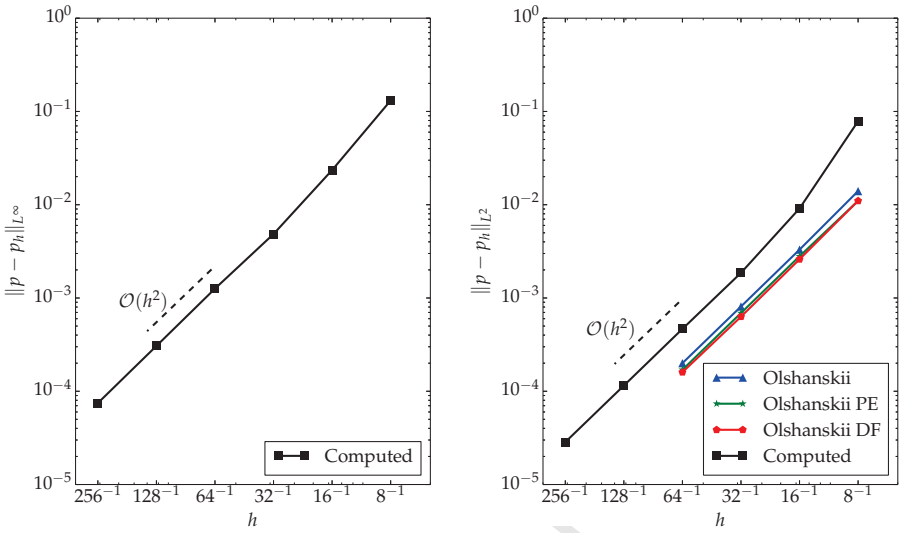


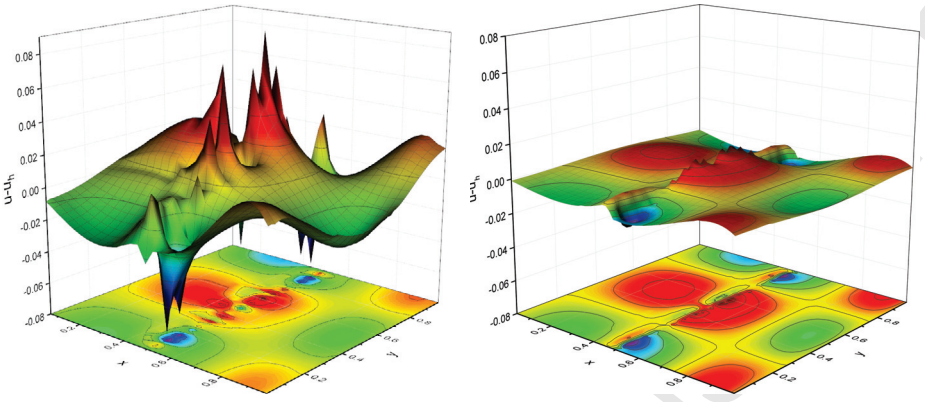
Figure 9: Error norms for the computed pressure, as compared to Olshanskii's method and its variants. Olshanskii et. al do not report pressure convergence in L_∞ norm.

where $\nabla \cdot \mathbf{u}^{(k+1)} = 0$, for $k \geq 0$. If the decomposition is stable, the error should stay stationary even over dozens of successive projetions. Results are reported in Fig. 12, where one can see the log of the error $\|\mathbf{u}^{(k+1)} - \mathbf{u}^{(k)}\|_{L_\infty}$, which stays the same for as far as one hundred iterations, the same behavior as the results presented in [12] and [4].

3.3. Projection method for the Navier-Stokes equations

With the stability of the Helmholtz-Hodge decomposition verified, we move on to the discretization of the Navier-Stokes equations. We will briefly describe here the projection method used in the Navier-Stokes discretization, known as *incremental projection method* or *pressure correction method* (the reader is referred to throughout discussions about different types of projection methods in [28, 29, 31, 32, 33] and references therein).

Upon discretizing Eq. (23) in time using, for instance, a first order semi-implicit discretization, the idea of the incremental projection method is to use the pressure at the previous time step, which yields an implicitly-computed



(a) $h_{\min} = 1/32$ (b) $h_{\min} = 1/64$

Figure 10: Error for the x-component of the velocity field for two different meshes, $h_{\min} = 1/32$ and $h_{\min} = 1/64$, illustrating how it behaves on refining.

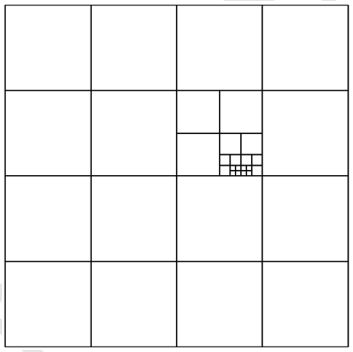


Figure 11: Non-graded mesh used in the projection stability test case.

velocity field $\mathbf{u}^*$ that is not divergence free, by the solution of

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\delta t} = -\nabla p^n - \mathbf{u}^n \cdot \nabla \mathbf{u}^n + \frac{1}{Re}\nabla^2 \mathbf{u}^* + \mathbf{f}^n \tag{30}$$

with $\mathbf{u}^*$ satisfying the same boundary conditions as $\mathbf{u}$. By applying the previously discussed Helmholtz-Hodge decomposition to $\mathbf{u}^*$ and defining the scalar $\varphi = -\delta t(p^{n+1} - p^n)$, the corrected velocity field can be computed from the

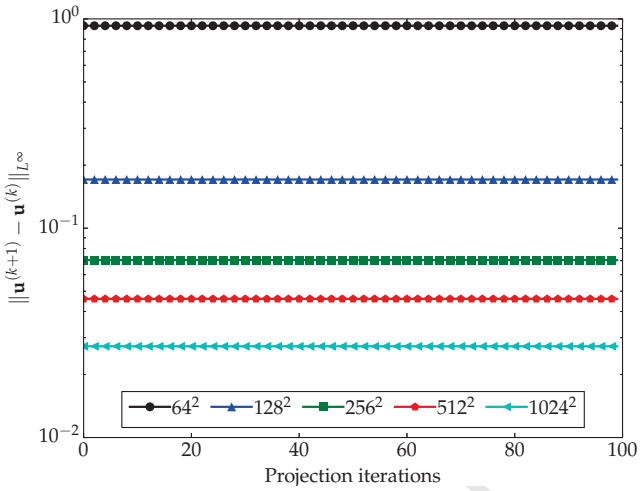


Figure 12: Log of the error $\|\mathbf{u}^{(k+1)} - \mathbf{u}^{(k)}\|_{L_\infty}$ after successive projections of the initial vector field.

decomposition itself (projection step) as

$$\mathbf{u}^{n+1} = \mathbf{u}^* - \nabla\varphi \tag{31}$$

where φ is obtained by solving the Poisson problem

$$\nabla^2\varphi = \nabla \cdot \mathbf{u}^*, \tag{32}$$

obtained by computing the divergence of equation (31), with homogeneous Neumann boundary conditions $\check{\mathbf{n}} \cdot \nabla\varphi = 0$ on $\partial\Omega$. This procedure is computed sequentially (Eqs. (30) $\rightarrow$ (32) $\rightarrow$ (31)), therefore one time step of this method is much faster than computing one time step of the coupled problem, explaining its popularity. The incremental version of the projection method has a splitting error (unrelated to the time discretization) of the order of $\mathcal{O}(\delta t^2)$, allowing the choice of 2nd-order accurate methods for the time discretization.

The system in Eq. (30) is discretized in time by a semi-implicit scheme, where viscous part is implicit, but the convective part is kept at the previ-

ous time-step to linearize the system. As a consequence, the time step δt has to satisfy the well known Courant-Friedrichs-Lewy (CFL) condition, which is $\delta t \|\mathbf{u}\|_\infty / h_{\min} \leq 1$.

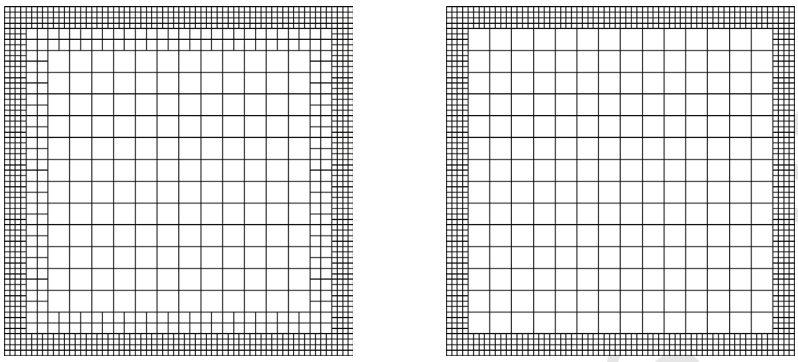
Remark 5. The non-incremental version of this projection method can be achieved by setting $\nabla p^n = \mathbf{0}$ in the momentum equation. This is sometimes referred as Chorin's method [34], and has a $\mathcal{O}(\delta t)$ splitting error, hindering the convergence of higher order time discretizations.

Spatial discretizations are performed for each step of the projection method described above. A staggered arrangement of the unknowns is employed, as explained in previous sections, avoiding known tensile instabilities. All discretizations are based on 2nd order accurate finite difference stencils, that are used throughout the hierarchical mesh. As already addressed, the interface between different mesh sizes is handled by the newly developed MLS interpolation technique.

In this work, for simplicity, we restrict ourselves to the case of low-Reynolds numbers flows, therefore the non-linear (convective) terms of the Navier-Stokes equations can be safely discretized by central differencing with MLS interpolations for off-grid point assessments.

4. Numerical results

In this section we present numerical results for verification tests by solving the Navier-Stokes equations with manufactured solutions, both in 2D and 3D. We display the convergence curves for these academic problems comparing to other results found in the literature. All simulations were performed with the meshless interpolation scheme described in Section 2, coupled with the projection method and time discretization both described in Section 3.3. The resulting discrete linear systems are solved with the GMRES method, as



(a) Graded case (b) Non-graded case

Figure 13: Meshes used in the 2D lid-driven cavity flow convergence test. Both correspond to $h_{\min} = 1/64$ and $h_{\max} = 1/16$

implemented in PETSc library [35, 36]. The last example in this section is a full three-dimensional simulation of a complex array of microchannels, using a locally-refined octree mesh, computed by our in-house parallel code that implements the techniques described in this work, illustrating the robustness and applicability of our methodology.

4.1. Lid-driven cavity flow in 2D

A simple 2D lid-driven cavity flow is simulated with the proposed methodology. To assess the convergence rates for this case, we employed a test with manufactured analytical steady-state solution, published by Shih & Tan [37]. As can be seen in [37], the manufactured solution for velocity and pressure consists of products of polynomials in x and y directions. The computational domain is set to be $\Omega = [0, 1]^2$, with $Re = 100$. We computed solutions for both graded and non-graded hierarchical meshes, starting with $h_{\min} = 1/64$ and $h_{\max} = 1/16$, and the subsequent meshes are produced by divisions by two in each direction. The starting meshes used in this test can be seen in Fig. 13.

The simulations were performed solving the transient Navier-Stokes equations for pseudo time-step marching until the steady-state solution is reached,

before computing the error norms. Figure 14 depicts the convergence of the L_2-norm error for both velocity (left) and pressure (right), for both graded and non-graded meshes. Results are consistent with previous convergence tests, yielding clean 2nd order spatial convergence. Despite the fact that non-graded mesh has significantly less elements with sharper transitions between coarse and fine cells, the order of convergence is maintained, with slightly larger errors as expected, but still comparable to the graded case. This test was performed using 2nd order polynomial for the MLS interpolations.

4.2. Unsteady flow with analytical solution in 3D

We test the order of convergence for three-dimensional simulations in non-graded grids by reproducing the benchmark test found in [4], section 4.4. The domain is a cube $[-\frac{\pi}{2}, \frac{\pi}{2}]^3$ with $Re = 1$, for which the analytical solution can be found in [4]. We used the same grid as in their work, which is a non-graded octree, shown in Fig. 6 for the mesh size $h_{\min} = \pi/32$. The time step is chosen to be $\delta t = h_{\min}/4$, satisfying the CFL restriction. This value is however smaller than the time step in [4], since they use a semi-Lagrangian approach for the advective term, which is free from CFL-type restriction. As in their work we computed the numerical solutions up to a time $t = \pi$. Meshless interpolations are computed with 2nd degree polynomial basis, such as in the two-dimensional tests.

Results can be seen in Fig. 15, where the L_2-norm of the error for the x-component of the velocity field is reported. The result computed with our method is compared to the results found in Min & Gibou [4], with close agreement between them. A slight advantage can be noticed for Min & Gibou's geometrical interpolation method for this case. Conversely, our method displays a clean 2nd order convergence rate for all computed meshes. We also report pressure errors in Fig. 15 (right), but unfortunately, we could not find

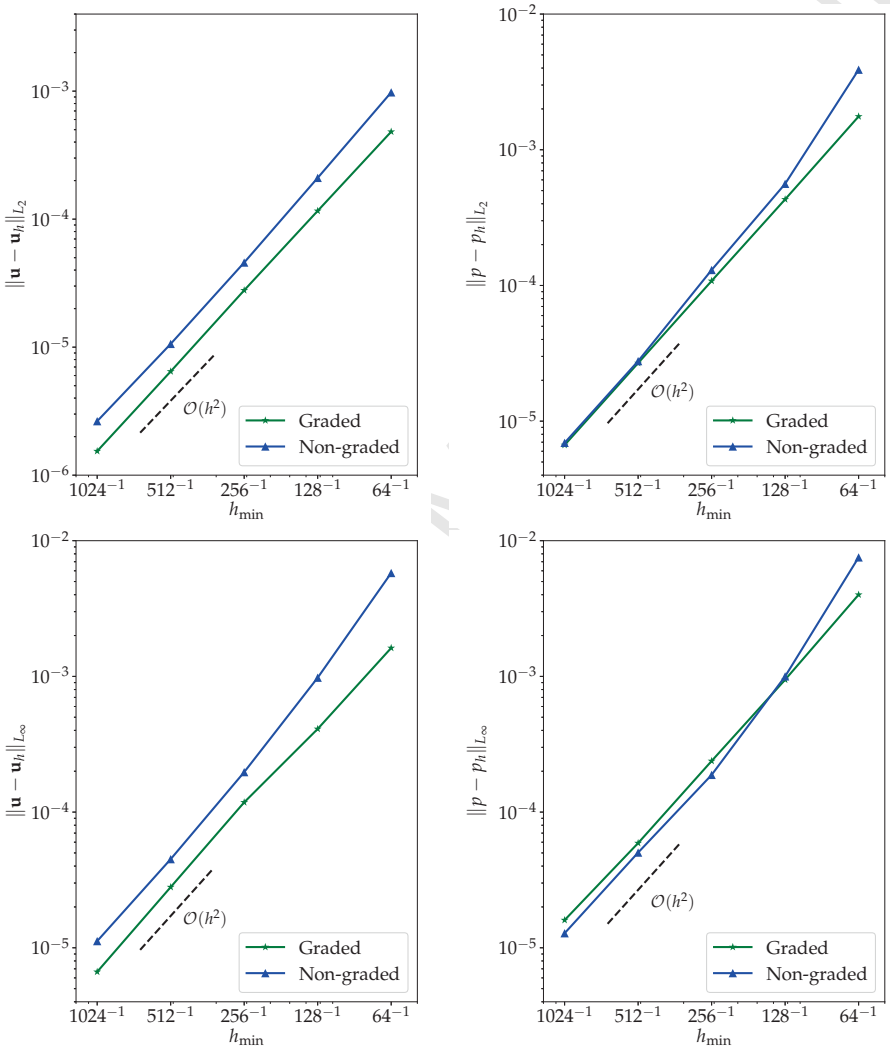


Figure 14: Convergence in the L_2-norm (top) and L_∞-norm (bottom) for both velocity (left) and pressure (right) errors, comparing graded and non-graded cases.

pressure errors in [4] for this test case to compare with our results. They do report however a convergence of $\mathcal{O}(h^2)$ for the divergence of the velocity field, that is not comparable to our method, since the discrete divergence of the solution is found to be constant circa 10^{-12}, independent of mesh size. This same behavior is observed in all Navier-Stokes simulations performed with our method.

4.3. Simulation of a complex 3D array of microchannels

In this test case, we simulate an incompressible single-fluid flow in an array of microchannels, introducing some level of geometric complexity in the three-dimensional flow domain. This test was designed specifically to demonstrate the simulation capabilities and robustness of the developed approximation methodology, as well as the flexibility of the parallel in-house code implemented for this purpose, with aid of the PETSc library [35, 36].

The geometry, as well as boundary conditions, can be seen in Fig. 16. The total width, length and height are set to be respectively $W = 0.8\ mm$, $L = 2.4\ mm$ and $H = 0.4\ mm$. The inlet is a channel of $0.1\ mm \times 0.1\ mm$, where water at room temperature is injected with constant velocity of $U_{\text{in}} = 0.1\ mm/s$. Scaling this geometry by $\ell = 0.1\ mm$, and using $\nu \approx 10^{-6}\ m^2/s$ as the kinematic viscosity of water at room temperature, we end up with a Reynolds number of $Re = \ell\ U_{\text{in}}/\nu = 10^{-2}$.

This array of microchannels is discretized by the union of several hierarchical grids, with $h_{\min} = \ell/40 = 2.5 \times 10^{-3}\ mm$ and $h_{\max} = \ell/10 = 10^{-2}\ mm$, resulting a total of 1,517,704 elements, and approximately 6 million variables. Figure 17 shows slices of the 3D mesh to give an idea of the level of refinement used in each section of the mesh. The simulation was performed using $\delta t = 10^{-3}\ s$ and the approximate solutions are computed until steady state is reached. Streamlines for this flow can be seen in Fig. 18. The result is qualitative, but demonstrates the robustness and applicability of this newly developed

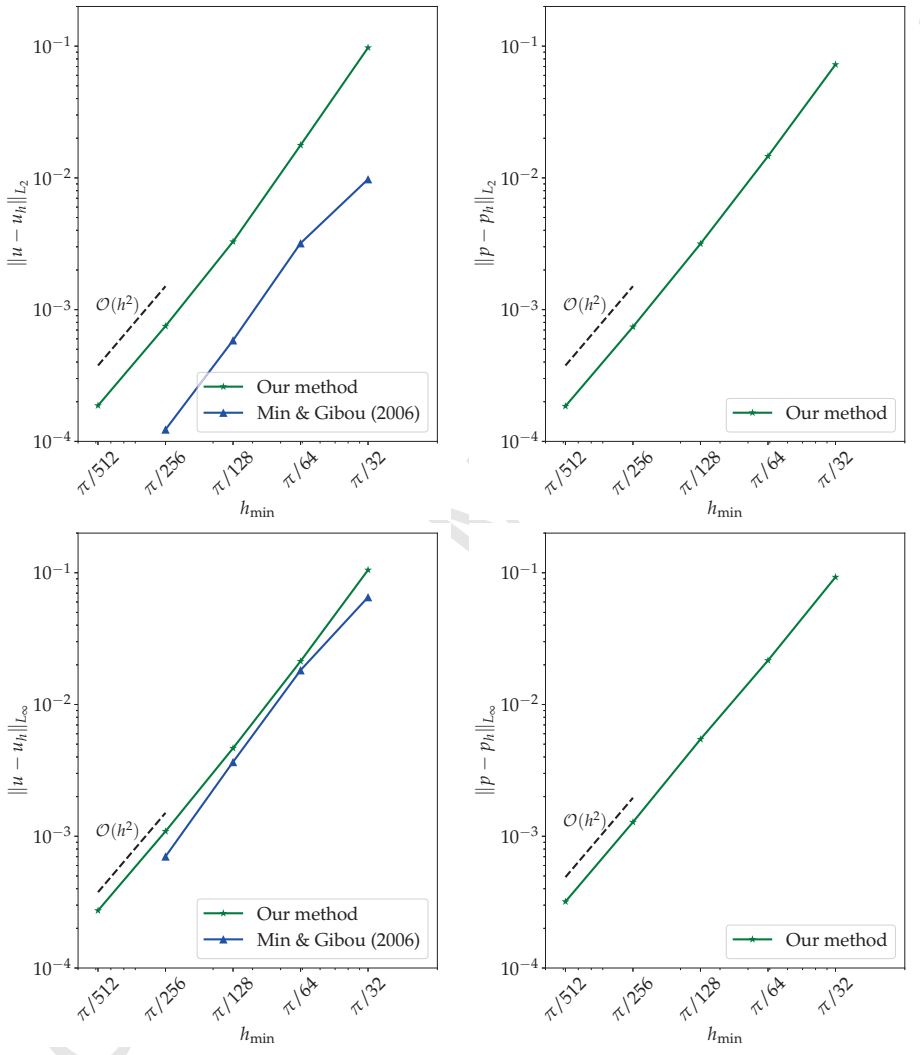


Figure 15: Convergence in the L_2-norm (top) and L_∞-norm (bottom) of the x-component of the velocity field (left) and pressure (right). Only the velocity error is compared to the results reported by Min et al. [4], since there is no information about pressure convergence in their work.

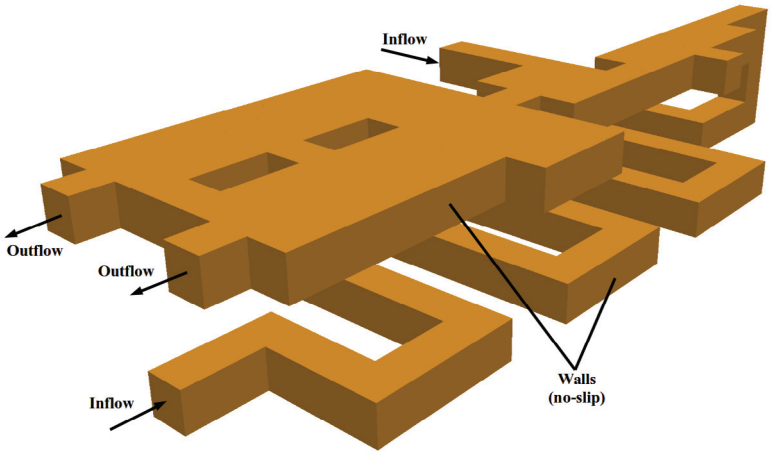
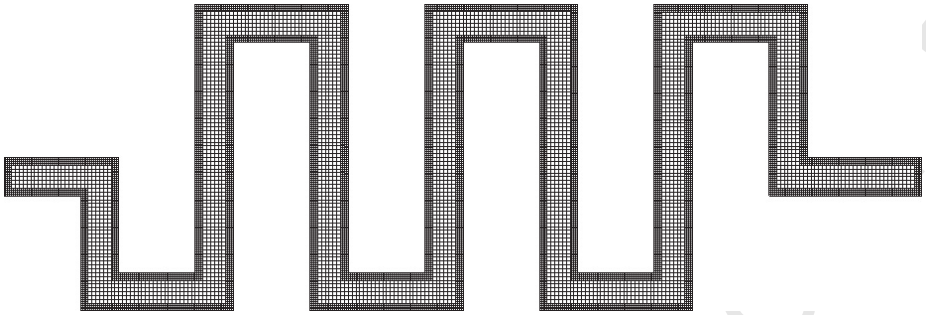


Figure 16: Microchannel problem configuration (Length $L = 2.4$, width $W = 0.8$ and height $H = 0.4$).

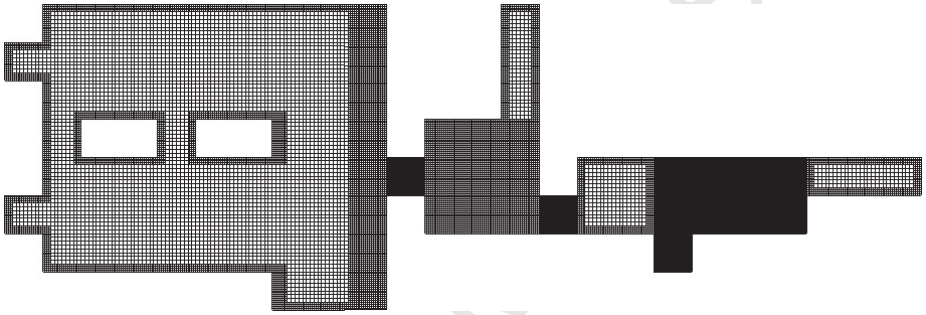
methodology.

5. Conclusions

A finite difference method with meshless interpolations in tree-based grids is presented. The method is capable of handling generic hierarchical tree-based grids as well as grids represented by both graded and non-graded quadtrees/octrees. The interpolations are based on Moving Least Squares approximations performed to compute the weights of the final finite difference stencil, and incorporated in the linear system matrix. We performed several convergence tests for a simple Poisson equation in graded and non-graded grids, with 2nd order convergence for both the solution and its gradient. Comparisons with other already published methods provide evidence of the superiority and flexibility of our technique. We further extended this methodology to compute the solution of the incompressible Navier-Stokes equations. After testing the robustness of our methodology with several academic tests involving the Helmholtz-Hodge decomposition, we presented convergence tests for incompressible flows with manufac-



(a) Plane $z = 0.05\ mm$



(b) Plane $z = 0.35\ mm$

Figure 17: Slices showing details of the octree mesh used to discretize the 3D array of microchannels.

tured solutions in both 2D and 3D, computed in graded and non-graded grids. The results are consistent with previous tests, yielding 2nd order convergence for the primitive variables. Finally, the incompressible flow of a single fluid in a three-dimensional array of microchannels is simulated, introducing a level of complexity to the geometry of the domain, together with a locally-refined grid. The developed method handles this simulation seamlessly, yielding reasonable qualitative results. Future work will focus on the extension of this methodology for non-Newtonian fluid models, as well as multi-fluid flows.

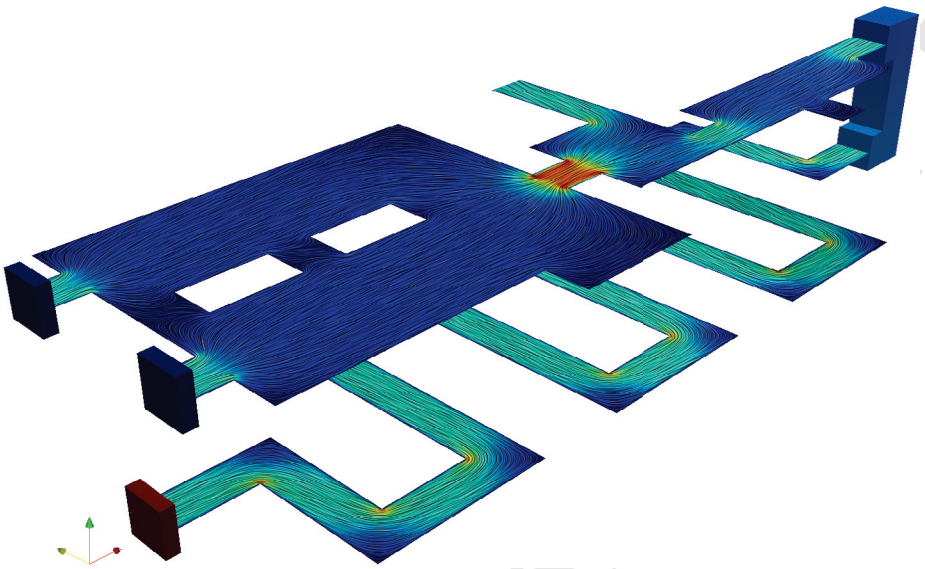


Figure 18: Streamlines for the complex 3D array of channels, taken from two slices, $z = 0.05\ mm$ and $z = 0.35\ mm$. The color scale varies from smallest (blue) to largest (red) velocity magnitude.

Acknowledgments

All authors thank the financial support from Brazilian Petroleum Agency (ANP)/Petrobras grant 0050.0075367.12.9, and from the São Paulo Research Foundation (FAPESP) grant 2013/07375-0. J.L. Ansoni and A. Castelo also thanks the financial support from the National Council for Scientific and Technological Development (CNPq) grants 152397/2016-7 and 307483/2017-7. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) Finance Code 001.

References

[1] S. Popinet, "Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries," *Journal of Computational Physics*, vol. 190, pp. 572–600, Sept. 2003.

[2] F. Losasso, F. Gibou, and R. Fedkiw, "Simulating water and smoke with an octree data structure," *ACM Trans. Graph.*, vol. 23, pp. 457–462, Aug. 2004.

[3] F. Losasso, R. Fedkiw, and S. Osher, "Spatially adaptive techniques for level set methods and incompressible flow," *Computers & Fluids*, vol. 35, pp. 995–1010, dec 2006.

[4] C. Min and F. Gibou, "A second order accurate projection method for the incompressible Navier-Stokes equations on non-graded adaptive grids," *Journal of Computational Physics*, vol. 219, pp. 912–929, 2006.

[5] A. Guittet, M. Theillard, and F. Gibou, "A stable projection method for the incompressible Navier-Stokes equations on arbitrary geometries and adaptive quad/octrees," *Journal of Computational Physics*, vol. 292, pp. 215–238, July 2015.

[6] F. Gibou, R. Fedkiw, and S. Osher, "A review of level-set methods and some recent applications," *Journal of Computational Physics*, vol. 353, pp. 82–109, jan 2018.

[7] P. Gómez, C. Zanzi, J. López, and J. Hernández, "Simulation of high density ratio interfacial flows on cell vertex/edge-based staggered octree grids with second-order discretization at irregular nodes," *Journal of Computational Physics*, vol. 376, pp. 478–507, jan 2019.

[8] M. J. Berger and J. Oliger, "Adaptive mesh refinement for hyperbolic partial differential equations," *Journal of Computational Physics*, vol. 53, no. 3, pp. 484–512, 1984.

[9] L. H. Howell and J. B. Bell, "An adaptive-mesh projection method for

viscous incompressible flow," *SIAM J. SCI. COMPUT*, vol. 18, pp. 18–996, 1996.

[10] M. R. Pivello, M. M. Villar, R. Serfaty, A. M. Roma, and A. Silveira-Neto, "A fully adaptive front tracking method for the simulation of two phase flows," *International Journal of Multiphase Flows*, vol. 58, pp. 72–82, 2014.

[11] A. M. Khokhlov, "Fully threaded tree algorithms for adaptive refinement fluid dynamics simulations," *Journal of Computational Physics*, vol. 143, no. 2, pp. 519–543, 1998.

[12] C. Batty, "A cell-centred finite volume method for the Poisson problem on non-graded quadtrees with second order accurate gradients," *Journal of Computational Physics*, vol. 331, pp. 49–72, Feb. 2017.

[13] M. A. Olshanskii, K. M. Terekhov, and Y. V. Vassilevski, "An octree-based solver for the incompressible Navier-Stokes equations with enhanced stability and low dissipation," *Computers & Fluids*, vol. 84, pp. 231 – 246, 2013.

[14] C. Min, F. Gibou, and H. D. Ceniceros, "A supra-convergent finite difference scheme for the variable coefficient Poisson equation on non-graded grids," *Journal of Computational Physics*, vol. 218, pp. 123–140, Oct. 2006.

[15] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, and P. Krysl, "Meshless methods: An overview and recent developments," *Computer Methods in Applied Mechanics and Engineering*, vol. 139, no. 1, pp. 3–47, 1996.

[16] J. J. Benito, F. Ureña, and L. Gavete, "Influence of several factors in the generalized finite difference method," *Applied Mathematical Modelling*, vol. 25, pp. 1039–1053, Dec. 2001.

[17] S. Li and W. K. Liu, "Meshfree and particle methods and their applications," *Appl Mech Rev*, vol. 55, no. 1, pp. 1–34, 2002.

[18] X. K. Zhang, K.-C. Kwon, and S.-K. Youn, "The least-squares meshfree method for the steady incompressible viscous flow," *Journal of Computational Physics*, vol. 206, pp. 182–207, June 2005.

[19] V. P. Nguyen, T. Rabczuk, S. Bordas, and M. Duflot, "Meshless methods: A review and computer implementation aspects," *Mathematics and Computers in Simulation*, vol. 79, pp. 763–813, Dec. 2008.

[20] B. Seibold, "Minimal positive stencils in meshfree finite difference methods for the Poisson equation," *Computer Methods in Applied Mechanics and Engineering*, vol. 198, pp. 592–601, Dec. 2008.

[21] X. Zheng, W.-y. Duan, and Q.-W. Ma, "Comparison of improved meshless interpolation schemes for SPH method and accuracy analysis," *Journal of Marine Science and Application*, vol. 9, no. 3, pp. 223–230, 2010.

[22] H. Ding, C. Shu, K. S. Yeo, and D. Xu, "Simulation of incompressible viscous flows past a circular cylinder by hybrid FD scheme and meshless least square-based finite difference method," *Computer Methods in Applied Mechanics and Engineering*, vol. 193, pp. 727–744, Mar. 2004.

[23] H. Ding, C. Shu, and Q. D. Cai, "Applications of stencil-adaptive finite difference method to incompressible viscous flows with curved boundary," *Computers & Fluids*, vol. 36, pp. 786–793, May 2007.

[24] C. S. Chew, K. S. Yeo, and C. Shu, "A generalized finite-difference (GFD) ALE scheme for incompressible flows around moving solid bodies on hybrid meshfree-cartesian grids," *Journal of Computational Physics*, vol. 218, pp. 510–548, Nov. 2006.

[25] X. Y. Wang, K. S. Yeo, C. S. Chew, and B. C. Khoo, "A SVD-GFD scheme for computing 3D incompressible viscous fluid flows," *Computers & Fluids*, vol. 37, pp. 733–746, July 2008.

[26] A. Quarteroni, R. Sacco, and F. Saleri, *Numerical Mathematics*. Springer, 2007.

[27] J. B. Perot, "An analysis of the fractional step method," *Journal of Computational Physics*, vol. 108, pp. 51–58, 1993.

[28] J. L. Guermond and L. Quartapelle, "On stability and convergence of projection methods based on pressure Poisson equation," *International Journal for Numerical Methods in Fluids*, 1998.

[29] J. L. Guermond, P. Minev, and J. Shen, "An overview of projection methods for incompressible flows," *Computer Methods in Applied Mechanics and Engineering*, vol. 195, no. 44-47, pp. 6011–6045, 2006.

[30] P. Gervasio and F. Saleri, "Algebraic fractional-step schemes for time-dependent incompressible Navier-Stokes equations," *J. Sci. Comput.*, vol. 27, pp. 257–269, 2006.

[31] F. S. Sousa, C. M. Oishi, and G. C. Buscaglia, "Spurious transients of projection methods in microflow simulations," *Computer Methods in Applied Mechanics and Engineering*, vol. 285, pp. 659–693, 2015.

[32] R. Codina, "Pressure stability in fractional step finite element methods for incompressible flows," *Journal of Computational Physics*, vol. 170, no. 1, pp. 112–140, 2001.

[33] A. Quarteroni, F. Saleri, and A. Veneziani, "Factorization methods for the numerical approximation of Navier-Stokes equations," *Computer methods in applied mechanics and engineering*, vol. 188, no. 1-3, pp. 505–526, 2000.

[34] A. J. Chorin, "Numerical solution of the Navier-Stokes equations," *Mathematics of Computation*, vol. 22, pp. 745–762, 1968.

[35] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang, "PETSc users manual," Tech. Rep. ANL-95/11 - Revision 3.7, Argonne National Laboratory, 2016.

[36] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang, "PETSc Web page," 2016.

[37] T. M. Shih, C. H. Tan, and B. C. Hwang, "Effects of grid staggering on numerical schemes," *International Journal for Numerical Methods in Fluids*, vol. 9, no. 2, pp. 193–212, 1989.