Contents lists available at ScienceDirect

Journal of Computational Science

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

Fast convergence of SPH numerical solutions using robust algebraic multilevel

L.P. da Silva [a,*], C.H. Marchi [a], M. Meneguette [b], R. Suero [c]

[a] Laboratory of Numerical Experimentation (LENA), Department of Mechanical Engineering (DEMEC), Federal University of Paraná (UFPR), P.O. Box 19040, Zip Code 81531-980, Curitiba, PR, Brazil
[b] São Paulo State University (UNESP), Department of Mathematics and Computer Science (DMC), Zip Code 19060-900, Presidente Prudente, SP, Brazil
[c] Federal Institute of Paraná (IFPR) – Campus of Paranaguá, Zip Code 83215-750, Paranaguá, PR, Brazil

ARTICLE INFO

Keywords:
Robust algebraic multilevel
SPH
Uniform and non-uniform particle discretization
Heat diffusion
Parallel Gauss–Seidel–Silva CSR

ABSTRACT

In our study we solve 2D equations that model the mathematical phenomenon of steady state heat diffusion. The discretization of the equations is performed with the smoothed particle hydrodynamics (SPH) method and the resolution of the associated system of linear equations is determined with a modified solver that we call the Gauss–Seidel–Silva (G–S–S). The single level parallel G–S–S solver is compared to the algebraic multilevel (AML) with serial G–S–S smoother which has the ability to smooth the error of the numerical solutions and accelerate convergence due to its iterative formulation. The AML with serial G–S–S smoother is responsible for determining speed-ups of 4084 times for uniform and 5136 times for non-uniform particle discretization. We estimate a speed-up of 41082 times for the AML with parallel G–S–S smoother.

1. Introduction

Computational heat transfer (CHT) [1] is an area that concentrates computational techniques with the objective of determining numerical solutions for independent variables of interest, such as the temperature at a given mesh node position, for example. To determine the numerical solution of a variable of interest, it is necessary to know the mathematical model, to choose a numerical scheme capable of discretizing the model, and to solve the associated system of linear or nonlinear equations. Also, the accuracy of the solution depends only on the numerical scheme applied; the efficiency, however, is dictated by the numerical scheme and also by the algorithm used to solve the system of equations.

Among CHT problems, heat diffusion in the steady state is a frequently studied mathematical model, because the mathematical equation representing this phenomenon is contained in the Navier–Stokes equations when the projection method is applied. Solving this equation using classical methods such as: finite difference method [2–5], finite volume method (FDM) [6,7], and finite element method (FEM) [8,9] is known from the literature. There are also high order methods, for example those based on FDM approximations, such as compact finite difference scheme (*Compact-4*) [10], and exponential finite difference scheme (*Exponential-4*) [10].

Geometric multigrid (GMG) is a method that can be applied to accelerate the convergence of numerical solutions for cases where the computational grid is structured. Its efficiency is highly appreciated, but its implementation is specific for each geometric structure, mathematical model, and boundary condition [11–15].

A detailed study on optimal parameters for the geometric multigrid was performed by [16]. The authors identified appropriate values for the number of levels of the multigrid when solving equations such as Laplace's, Burger's, and Navier–Stokes with MSI and SOR solver. In the cases shown, they identified that the CPU time increased for a number of levels other than the optimal one. Also, the excellent performance of the geometric multigrid method over the singlegrid (SG), obtained for the coupled problems, was not maintained when using the Navier–Stokes equation written with the streamfunction–vorticity formulation.

In [12] it was used local Fourier analysis to show the robustness of the multigrid method with ILU smoother on anisotropic problems on triangular meshes. Excellent results were obtained for oblique flow problems and problems on semi-structured triangular grids. In other studies [17,18], the authors used local Fourier analysis to show the efficiency and robustness of the geometric multigrid method in physical

* Corresponding author.
E-mail addresses: luciano.silva@ufpr.br (L.P. da Silva), chmcfd@gmail.com (C.H. Marchi), messias.meneguette@unesp.br (M. Meneguette), roberta.suero@ifpr.edu.br (R. Suero).
URLs: http://lattes.cnpq.br/1873355458944068 (L.P. da Silva), http://www.cfd.ufpr.br/ (C.H. Marchi), http://lattes.cnpq.br/1531018187057108 (M. Meneguette), http://lattes.cnpq.br/6702579185974800 (R. Suero).

https://doi.org/10.1016/j.jocs.2024.102369
Received 2 March 2023; Received in revised form 5 February 2024; Accepted 18 June 2024
Available online 27 June 2024

anisotropy/orthotropy problems. They used different restriction and prolongation operators and x–y-zebra Gauss–Seidel [17] and ILU [18] smoother with different orderings. Excellent results were obtained for problems with strong diffusive orthotropy.

In the study [19], the authors solved the Navier–Stokes equations using a projection method with an incremental correction scheme on pressure and the conjugate gradient method preconditioned with the geometric multigrid method and ILU solver to solve the velocity and pressure equations. Such a methodology considerably accelerated the iterative process compared to the classical methods available in the literature. Furthermore, the acceleration between the solution obtained with the Gauss–Seidel method without geometric multigrid and for the ILU method with geometric multigrid was determined using the speed-up equal to 105 times for a grid of 128×128.

In [10] the authors showed how to solve the 2D steady state heat diffusion equation using two fourth order of accuracy methods. In addition, using repeated Richardson extrapolation (RRE) [20–26], they determined tenth order of accuracy for the numerical solutions that were initially fourth order. To determine the numerical solutions, they applied the geometric multigrid method which accelerated the convergence of the solutions, obtaining the solution for the most refined grid shown, 4097×4097, in *4.75E+02* seconds for *Compact-4*, and *4.12E+02* seconds for *Exponential-4*.

In [27] it was applied the multigrid method to the 1D and 2D wave equation, with discretization using the weighted finite difference method and time-stepping. The multigrid method proved to be very advantageous for this class of problems, as it showed good convergence and robustness factors.

In the recent study [28], the authors introduced a generalized multigrid method as an extension of the monotone multigrid method, which can handle linear inequality restrictions. They showed the robustness and efficiency of the multigrid method to solve Signorini's problem and two-body contact problems. The generalized multigrid method introduced in the paper can be used to solve quadratic constraint minimization problems where the number of constraints is significantly smaller than the number of unknowns. In addition, the authors point out that the application of the method to models that consider hyperelastic material can be quite attractive.

The main difficulties in applying the geometric multigrid method are that it is the only one for each problem and requires the use of structured grids. To get around this limitation, they developed another type of multigrid, which they called the algebraic multigrid (AMG) [29, 30]. Over the years, a review study has pointed out that no algebraic method is able to solve all problems indiscriminately [31]. However, the fact that the algebraic multigrid uses the information directly from the coefficient matrix associated with the system of linear or nonlinear equations makes the method robust.

The study performed by [32] points out optimal values for some parameters used in AMG when solving the equation that models the phenomenon of heat diffusion in the steady state; the highest efficiency of AMG is obtained when considering the maximum number of levels. In all cases, AMG was slightly slower than GMG, however, when compared to SG, both AMG and GMG showed high efficiency to accelerate the convergence of the numerical solutions.

It is very important to understand that AMG solves all the problems that GMG also solves, but the reciprocal is not true. GMG is a method that only solves problems involving structured grids. In this way, it becomes a particular case of the algebraic multigrid method [33]. AMG is based directly on the characteristics of the matrix of coefficients, boundary conditions, discontinuities, and geometric anisotropies. Constructing the matrix at the levels with the fewest elements is a purely algebraic problem, making it adaptable to cases of great geometric complexity.

Thinking about the robustness of AMG, which is suitable for solving problems on unstructured grids [31], and the discretization characteristic with the SPH method [34,35], which generally considers disordered particles, we decided to investigate the behavior of AMG to accelerate the convergence of numerical solutions also in methods that do not use computational meshes, such as SPH, for example.

In SPH the domain is discretized by a set of particles that replace the computational mesh. To adapt AMG to SPH, we replace the term grid by level, since the particles form discretization levels and not grids. Furthermore, we will call the algebraic multigrid by algebraic multilevel (AML) without causing any loss of authorship.

We understand that the study involving the application of AML to determine the numerical solutions obtained from the discretization of the mathematical models using the SPH, can contribute to the state-of-the-art in the sense that implicit and stable formulations can be adopted to solve flow models, where the time variable is very relevant.

2. Mathematical and numerical models

For our study, we chose the 2D mathematical model of steady state heat diffusion equation. We selected two distinct cases for the analyses of the results. One is n-derivable and the other has a finite number of derivatives. Table 1 presents all the symbols shown in the study along with their respective descriptions.

2.1. Mathematical model

The 2D model of steady state heat diffusion applied to particle $\mathbf{x} = \mathbf{x}(x, y)$ is

$$\nabla^2 \boldsymbol{\Psi}(\mathbf{x}) = f(\mathbf{x}), \quad 0 \le x \le 1, \quad 0 \le y \le 1, \tag{1}$$

with $\boldsymbol{\Psi}(0, y)$, $\boldsymbol{\Psi}(1, y)$, $\boldsymbol{\Psi}(x, 0)$, and $\boldsymbol{\Psi}(x, 1)$ being Dirichlet boundary conditions.

Case I of a 2D steady state heat diffusion model (C1-2Ds): The source term is polynomial.

$$\nabla^2 \boldsymbol{\Psi}(\mathbf{x}) = -2[(1-6x^2)y^2(1-y^2) + (1-6y^2)x^2(1-x^2)], \tag{2}$$

where $\boldsymbol{\Psi}(\mathbf{x}) = (x^2 - x^4)(y^4 - y^2)$ is analytical solution, and $\boldsymbol{\Psi}(x, 0) = \boldsymbol{\Psi}(x, 1) = \boldsymbol{\Psi}(0, y) = \boldsymbol{\Psi}(1, y) = 0$ being boundary conditions.

Case II of a 2D steady state heat diffusion model (C2-2Ds): The source term is trigonometric.

$$\nabla^2 \boldsymbol{\Psi}(\mathbf{x}) = \sin(\pi x)\sin(\pi y), \tag{3}$$

where $\boldsymbol{\Psi}(\mathbf{x}) = -1/(2\pi^2)\sin(\pi x)\sin(\pi y)$ is analytical solution, and $\boldsymbol{\Psi}(x, 0) = \boldsymbol{\Psi}(x, 1) = \boldsymbol{\Psi}(0, y) = \boldsymbol{\Psi}(1, y) = 0$ being boundary conditions.

2.2. Numerical model

In this section we present the discrete form of the two-dimensional steady state heat diffusion equation using the SPH method.

2.2.1. Discretization of the 2D steady state heat diffusion model

We consider in our study that the discretization error is generated exclusively by the truncation error, this means that other sources of numerical error are negligible, according to the equation

$$\langle \nabla^2 \boldsymbol{\Psi}(\mathbf{x}_i) \rangle = \boldsymbol{\psi}(\mathbf{x}_i) + \varepsilon_\tau(\boldsymbol{\psi}(\mathbf{x}_i)). \tag{4}$$

Eq. (5) describes the SPH approximation for the second-order derivative [36] and the associated truncation error.

$$\langle \nabla^2 \boldsymbol{\Psi}(\mathbf{x}_i) \rangle = 2 \sum_{j \in V_i} \frac{(\boldsymbol{\psi}(\mathbf{x}_i) - \boldsymbol{\psi}(\mathbf{x}_j))}{r_{ij}^2} \frac{m_j}{\rho_j} \mathbf{x}_{ij} . \nabla_i W_{ij} + O(\mathbf{x}_i - \mathbf{x}_j)^2, \tag{5}$$

$$\nabla^2 \boldsymbol{\Psi}(\mathbf{x}_i) \approx 2 \sum_{j \in V_i} \frac{(\boldsymbol{\psi}(\mathbf{x}_i) - \boldsymbol{\psi}(\mathbf{x}_j))}{r_{ij}^2} \frac{m_j}{\rho_j} \mathbf{x}_{ij} . \nabla_i W_{ij} \tag{6}$$

and

$$\nabla_i W(\mathbf{x}_{ij}, h) = \frac{\alpha_{w,d} \mathbf{x}_{ij}}{h^{d+1} r_{ij}} \frac{\partial W_{ij}}{\partial \phi}, \tag{7}$$

Table 1
Symbol description.

Symbol	Description
$A, A_1, A_2, \ldots$	Coefficient matrix
A^h	Coefficient matrix associated to finer level
A^H	Coefficient matrix associated to coarser level
AA	Row or column vector with all the elements of matrix A
AD	Row or column vector with elements of the principal diagonal of A
d	Dimension
DS	Degree of sparsity
$f, f_1, f_2, \ldots$	Source term
f^h	Source term associated to finer level
f^H	Source term associated to coarser level
f_p	Portion of the algorithm that can be improved (parallelized fraction)
g	Smallest distance between two particles
G–S–S	Gauss–Seidel–Silva solver and smoother
h	Smoothing length ($h = \kappa \Delta x = \kappa \Delta y$), where $\kappa = 1$
IA	Row or column vector with start and end pointer to the non-null elements in each row of A
JA	Row or column vector with all the numbers of the non-null columns of A
k	Scaling factor
kh	Radius of the circumference of the domain of influence of the kernel function
m	Mass
n_{aa}	Dimension of vectors AA and JA
n_{ia}	Dimension of vector IA
N_{p_x}	Numbers of uniform mean particle spacing in x direction
N_{p_y}	Numbers of uniform mean particle spacing in y direction
N_{main}	Number of main discretizations
N_L	Number of AML sublevels
N_{Lmax}	Maximum number of AML sublevels ($\log2(N_{p_x})-1 = \log2(N_{p_y})-1$)
N_x	Number of inner particles in the x direction ($N_x = N_{p_x} - 1$)
N_y	Number of inner particles in the y direction ($N_y = N_{p_y} - 1$)
N_{ti}	Number of inner particles ($N_{ti} = N_x N_y$)
N_t	Total number of particles including the boundaries ($N_t = (N_x+1)(N_y+1)$)
q	Constant refinement ratio
Q	Particle coordinates
r	Distance between particles ($\|\|\mathbf{x}_{ij}\|\|$)
re	Constant coarsening ratio
S_p	Speed-up
S_τ	Speed-up
S_{Amdahl}	Speed-up
t_0, t_τ, t_{CPU}	CPU time
W	Kernel function
$\mathbf{x}_i$	Particle where the kernel function is centered
$\mathbf{x}_j$	Neighboring particle
$\mathbf{x}_{ij}$	Difference between particle coordinates $\mathbf{x}_i$ and $\mathbf{x}_j$ ($\mathbf{x}_i - \mathbf{x}_j$)
$\mathbf{x}_s$	Particle sensor fixed
x_i	Is the abscissa of the fixed particle
x_j	Is the abscissa of the neighboring particle
y_i	Is the ordinate of the fixed particle
y_j	Is the ordinate of the neighboring particle
$\alpha_{w,d}$	Normalization constants
Δx	Uniform mean particle distance ($\Delta x = 1/N_{p_x}$)
Δy	Uniform mean particle distance ($\Delta y = 1/N_{p_y}$)
κ	Abscissa shift factor ϕ
ϕ	Independent variable of the kernel function ($\phi = r/h$)
Y	Algorithm
$\boldsymbol{\Psi}$	Analytical solution
$\boldsymbol{\psi}$	Numerical solution
ρ	Density
∇W	Kernel gradient

where ρ_j and m_j are the density and mass of particle $\mathbf{x}_j$, respectively. Furthermore, $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$, $r_{ij} = \|\mathbf{x}_{ij}\|$, $\alpha_{w,d}$ is a normalization constant, and d is the domain dimension, according to [20]. The kernel gradient $\nabla_i W(\mathbf{x}_{ij}, h)$ is written based on the first derivative of the kernel $\partial W_{ij}/\partial\phi$ [37], and the kernel function used in the study was the quintic spline shown in [38].

From Eq. (5), we have the 2D discretization of the steady state heat diffusion model is given by

$$-\nabla^2 \boldsymbol{\Psi}(\mathbf{x}_i) = f(\mathbf{x}_i). \tag{8}$$

We know that $\boldsymbol{\Psi}$ represents the analytical solution that is unknown, so preserving mathematical rigor, we take care to replace such solution $\boldsymbol{\Psi}$ by $\boldsymbol{\psi}$, which will be the solution we will calculate (see the full deduction in [37,39]). Therefore,

$$-\nabla^2 \boldsymbol{\psi}(\mathbf{x}_i(x_i, y_i)) \approx f(\mathbf{x}_i(x_i, y_i)), \tag{9}$$

or using simplicity

$$-\nabla^2 \boldsymbol{\psi}(\mathbf{x}_i) = f(\mathbf{x}_i). \tag{10}$$

Thus, we have

$$-2 \sum_{j \in V_i} \frac{(\boldsymbol{\psi}(\mathbf{x}_i) - \boldsymbol{\psi}(\mathbf{x}_j))}{r_{ij}^2} \frac{m_j}{\rho_j} \mathbf{x}_{ij} . \nabla_i W_{ij} = f(\mathbf{x}_i). \tag{11}$$

Eq. (11), applied in our cases, determines sparse systems of linear equations with 25 diagonals, where the diagonal elements a_{ii} determined by the sum of all elements a_{ij} of row i for all $j \neq i$ with opposite sign [37].

3. Methodology

3.1. Compressed sparse row (CSR)

Compressed sparse row (CSR) method [40], allows a sparse matrix to be stored in three vectors: AA, JA, and IA. The vector AA stores all nonzero coefficients of the matrix A. Meanwhile, JA carries the number of the columns of each of the coefficients of A, and finally, the vector IA stores the start and end of each of the rows, as shown in Algorithm 1 [39]. The dimension of the vectors AA and JA, Eqs. (13) and (14), respectively, are the same and thus denoted by (n_{aa}). The vector IA has dimension $n_{ia} = n + 1$, where n is the order of the matrix A defined in Eq. (12), for example. The first element of IA is always equal to unity ($IA(1) = 1$) and the last element of IA is always the number of elements of the vector AA + 1 ($IA(n_{ia}) = n_{aa} + 1$). The vector AD, Eq. (16), is not part of the CSR method, but it is an important vector for the development of the Gauss–Seidel CSR smoother parallel programming. In AD, only the main diagonal elements of the matrix A are stored, and its dimension is $N_x N_y = N_{ti}$, for $N_x = N_y$.

$$A = \begin{bmatrix} 3 & 4 & 0 & 0 & 0 \\ 2 & 5 & 2 & 0 & 0 \\ 0 & 7 & 9 & 3 & 0 \\ 0 & 0 & 6 & 1 & 8 \\ 0 & 0 & 0 & 7 & 3 \end{bmatrix}, \tag{12}$$

$$AA = \begin{bmatrix} 3 & 4 & 2 & 5 & 2 & 7 & 9 & 3 & 6 & 1 & 8 & 7 & 3 \end{bmatrix}, \tag{13}$$

$$JA = \begin{bmatrix} 1 & 2 & 1 & 2 & 3 & 2 & 3 & 4 & 3 & 4 & 5 & 4 & 5 \end{bmatrix}, \tag{14}$$

$$IA = \begin{bmatrix} 1 & 3 & 6 & 9 & 12 & (n_{aa}+1) \end{bmatrix} = \begin{bmatrix} 1 & 3 & 6 & 9 & 12 & 14 \end{bmatrix}, \tag{15}$$

and

$$AD = \begin{bmatrix} 3 & 5 & 9 & 1 & 3 \end{bmatrix}. \tag{16}$$

3.2. Robust parallel Gauss–Seidel–Silva CSR smoother

The linear systems of algebraic equations associated with discretization of mathematical models using numerical methods are, in general, sparse. Degree of sparsity (DS), which represents the percentage of null elements in the matrix, increases as the domain is discretized with an increasing number of spacial step size or particles, for example. In this way, the CSR method contributes to memory savings by storing only

Algorithm 1: Compressed sparse row (CSR) with principal diagonal vector AD.

```
Input: $A$
Output: $AA, JA, IA, AD$
begin
    $IA(1) = 1$
    $k0 = 0$
    $k1 = 0$
    for $i = 1, \dots, n$ do
        for $j = 1, \dots, n$ do
            if $A(i,j) \neq 0$ then
                $k0 = k0 + 1$
                $AA(k0) = A(i,j)$
                $JA(k0) = j$
                if $i = j$ then
                    $k1 = k1 + 1$
                    $AD(k1) = A(i,j)$
                end
            end
            $IA(i+1) = k0 + 1$
        end
    end
end
```

the non-null coefficients of the matrix and, in addition, allows faster operations than in cases where the full sparse matrix is used. The gain in speed happens with or without parallelization, however, using the parallel solver with the CSR method, it will have a further reduction in processing time which in turn is attractive [40,41].

To solve the linear systems in our study, we adapted the original Gauss–Seidel to receive parallelization and also the CSR structure. Thus, we call this modified model Gauss–Seidel–Silva (G–S–S) (see Algorithm 2) [42]. This modification implies convergence of the numerical solution of the linear system in such a way that it is impossible to distinguish between Gauss–Seidel and Gauss–Jacobi, and may even be a combination of both methods. We kept the same structure for the serial Gauss–Seidel–S and will always differentiate between parallel and serial in our analyses. In addition, the Gauss–Seidel–S solver within the AML will be called a smoother due to its property of smoothing the oscillatory modes of the error.

In order to highlight the contributions made by G–S–S, we will point out a few observations: (i) the Gauss–Seidel solver well known in the literature is also known for not being parallelizable; (ii) the linear systems determined by SPH may vary the number of diagonals a lot. In our study we kept at most 25 diagonals; (iii) SCR structure only serves to compact the coefficients matrix; (iv) we adapted the CSR structure to capture the coefficients of the main diagonal, so that if we change the total number of diagonals, we can still solve the system, as this adaptation generated robustness; (v) we included the robust SCR structure within Gauss–Seidel.

Putting all this together, it is impossible within the parallel region to establish for the loop (Algorithm 2) which update returned first (upper or lower triangular matrix). Thus, the solution will converge regardless of which update returns first. This certainly differs from a modified Gauss–Seidel known in the literature. To reinforce our contribution, we present the scalability of Gauss–Seidel–S, as shown in Fig. 7, where we can confirm that our solver has a parallelized fraction of 90%.

To facilitate understanding about the gains in processing time when using the parallel Gauss–Seidel–S CSR smoother, it is necessary to know some performance metrics and the phenomenon known as “Amdahl’s law” [43].

To evaluate the performance of algorithms, the following metric was defined [44]:

Definition 1. Speed-up (S_p) is a metric used to measure the increase in speed of a Y_1 algorithm relative to Y_2.

$$S_p = \frac{t_{CPU}(Y_1)}{t_{CPU}(Y_2)}. \tag{17}$$

In terms of parallelization, S_p is used to measure the speed increase by comparing the CPU time of the Y algorithm with only one processor (Y-serial algorithm) and with number τ of processors (Y-parallel algorithm) [11]. The S_p shown in Eq. (17) considers that the two algorithms are running with the same number of processors. The metric for the case where number τ of processors are used can be rewritten as

$$S_\tau = \frac{t_0(Y)}{t_\tau(Y)}, \tag{18}$$

where S_τ is the speed-up that measures the speed increase when comparing the CPU time t_0 of the serial Y algorithm with one processor only and the CPU time t_τ of the Y algorithm with number τ of processors. Note that the time $t_0(Y)$ is calculated without parallelization and $t_1(Y)$ has a single parallel region. For S_τ to be calculated coherently, the $S_1 = t_0(Y)/t_1(Y)1 \approx 1$. The larger the value of S_τ, the shorter the CPU time. However, due to some serial calculation regions within the algorithm, the value of S_τ has a limiting factor, which is the CPU time of the serial S_τ algorithm itself. This limitation is known as “Amdahl’s law”. A more generalized form of Amdahl’s law [45] can be described by

$$S_{Amdahl} = \frac{1}{(1 - f_p) + \dfrac{f_p}{\tau}}, \tag{19}$$

where f_p (parallelized fraction) is the portion of the algorithm that can be improved by a τ factor due to the number of processors, and $(1 - f_p)$ is the portion that cannot be improved. In terms of parallelism, f_p represents the portion of the algorithm that can be improved by using a τ number of processors.

In Fig. 1, we present the theoretical speed-up determined by Amdahl’s law, Eq. (19), for algorithms Y_1, Y_2, Y_3, Y_4, Y_5, and Y_6. As a way of illustrating such a law, it is assumed that the algorithms are parallelized and that their parallelized fractions are, respectively, $f_p = 50\%$, $f_p = 80\%$, $f_p = 90\%$, $f_p = 95\%$, $f_p = 99\%$, and $f_p = 100\%$. The algorithm Y_1 hypothetically has $f_p = 50\%$, Y_2 hypothetically has $f_p = 80\%$ and so on. Thus, by analyzing Fig. 1, one can conclude, for example, that the Y_5 algorithm hypothetically has 99% of parallelization and, when executed with 28 processors, its speed-up is 22 times that of the Y_0 algorithm executed serially with only 1 processor.

3.3. Iteration error

The iteration error (ε_l) [39,46] is simply defined as the difference not absolute between the exact and approximate solution of the system of algebraic equations in a given iteration [3,47]

$$\varepsilon_l(\boldsymbol{\psi}(\mathbf{x}_i)) = \boldsymbol{\Psi}(\mathbf{x}_i) - [\boldsymbol{\psi}(\mathbf{x}_i)]^l, \tag{20}$$

where l represents the number of the current iteration in the solution process of the system of linear or nonlinear equations.

In this study, we use the Euclidean norm vector (L_2) [48] to perform the iteration error analyses. To do this, we evaluate L_2 over the difference between the numerical solution at iteration l and $l-1$, according to equation

$$L_2^l = \sqrt{\sum_{i=1}^{n} \left(\boldsymbol{\psi}(\mathbf{x}_i)^l - \boldsymbol{\psi}(\mathbf{x}_i)^{l-1}\right)^2}, \tag{21}$$

or on the residue $\mathbf{r}^l = \mathbf{f} - A\boldsymbol{\psi}^l$

$$L_2^l = \sqrt{\sum_{i=1}^{n} \left(\mathbf{r}^l - \mathbf{r}^{l-1}\right)^2}, \tag{22}$$

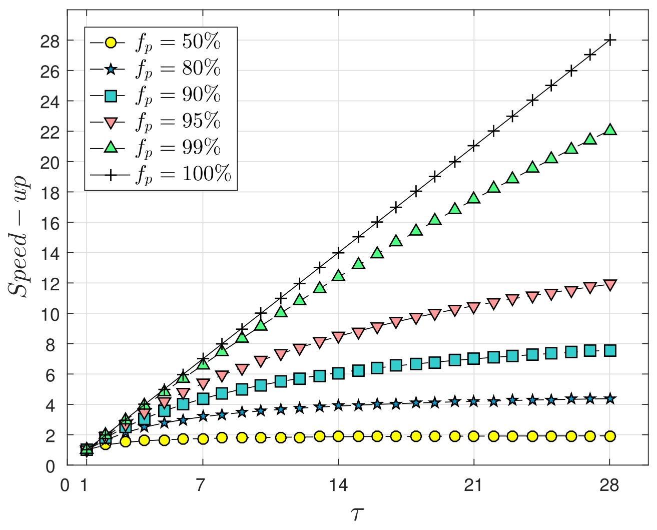


Fig. 1. Upper bound on parallelized algorithms with f_p ranging between 50% and 100%.

where n is order of the matrix.

In all analyses, we evaluate the dimensionless iteration error, that is, L_2^l/L_2^0, where L_2^0 is the error at iteration $l = 0$ with $\boldsymbol{\psi}(\mathbf{x}_i) = 0$. Note that $L_2^1/L_2^0 \approx 1$.

3.4. Multigrid

Geometric multigrid is a method capable of smoothing the error in a grid, and by means of transfer operators, injects information from the finer grid into the coarser grid using, for example, full weighting operator (process of restriction) [11,49] and from the coarser grid into the finer grid using bilinear interpolation operator (process of prolongation) [11,49] in a number of cycles, which can be $V(v_1, v_2)$, as shown in Fig. 2(a), or $W(v_1, v_2)$ or $F(v_1, v_2)$, with v_1 pre and v_2 post-smoothing. The information transferred from one grid to the other can be from just the residue, correction scheme (CS), or residue and solution, full approximation scheme (FAS). To facilitate the understanding of the transfer of information between grids, we highlight in red the idea of restriction (advance to coarser grid) operator, and in blue the prolongation (return to the finer grid) operator.

The algebraic multigrid is a method that performs procedures similar to the geometric multigrid, but the idea of information transfer between grids, Fig. 2(b), is applied directly to the matrix of coefficients associated with the system of algebraic equations determined by discretizing the mathematical model with some chosen numerical method [29–31]. In Table 3 we show the relationship among the coefficient matrix associated with the consistent discretization level and the matrices associated with the intermediate discretization sublevels until the inconsistent sublevel is reached. Since in our study we do not have a hierarchy of grids but rather discretization levels, the nomenclature was adapted to better represent the discretization levels using particles, which can be from the SPH method or any other particle method, as we can see in Fig. 2. In this way, the algebraic multigrid (AMG) becomes algebraic multilevel (AML).

In Table 2 we show the relationship between the main discretization and the number of uniform mean particle spacing. The symbol g represents distance between $\mathbf{x}_i$ particle and the immediately neighboring orthogonal projection of $\mathbf{x}_j$ on the straight line parallel to the x or y axis that passes through $\mathbf{x}_i$ (smallest distance between two particles).

To facilitate the understanding of this technique, we will adopt the following nomenclature:

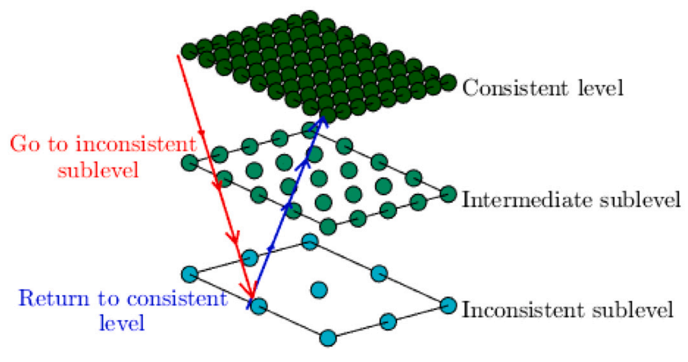
- $A_0\boldsymbol{\psi}_0 = \boldsymbol{f}_0$ main system with main matrix A_0 associated with consistent level;

Algorithm 2: Parallel Gauss-Seidel-Silva CSR smoother.

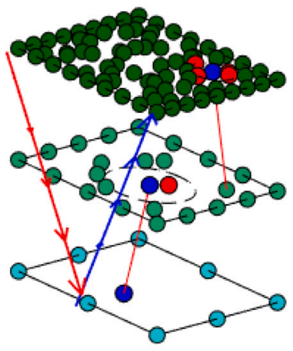
```
Input: $AA, JA, IA, AD, \boldsymbol{f}, tol, N_{ti}, itemax$
Output: $\boldsymbol{\psi}, erro, erro1, ite$
begin
    allocate: $res_{L2}(1 : itemax)$, $erro1(1 : itemax)$
    $ite = 0$
    $L_2sum = 0$
    $res_{L2} = \vec{0}$
    $res00_{L2} = 0$
    $erro = 0$
    $erro1 = \vec{0}$
    while $erro >= tol$ do
        $ite = ite + 1$
        !$OMP BARRIER
        for $k = 1, \dots, N_{ti}$ (performed in parallel by each thread) do
            $cont = IA(k) - 1$
            $c = IA(k+1) - IA(k)$
            $s = 0$
            for $i = 1, \dots, c$ do
                $cont = cont + 1$
                $j = JA(cont)$
                if $j > k$ then
                    $s = s + AA(cont)\boldsymbol{\psi}_{old}(j)$
                end
                if $j < k$ e $k \neq AD(k)$ then
                    $s = s + AA(cont)\boldsymbol{\psi}(j)$
                end
            end
            $\boldsymbol{\psi}(k) = [1/AA(AD(k))][\boldsymbol{f}(k) - s]$
            $L_2sum = L_2sum + [\boldsymbol{\psi}(k) - \boldsymbol{\psi}_{old}(k)]^2$
            $\boldsymbol{\psi}_{old}(k) = \boldsymbol{\psi}(k)$
        end
        serial region realized by single thread do
            !$OMP SINGLE
            $res_{L2}(ite) = \sqrt{L_2sum}$
            $res00_{L2} = res_{L2}(1)$
            $erro = res_{L2}(ite)/res00_{L2}$
            $erro1(ite) = erro$
            $L_2sum = 0$
            !$OMP END SINGLE
        end
        if $ite == itemax$ then
            stop
        else
            continue
        end
    end
end
```

- $A_{1:N_{Lmax}-2}\boldsymbol{\psi}_{1:N_{Lmax}-2} = \boldsymbol{f}_{1:N_{Lmax}-2}$ subsystems with submatrices $A_{1:N_{Lmax}-2}$ associated with intermediate sublevels;
- $A_{N_{Lmax}-1}\boldsymbol{\psi}_{N_{Lmax}-1} = \boldsymbol{f}_{N_{Lmax}-1}$ subsystem with submatrix $A_{N_{Lmax}-1}$ associated to the inconsistent sublevel.

Remark. we see in Table 3 that each main discretization determines a set of submatrices. In the case of the table, for example, we choose the main discretization 4096 × 4096 (number of uniform mean particle spacing), which generates a level of 4097 × 4097 particles (including the particles in the boundaries). In this way, A_{N_L+1} will always be coarser than A_{N_L}. Consequently, A_{N_L} always more finer than A_{N_L+1}, for all $N_L = 0, 1, \dots, N_{Lmax}-2$. In this study the texts have been adapted



(a)



(b)

Fig. 2. Representation of an AML $V(\nu_1, \nu_2)$ cycle (a) uniform particle discretization (b) non-uniform particle discretization.

Table 2
Spatial discretization parameters.

Number of main discretizations (N_{main})	$g = \Delta x = \Delta y$ (initial discretization)	Number of uniform mean particle spacing ($N_{p_x} \times N_{p_y}$)
1	g_1	8 × 8
2	g_2	16 × 16
3	g_3	32 × 32
4	g_4	64 × 64
5	g_5	128 × 128
6	g_6	256 × 256
7	g_7	512 × 512
8	g_8	1024 × 1024
9	g_9	2048 × 2048
10	g_{10}	4096 × 4096

so that the focus is on the hierarchy of matrices associated with the AML hierarchy. This adaptation does not cause any harm to the studies reported in our work, which have a focus, in general, on the hierarchy of grids.

The AML is also composed of two phases, namely: (i) setup phase and (ii) solution phase [29–31,50].

(i) setup phase: is responsible for generating the levels, matrices of coefficients and source terms, and for constructing the transfer operators (restriction and prolongation) among the levels involved in this process;
(ii) solution phase: is direct and applies operators previously defined in the setup phase. In this phase, the matrices and coarser source terms are visited according to a previously proposed cycle, such as the $V(\nu_1, \nu_2)$ cycle, for example.

To construct the matrices associated with the intermediate and inconsistent sublevels (A_1 until $A_{N_{Lmax}}$), we need to partition the matrix associated with the consistent level (A_0). The symbol $N_{Lmax} = log2(N_{p_x}) - 1 = log2(N_{p_y}) - 1$. To partition the coefficients of A_0, one needs to proceed as follows: let $i \in \Omega^0$, where Ω^0 denotes the indices set $\{1, 2, \ldots, n\}$. This partition generates two disjoint subsets, C^0 and F^0, where $\Omega^0 = C^0 \cup F^0$. The C^0, representing the coefficients which should be contained in the intermediate/inconsistent sublevel (C-coefficients), and F^0, the complementary set, in which are placed the coefficients not included in the intermediate/inconsistent sublevel. This partition is based on the strong algebraic connections. In addition, some other sets must be defined as [31].

$$\overline{N}_i = \{j \in \Omega^0 : j \neq i, a_{ij} \neq 0\}, \tag{23}$$

$$S_i = \{j \in \overline{N}_i : -a_{ij} \geq \theta \max_{a_{ik}<0} |a_{ik}| \text{with fixed } \theta,\ 0 < \theta < 1\}, \tag{24}$$

$$S_i^T = \{j \in \Omega^0 : i \in S_j\}, \tag{25}$$

where $\overline{N}_i$ is the set of neighboring coefficients associated with the particle $\mathbf{x}_j$, S_i determines the set of coefficients that strongly influences i and S_i^T is the set of coefficients which are strongly dependent of i. The constant parameter $\theta(0 < \theta < 0)$, represents the matrix order reduction (or level reduction) factor. This parameter is a measure that determines how strongly a matrix coefficient is connected to others. The Influence of θ on CPU time can be found in [32].

Upon completion of the $\Omega^0 = C^0 \cup F^0$ partition for each coefficient associated with particle $\mathbf{x}_i$, three other subsets are determined: C_i, D_i^S and D_i^w.

According to [51], these three subsets represent:

1. $C_i = C^0 \cap S_i$ are the coefficients of the new submatrix that strongly influence the particle $\mathbf{x}_i$.
2. $D_i^S = D_i \cap S_i$, and $D_i = N_i - C_i$, are all the neighboring coefficients of F^0 that strongly influence the coefficients of $\mathbf{x}_i$;
3. $D_i^W = D_i - S_i$, the coefficients that have a weak influence on $\mathbf{x}_i$. This set may contain coefficients in the matrix $A_{N_L - 1}$ (finer) and A_{N_L} (coarser). These coefficients are called weakly connected neighbors.

The interpolator operator (responsible for transferring information from a coarser matrix to a finer matrix) is given by [51], where h and H generically represent two consecutive levels, respectively, h being the finer and H the coarser,

$$(I_H^h e^H)_i = \begin{cases} e_i^H, & \text{if} \quad i \in C^h \\ \sum_{j \in C_i} \omega_{ij} e_j^H, & \text{if} \quad i \in F^h \end{cases}, \tag{26}$$

where

$$\omega_{ij} = -\frac{a_{ij} + \sum_{m \in D_i^S} \left(\frac{a_{im} a_{mj}}{\sum_{k \in C_i} a_{mk}} \right)}{a_{ii} + \sum_{n \in D_i^W} a_{in}}. \tag{27}$$

The constant parameter $\varepsilon > 0$, which represents the strong dependency in the submatrix, also implies the change in CPU time, as shown in [32,52]. The parameter ε is employed in the judgment if a coefficient $j \in D_i^S$ strongly influences a given coefficient in C, in such way that the interpolation is also influenced. The left-hand side of Eq. (26) for two levels, for example, can be written as $(I_1^0 e^1)_i$. As we are transferring information from the system $A_0 \boldsymbol{\psi}_0 = \boldsymbol{f}_0$ to $A_1 \boldsymbol{\psi}_1 = \boldsymbol{f}_1$, then we define

Table 3
Association between coefficient matrix order and number of uniform mean particle spacing.

Number of sublevel (N_L)	Number of uniform mean particle spacing ($N_{p_x} \times N_{p_y}$)	Matrix order (A)	Source term order (f)	Representation
0	4096 × 4096	$4095^2 \times 4095^2$	$4095^2 \times 1$	A_0, f_0 → consistent level
1	2048 × 2048	$2047^2 \times 2047^2$	$2047^2 \times 1$	A_1, f_1 → intermediate sublevel
2	1024 × 1024	$1023^2 \times 1023^2$	$1023^2 \times 1$	A_2, f_2 → intermediate sublevel
3	512 × 512	$511^2 \times 511^2$	$511^2 \times 1$	A_3, f_3 → intermediate sublevel
4	256 × 256	$255^2 \times 255^2$	$255^2 \times 1$	A_4, f_4 → intermediate sublevel
5	128 × 128	$127^2 \times 127^2$	$127^2 \times 1$	A_5, f_5 → intermediate sublevel
6	64 × 64	$63^2 \times 63^2$	$63^2 \times 1$	A_6, f_6 → intermediate sublevel
7	32 × 32	$31^2 \times 31^2$	$31^2 \times 1$	A_7, f_7 → intermediate sublevel
8	16 × 16	$15^2 \times 15^2$	$15^2 \times 1$	A_8, f_8 → intermediate sublevel
9	8 × 8	$7^2 \times 7^2$	$7^2 \times 1$	A_9, f_9 → intermediate sublevel
10	4 × 4	$3^2 \times 3^2$	$3^2 \times 1$	A_{10}, f_{10} → intermediate sublevel
$N_{Lmax} = log2(4096) - 1 = 11$	2 × 2	$1^2 \times 1^2$	$1^2 \times 1$	A_{11}, f_{11} → inconsistent sublevel

$A_1 = I_0^1 A_0 I_1^0$, and $(I_0^1) = (I_1^0)^T$ (see Algorithm 3). Therefore, it is necessary to define a data set, as seen in [52]:

$$S_i^D = \left\{ j \in D_i^S : \sum_{l \in C_i} |a_{ij}| > \varepsilon \left(\frac{|a_{ij}|}{\max |a_{ik}|} \right) \max |a_{jl}|, \text{with fixed } \varepsilon > 0 \right\}. \tag{28}$$

See also other details about interpolation in this same study [52]. Generically, the restriction operator is defined as $I_h^H = (I_H^h)^T$.

Remark. The G–S–S used within AML is not parallel because the algebraic multilevel structure has not been prepared to receive parallelization. Nevertheless, we have estimated results that account for parallelization, since studies involving AML with parallel G–S–S are attractive.

Algorithm 3: AML two levels correction scheme (CS).

```
Input: $\psi_0^0, A_0, f^0$
Output: $\psi^0$
1  Go to inconsistent sublevel - transfer operator: restriction
2  begin
3      Relax $\nu_1$ times on $A_0\psi^0 = f^0$ with initial guess $\psi_0^0 = 0$;
4      Compute the consistent level residual $\Omega^0 : \mathbf{r}^0 = f^0 - A_0\psi^0$;
5      Restrict it to be the inconsistent sublevel by $\mathbf{r}^1 = I_0^1\mathbf{r}^0$;
6      Solve $\Omega^1 : A_1\mathbf{e}^1 = \mathbf{r}^1$ with initial guess $\mathbf{e}_0^1 = 0$.
7  end
8  Return to consistent level - transfer operator: interpolation
9  begin
10     Interpolate the inconsistent sublevel error to the consistent
         level by $\Omega^0 : \mathbf{e}^0 = I_1^0\mathbf{e}^1$;
11     Correct the consistent level approximation by $\psi^0 \leftarrow \psi^0 + \mathbf{e}^0$;
12     Relax $\nu_2$ times on $A_0\psi^0 = f^0$ with initial guess $\psi^0$ (updated
         in line 11).
13 end
```

3.5. Complexity order

The equation that measures the complexity order over computational operations chosen to determine this metric is given by [11,49]:

$$t_{cpu}(N_t) = c(N_t)^{\overline{p}}, \tag{29}$$

where $t_{cpu}(N_t)$ is the CPU time for each discretization level, N_t is the number of variables, c a constant of proportionality and $\overline{p}$ the complexity order.

4. Numerical results

The computational code is implemented in Fortran 95 language that runs AML to determine the numerical solutions obtained with the SPH method. Our study aims to verify the efficiency of AML to accelerate the convergence of SPH numerical solutions when considering uniform and non-uniform discretizations. Thus, to make our study coherent, we determine all real-type variables with quadruple (or extended) precision. However, as a code checking tool, the use of double precision is sufficient. The software used was Microsoft® Visual Studio® 2008 compiler v. 9.0.21022.8 RTM. The hardware architecture has a 3.4 GHz Intel Core (TM)™ i7-6700 processor, and with 16 GB of RAM hosting 64-bit Windows® 10. After verification of the computational code, the simulations were run on the GridUNESP cluster.

4.1. Canonical discretization with fixed sensor particle

According to [53], the canonical disorder (or perturbation) is defined as a measure assigned to the particles in the initial configuration, i.e., over the uniform discretization that considers $\Delta y = \Delta x = 1/h$, where $h = 1/N_{p_x}$. This measure is given by

$$R = \eta \Delta x, \tag{30}$$

where $\eta \in [0, 0.5[$. The representation of the canonical disorder defined by Eq. (30) can be seen in Fig. 3. The canonical perturbation imposes a multidirectional displacement, so that, the particle moves only within the region bounded by the circle of radius R. In other words, analyzing a Fig. 3, we see that the particle 41 does not invade the bounded space of any of the neighboring particles 31, 32, 33, 40, 42, 49, 50 and, 51. This dynamic prevents collisions among particles and allows analyses of the effects of disorder to be performed with less difficulty.

With the discretization shown in [53], in [20] the canonical discretization with fixed sensor particle was defined, allowing a variable to be evaluated at all levels of the main discretization, because a fixed particle remains positioned in all these discretizations, while its neighbor can move according to the canonical disorder. This technique will contribute to the advancement of studies of repeated Richardson extrapolation (RRE) [10,21–26] in multidimensional problems, continuing the studies shown in [20].

Definition 2 describes what [20] define as a fixed sensor particle, and this technique was applied in all our studies presented in this paper. Thus, with $\eta = 0$, we have the uniform particle discretization (ordered distribution), and for $\eta \in [0, 0.5[$ we determine the non-uniform particle discretization (disordered distribution).

Definition 2. A fixed sensor particle $\mathbf{x}_s(\mathbf{Q})$ or $\mathbf{x}_s(\mathbf{Q}, t)$, where $\mathbf{x}_s$ is any chosen particle that has coincident coordinates $\mathbf{Q}$ at all discretization levels and time steps simultaneously [20,37,39].

In Figs. 4(a) and 4(b), the sensor particle was defined as $\mathbf{x}_s = \mathbf{x}(1/2, 1/2)$, occupying the 25 position ($\mathbf{x}_{25}$) obtained with lexicographic

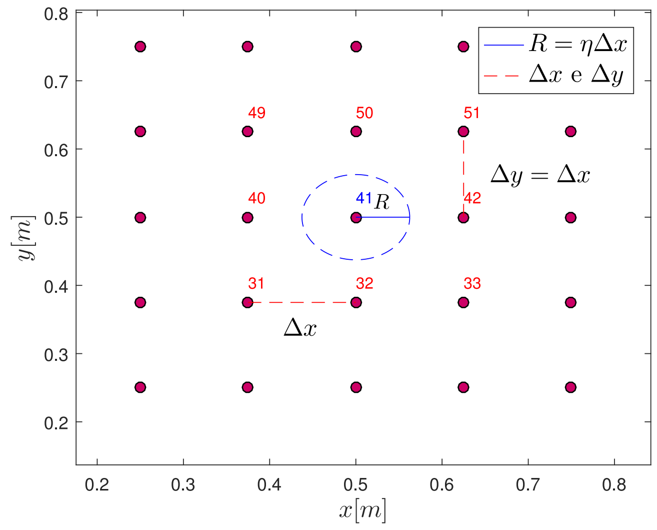


Fig. 3. Representation of the particle motion area with canonical disorder.

Table 4
Simulation parameters.

Parâmetro	AML	SL
Tolerance for G–S–S	$1.0E{-}08$	$1.0E{-}08$
Internal iterations	2	–
External cycles	7	–
Level reduction factor	0.25	–
Strong dependency factor at the inconsistent sublevel	0.35	–

ordering. In Fig. 4(a) the value of $\eta = 0$, i.e. we have a uniform discretization, while in Fig. 4(b), the disorder $\eta = 0.499999$ is applied to all particles except for the fixed sensor particle that occupies position 25. Thus, it is equivalent to say that $\boldsymbol{\psi}(1/2, 1/2) = \boldsymbol{\psi}(\mathbf{x}_s)$, for $\mathbf{x}_s(1/2, 1/2)$

See that in our examples, we can create 49 fixed sensor particles, which are all the particles from the coarsest main discretization, i.e., $N_{ti} = 7 \times 7$ (inner particles only). This allows variables of interest, for example the temperature $\boldsymbol{\psi}$, to be evaluated locally in all regions of the entire domain.

4.2. Quantitative verification of numerical solutions

The AML parameters used in the simulations were reproduced from the studies [32,54] and in SL the same tolerance for the iteration error (ϵ_l) was maintained. These parameters are shown in Table 4. The total number of particles varied among 81 and 16 785 409 (over 16 million) always with constant refinement ratio $q = 2$ ($g_{N_{\text{main}}}/g_{N_{\text{main}}+1} = 2$) in x and y directions for main discretization and constant coarsening ratio $re = 1/2$ (g_{N_L+1}/g_{N_L}) for generating the submatrices of the AML.

In our study, the source of numerical error called iteration error is considered negligible, because as shown in Figs. 5(a) and 5(b), the order of magnitude of this error between two consecutive iterations has already reached the order of magnitude of the round-off error. This error is calculated using the dimensionless form, that is, the ratio between the Euclidean norm of the error at the lth iteration (L_2^l) and the initial iteration (L_2^0). In Fig. 5(b) we can see the dimensionless error oscillating around *1.0E–32*, which makes the influence of ϵ_l negligible.

In Table 5, the degree of sparsity (DS) of the matrices associated with the system of linear equations, obtained when discretizing the mathematical model of steady state heat diffusion with the SPH method, is presented. This information is important, because one should not make the mistake of solving the systems in sparse matrix form. In this study we use the CSR method. CSR method is capable of

Table 5
Degree of sparsity of the matrices [55].

Number of uniform mean particle spacing	Matrix order	Degree of sparsity
8 × 8	$7^2 \times 7^2$	64.9729%
16 × 16	$15^2 \times 15^2$	90.5956%
32 × 32	$31^2 \times 31^2$	97.5960%
64 × 64	$63^2 \times 63^2$	99.3939%
128 × 128	$127^2 \times 127^2$	99.8479%
256 × 256	$255^2 \times 255^2$	99.9619%
512 × 512	$511^2 \times 511^2$	99.9905%
1024 × 1024	$1023^2 \times 1023^2$	99.9976%
2048 × 2048	$2047^2 \times 2047^2$	99.9994%
4096 × 4096	$4095^2 \times 4095^2$	99.9999%

accelerating the convergence of the solution of the systems with respect to the sparse form.

4.2.1. 2D steady state heat diffusion model (C1-2Ds)

In multidimensional problems the complexity order of solver is an appropriate metric to evaluate the coherence of the time required to determine the numerical solutions when using single level and multilevel methods, as well as for singlegrid and multigrid. In the single level (equivalent to a singlegrid) methods the complexity order is $\overline{p} = 2$, while in the multilevel (equivalent to a multigrid) methods it is unitary, that is, $\overline{p} = 1$ in accordance with [29,30,49]. This information can be confirmed by looking at the data in Table 6. It can also be seen that the parallel solver does not have the same efficiency as the algebraic multilevel with G–S–S smoother that can reduce one complexity order compared to the single level solver. It is important to note that the solver of the AML method is not parallelized, that is, by parallelizing such a smoother, the complexity order unity will be reduced.

Regarding the complexity order of the cases where non-uniform discretization is applied, the values obtained are slightly larger than for uniform discretization and this change is related to the slower convergence of the numerical solutions, as we can see in Table 7.

After the verification of the complexity order, in Figs. 6(a) and 6(b), we present the efficiency of AML by varying the number of sublevels. For this we use CPU time (s) in y-log scale *versus* number of sublevels (N_L) for the case of uniform particle discretization (ordered distribution) and non-uniform particle discretization (disordered distribution). We change the three levels of principal discretization in the case of the non-uniform discretization, Fig. 6(b), because the goal is to show the efficiency of AML with respect to the number of sublevels, and this can be proven in any principal discretization. In addition, the non-uniform discretization presents slower convergence and therefore we decided not to show the same values as in Fig. 6(a). Evidently, this change does not harm the analyses. In all cases we note that the efficiency of the AML is higher when considering the maximum number of sublevels. Also in Fig. 6(a), one can see that we use the minimum of 2 sublevels because we have reached the maximum simulation time within GridUNESP which is thirty days. See that for $N_t = 4097 \times 4097$ the time needed to determine the numerical solution with only the main discretization level (zero sublevel or SL), would be something over thirty years.

Remark. the total number of sublevels is defined by $N_{Lmax} - 1$. Revisiting Table 3, we can conclude that the sublevels are all those among $N_L = 1$ and $N_L = 11$. In practice, the analyses in Figs. 6(a) and 6(b) are performed by varying the number of sublevels, but saying that we are varying the number of levels is acceptable and may be familiar in the literature.

To highlight the efficiency of AML, we perform a theoretical analysis on parallelization and present some results so that the comparison between the results obtained with parallel G–S–S and AML with serial G–S–S can be made with coherence. We adopt the application

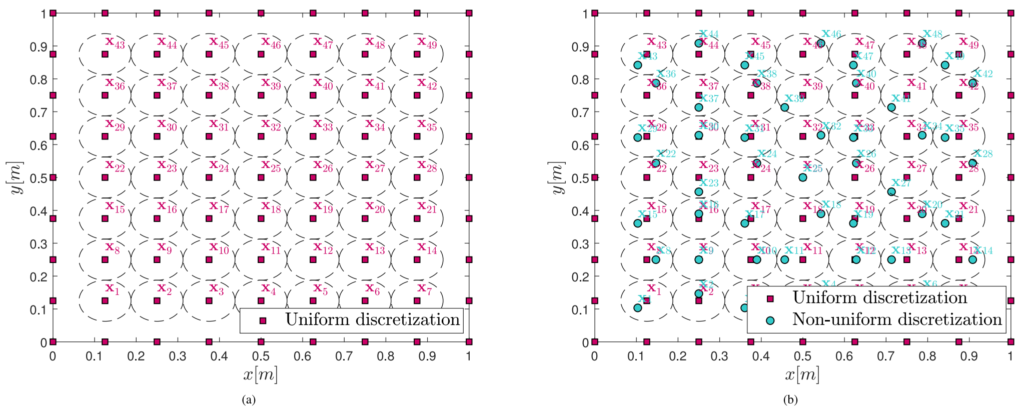


Fig. 4. Representation of the canonical discretization with fixed sensor particle (a) $\eta = 0$ (b) $\eta = 0.499999$.

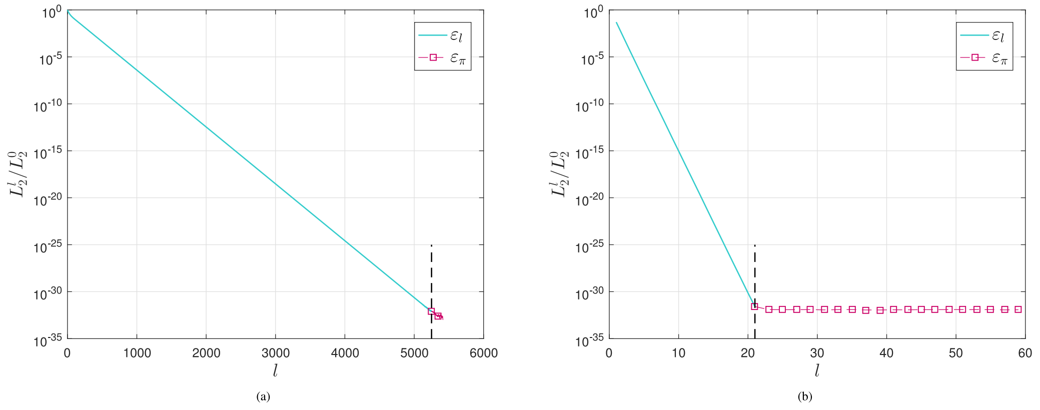


Fig. 5. Decrease iteration error *versus* number of iterations for C1-2Ds (a) parallel G–S–S SL (b) serial G–S–S AML.

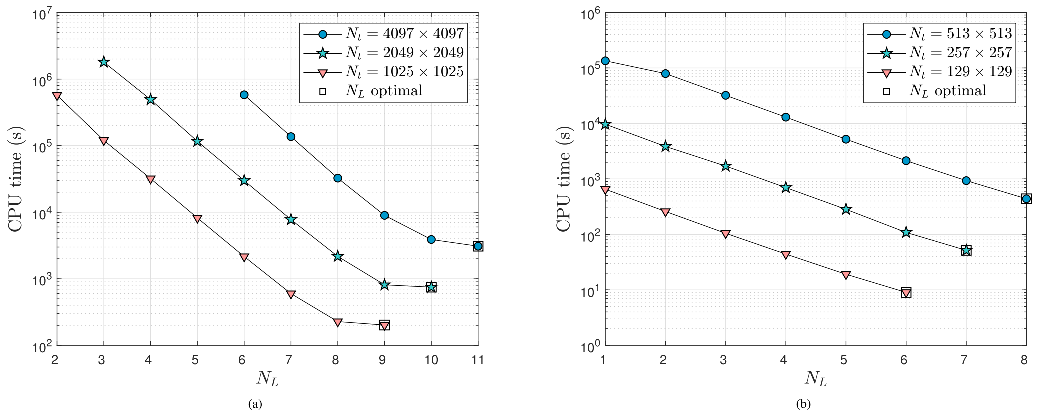


Fig. 6. CPU time *versus* number of AML sublevels for C1-2Ds (a) uniform particle discretization (b) non-uniform particle discretization.

programming interface (API) called Open Multi-processing (OpenMP) and, according to Amdahl's law [43], we identify the 90% parallelized fraction for parallel G–S–S, described in Algorithm 2. This means that using 28 CPU's, the speed-up over serial G–S–S is $S_{28} = 8$ times. This information is verified in the main discretization with $N_t = 513 \times 513$ particles, as we show in Fig. 7. To calculate the parallelized fraction (f_p), we averaged the processing time across several simulations, in each main discretization. In addition, the speed-up S_τ is obtained by the ratio between the CPU time of the G–S–S series (t_0) and parallel G–S–S (t_τ) using τ CPU's, i.e., $S_\tau = t_0/t_\tau$. For coherence in the speed-ups results, $S_1 \approx 1$.

In Fig. 8(a), we present the CPU time to determine the numerical solutions at all the principal discretization levels and compare SL with serial G–S–S, SL with parallel G–S–S, AML with serial G–S–S and the projection of AML with parallel G–S–S smoother, considering $f_p = 90\%$. In all these cases, the CSR structure is considered, including in the projection calculations. We realize that parallelization of G–S–S is advantageous over serial G–S–S only from the main discretization

Table 6
Complexity order with canonical uniform discretization for case C1-2Ds.

Method	c	$\bar{p}$
SL with serial G–S–S CSR solver	$1.996E{-}06$	2.013
SL with parallel G–S–S CSR solver	$1.672E{-}04$	1.492
AML with serial G–S–S CSR smoother	$8.878E{-}05$	1.005

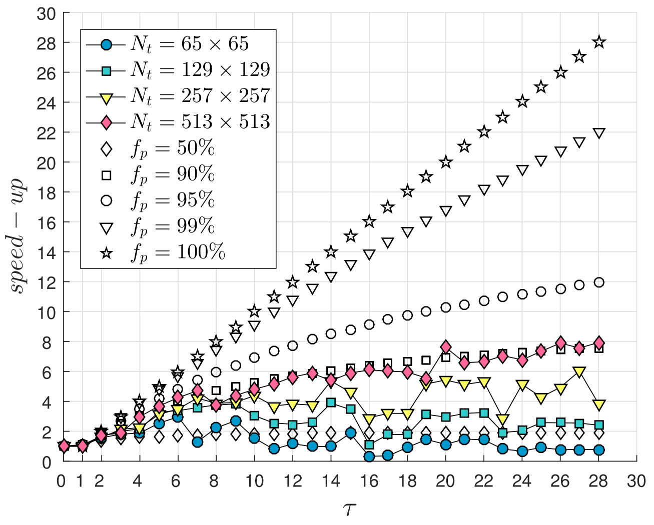


Fig. 7. Scalability of the G–S–S solver CSR for C1-2Ds.

Table 7
Complexity order with canonical non-uniform discretization for case C1-2Ds.

Method	c	$\bar{p}$
SL with serial G–S–S CSR solver	2.230E−06	2.023
SL with parallel G–S–S CSR solver	7.883E−06	1.729
AML with serial G–S–S CSR smoother	7.414E−05	1.227

with $N_t = 129 \times 129$ particles, while AML and its projection is more advantageous always. Note that using SL with serial G–S–S we simulate up to $N_t = 513 \times 5013$ and using SL with parallel G–S–S, up to $N_t = 1025 \times 1025$, because the time to determine the another solutions is quite high. Furthermore, the slope of the single level straight lines is different from that of the multilevels, which means that AML is progressively more advantageous, i.e. the more particles we use to discretize the domain, the more efficient AML becomes. In all our study every time we use the G–S–S solver or the G–S–S smoother (for AML only) the CSR structure is being used, so we will hide this information from now on because it is known. For the case of the non-uniform discretization, Fig. 8(b), parallelization starts to be more advantageous from level $N_t = 65 \times 65$, this is because the time to determine the numerical solution using the serial G–S–S was longer than with the uniform discretization.

Analyzing the results shown in Fig. 9(a), one can observe the big difference between the speed-up of the SL with G–S–S parallel *versus* SL with serial G–S–S and the AML with serial G–S–S smoother *versus* SL with serial G–S–S solver. While parallel determines an $S_{28} = 8$, AML determines $S_0 = 4084$ and its projection $S_{28} = 32670$. We are faced with an excellent result for determining the fast convergence of the numerical solutions determined with the SPH method. The previous values $S_{28} = 3$ for parallel G–S–S are not shown because only from the main discretization level $N_t = 129 \times 129$ does parallelization start to be advantageous, i.e. the speed-up greater than 1 is determined at the level $N_t = 129 \times 129$. With the non-uniform discretization, Fig. 9(b), the speed-up for the parallel G–S–S is higher due to the increased time to determine the numerical solutions with the serial solver. However, we kept the estimated projection with the same 8 times speed-up as with the uniform discretization. See also that as a consequence of the results shown in Fig. 8(b), parallelization is already more advantageous from the level $N_t = 65 \times 65$. Moreover, the speed-up of AML is more than a thousand times when compared to uniform discretization, reinforcing our results that particle disorder does not characterize an impossibility

Table 8
Number of cycles and iteration erro using AML.

External cycle	Internal iteration ν_1	Internal iteration ν_2	Iteration error
1	2	2	6.01E−01
2	2	2	1.78E−02
3	2	2	6.03E−04
4	2	2	2.13E−05
5	2	2	7.69E−07
6	2	2	2.85E−08
7	2	2	1.06E−09

to efficiently solve systems of linear equations, and the more particles we use in the discretization of the domain, the greater the reduction in CPU time. Again we use a scale y-log.

Now that we have identified the efficiency of the AML with serial G–S–S smoother with respect to the SL with parallel G–S–S and also SL with serial G–S–S, we will present the graph of the global analytical and numerical solutions for the entire temperature field. In Figs. 10(a) and 10(b), we note that the profile of the numerical solution is coherent with the analytical solution for the case where the non-uniform discretization is applied. We do not present the graph of the numerical solution with uniform discretization because we are emphasizing the results obtained when applying disorder on the particles, which sets up something closer to the flow models where the particles have motion. However, the results with uniform discretization are coherent and are therefore not shown.

In Table 8, we show the number of external cycles and internal iterations (pre and post-smoothing) and also the iteration error until the stop criteria (or tolerance) is reached, which is *1.0E–08*.

4.2.2. 2D steady state heat diffusion model (C2-2Ds)

In the second case, we confirmed all the results presented in the first case. In Fig. 11, we see that, once again, using the maximum number of sublevels guarantees the lowest CPU time for uniform, Fig. 11(a), and non-uniform discretization, Fig. 11(b). We note that the results of the second case for CPU time and speed-up are very similar to the first case, so to avoid repetition we decided to show only the graph of the AML efficiency *versus* number of sublevels of the AML and the analytical and numerical solutions, Figs. 12(a) and 12(b).

5. Conclusions

We present a modification to the standard Gauss–Seidel, which we call Gauss–Seidel–Silva. Such a solver can be a combination between Gauss–Jacobi and standard Gauss–Seidel that allows the use of parallelization. This allows such a solver to be used as a parallel smoother within AML. Furthermore, in both cases presented in the study, we prove the efficiency of AML with serial G–S–S smoother over SL with parallel G–S–S solver. We also apply discretization with fixed sensor particle that will extend the possibilities of studies to increase the accuracy of order and reduce the discretization error in flows with particle motion.

Based on the numerical results we make the following highlights:

1. The highest efficiency of AML is achieved when we use the maximum number of sublevels;
2. We determine the speed-up of 4084 times for uniform discretization and 5136 times for non-uniform discretization;
3. The AML can be applied to accelerate the convergence of numerical solutions in any kind of geometry;
4. We present the estimated speed-up of 41 082 times for the AML with parallel G–S–S smoother using non-uniform particle discretization.

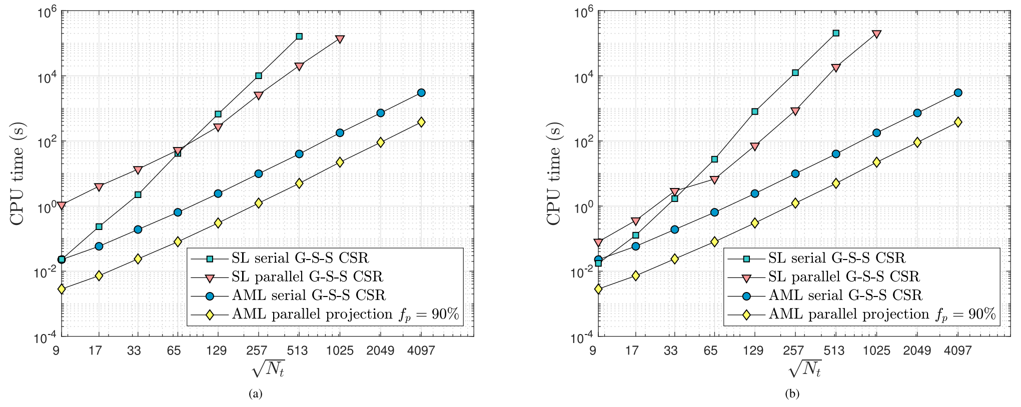


Fig. 8. CPU time *versus* number of particles for C1-2Ds (a) uniform particle discretization and (b) non-uniform particle discretization.

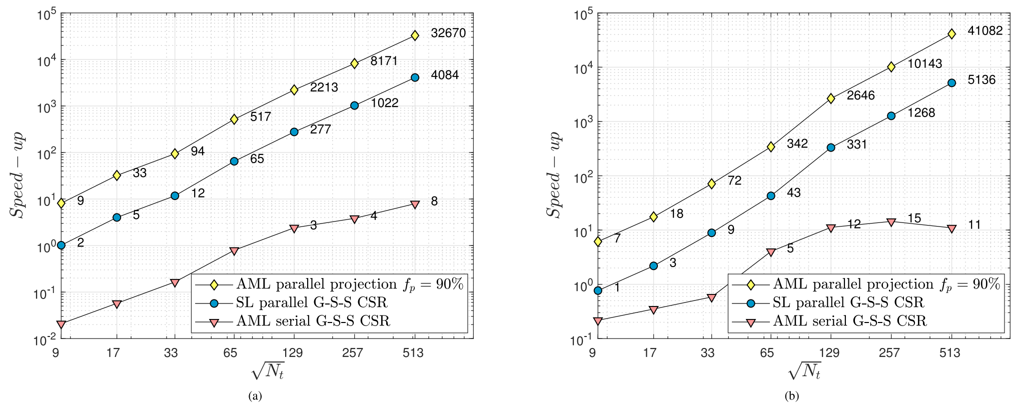


Fig. 9. Speed-up *versus* number of particles for C1-2Ds (a) uniform particle discretization and (b) non-uniform particle discretization.

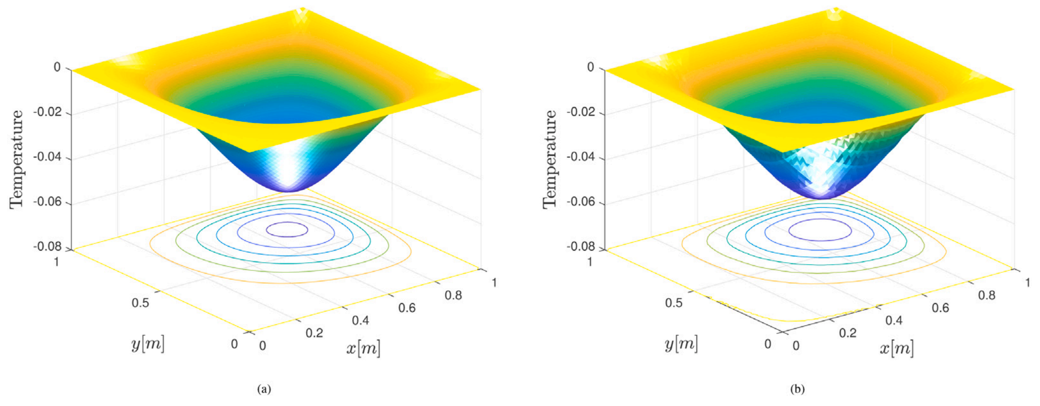


Fig. 10. Solutions (a) analytical and (b) numerical with non-uniform particle discretization for C1-2Ds.

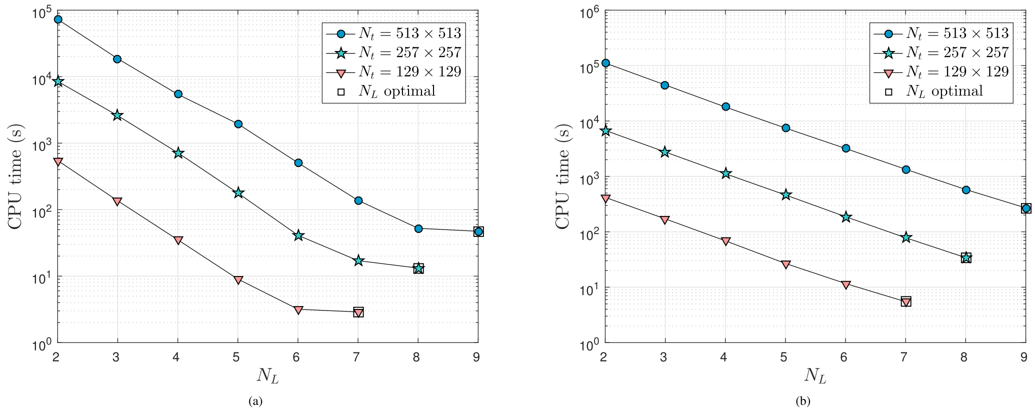


Fig. 11. CPU time *versus* number of AML sublevels for C2-2Ds (a) uniform particle discretization (b) non-uniform particle discretization.

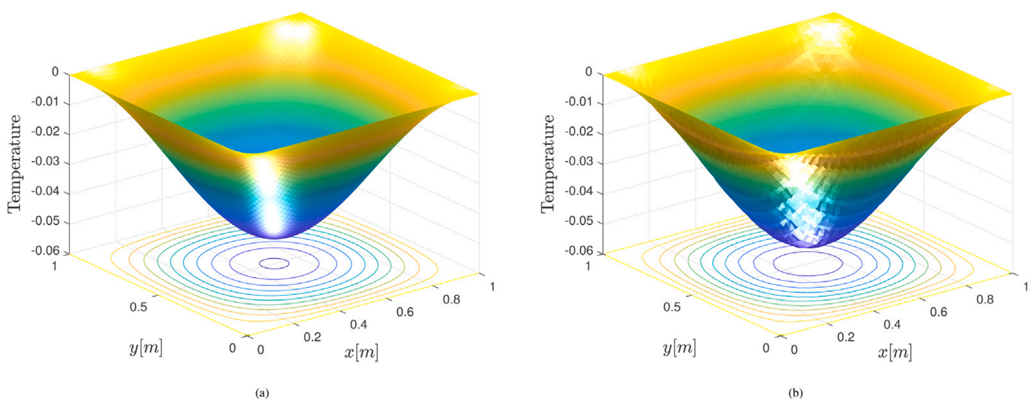


Fig. 12. Solutions (a) analytical and (b) numerical with non-uniform particle discretization for C2-2Ds.

CRediT authorship contribution statement

L.P. da Silva: Formal analysis, Investigation, Methodology, Validation, Writing – original draft, Writing – review & editing. **C.H. Marchi:** Supervision. **M. Meneguette:** Supervision. **R. Suero:** Formal analysis, Methodology, Validation.

Declaration of competing interest

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

Data availability

The data that has been used is confidential.

Acknowledgments

We would like to thank the Graduate Program of Numerical Methods in Engineering (PPGMNE) from the Federal University of Paraná, Curitiba, PR, Brazil, and the Brazilian Space Agency (AEB), National Council of Technological and Scientific Development (CNPq, Brazil). The second author was supported by a CNPq scholarship. The authors are also grateful to Dr. Stüben, for sharing his AMG1R6 code, which was used as reference for all the simulations and optimizations presented in this work. This study was supported by the resources supplied by the Center for Scientific Computing (NCC/GridUNESP) of São Paulo State University (UNESP) and the support of the Laboratory of Numerical Experimentation (LENA). This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

References

[1] S.V. Patankar, Recent developments in computational heat transfer, J. Heat Transfer 110 (4b) (1988) 1037–1045.
[2] G.D. Smith, Numerical Solution of Partial Differential Equations: Finite Difference Methods, Oxford University Press, 1985.
[3] J.H. Ferziger, M. Perić, Computational Methods for Fluid Dynamics, third ed., Springer Science & Business Media, 2002.
[4] R.J. LeVeque, Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems, Vol. 98, SIAM, 2007.
[5] P. Gallagher, R. Marcer, C. Berhault, C. de Jouette, H.C. Raven, L. Eça, L. Broberg, C.E. Janson, Q.X. Gao, S. Txopeus, B. Alessandrini, T. van Terwisga, M. Hoekstra, H. Streckwall, F. Salvatore, Best practice guidelines for the application of computational fluid dynamics in marine hydrodynamics, Virtue Proj. Relat. Técnico 221 (2009) 16.
[6] S. Patankar, Numerical Heat Transfer and Fluid Flow, CRC Press, 1980.
[7] R.J. LeVeque, Finite Volume Methods for Hyperbolic Problems, Vol. 31, Cambridge University Press, 2002.
[8] O.C. Zienkiewicz, R.L. Taylor, P. Nithiarasu, J.Z. Zhu, The Finite Element Method, Vol. 3, McGraw-hill London, 1977.
[9] P.G. Ciarlet, The Finite Element Method for Elliptic Problems, Vol. 40, SIAM, 2002.
[10] L.P. da Silva, B.B. Rutyna, A.R.S. Righi, M.A.V. Pinto, High order of accuracy for Poisson equation obtained by grouping of repeated richardson extrapolation with fourth order schemes, CMES Comput. Model. Eng. Sci. 128 (2) (2021) 699–715.
[11] U. Trottenberg, C.W. Oosterlee, A. Schüller, Multigrid, Elsevier, 2001.
[12] M.A.V. Pinto, C. Rodrigo, F.J. Gaspar, C.W. Oosterlee, On the robustness of ILU smoothers on triangular grids, Appl. Numer. Math. 106 (2016) 37–52.
[13] S.R. Franco, F.J. Gaspar, M.A.V. Pinto, C. Rodrigo, Multigrid method based on a space-time approach with standard coarsening for parabolic problems, Appl. Math. Comput. 317 (2018) 25–34.
[14] D.C. Zanatta, L.K. Araki, M.A.V. Pinto, D.F. Moro, Performance of geometric multigrid method for two-dimensional Burgers' equations with non-orthogonal, structured curvilinear grids, CMES Comput. Model. Eng. Sci. 125 (3) (2020) 1061–1081.
[15] C.H. Marchi, C.D. Santiago, C.A.R.D. Carvalho Jr., Lid-driven square cavity flow: A benchmark solution with an 8192× 8192 grid, J. Verif. Valid. Uncertain. Quant. 6 (4) (2021).

[16] C.D. Santiago, C.H. Marchi, L.F. Souza, Performance of geometric multigrid method for coupled two-dimensional systems in CFD, Appl. Math. Model. 39 (9) (2015) 2602–2616.

[17] M.L. de Oliveira, M.A.V. Pinto, S.D.F.T. Gonçalves, G.V. Rutz, On the robustness of the xy-zebra-Gauss-seidel smoother on an anisotropic diffusion problem, CMES Comput. Model. Eng. Sci. 117 (2) (2018) 251–270.

[18] G.V. Rutz, M.A.V. Pinto, S.D.F.T. Gonçalves, On the robustness of the multigrid method combining ILU and partial weight applied in an orthotropic diffusion problem, Rev. Int. Métodos Numér. para Cálc. Diseño Ing. 35 (1) (2019).

[19] M. Anunciação, R.L.A. Pinto, Solution of the Navier-Stokes equations using projection method and preconditioned conjugated gradient with multigrid and ilu solver, Rev. Int. Métodos Numér. para Cálc. Diseño Ing. 36 (1) (2020).

[20] L.P. da Silva, C.H. Marchi, M. Meneguette, A.C. Foltran, Robust RRE technique for increasing the order of accuracy of SPH numerical solutions, Math. Comput. Simulation 199 (2022) 231–252.

[21] P.J. Roache, P.M. Knupp, Completed richardson extrapolation, Commun. Numer. Methods Eng. 9 (5) (1993) 365–374.

[22] C.H. Marchi, L.A. Novak, C.D. Santiago, A.P.S. Vargas, Highly accurate numerical solutions with repeated richardson extrapolation for 2D Laplace equation, Appl. Math. Model. 37 (2013) 7386–7397.

[23] C.H. Marchi, L.K. Araki, A.C. Alves, R. Suero, S.F.T. Gonçalves, M.A.V. Pinto, Repeated richardson extrapolation applied to the two-dimensional Laplace equation using triangular and square grids, Appl. Math. Model. 37 (2013) 4661–4675.

[24] C.H. Marchi, E.M. Germer, Effect of the CFD numerical schemes on repeated Richardson extrapolation (RRE), Appl. Comput. Math. 2 (2013) 128.

[25] C.H. Marchi, M.A. Martins, L.A. Novak, L.K. Araki, M.A.V. Pinto, S.F.T. Gonçalves, D.F. Moro, I.S. Freitas, Polynomial interpolation with repeated richardson extrapolation to reduce discretization error in CFD, Appl. Math. Model. 40 (21–22) (2016) 8872–8885.

[26] C.H. Marchi, F.F. Giacomini, C.D. Santiago, Repeated richardson extrapolation to reduce the field discretization error in comptational fluid dynamics, Numer. Heat Transfer B 70 (4) (2016) 340–353.

[27] M.F. Malacarne, M.A.V. Pinto, S.R. Franco, Performance of the multigrid method with time-stepping to solve 1D and 2D wave equations, Int. J. Comput. Methods Eng. Sci. Mech. 23 (1) (2022) 45–56.

[28] H. Kothari, R. Krause, A generalized multigrid method for solving contact problems in Lagrange multiplier based unfitted finite element method, Comput. Methods Appl. Mech. Engrg. 392 (2022) 114630.

[29] K. Stüben, Algebraic multigrid (AMG): experiences and comparisons, Appl. Math. Comput. 13 (3–4) (1983) 419–451.

[30] J.W. Ruge, K. Stüben, Algebraic multigrid, in: Multigrid Methods, SIAM, 1987, pp. 73–130.

[31] K. Stüben, A review of algebraic multigrid, in: Numerical Analysis: Historical Developments in the 20th Century, Elsevier, 2001, pp. 331–359.

[32] R. Suero, M.A.V. Pinto, C.H. Marchi, L.K. Araki, A.C. Alves, Analysis of algebraic multigrid parameters for two-dimensional steady heat diffusion equations., Appl. Math. Model. 36 (1) (2012) 2996–3006.

[33] A. Brandt, Algebraic multigrid theory: The symmetric case, Appl. Math. Comput. 19 (1–4) (1986) 23–56.

[34] V. Gingold, J.J. Monaghan, Smoothed particle hydrodynamics - theory and application to non-spherical stars, R. Astron. Soc. 181 (1977) 375–389.

[35] L.B. Lucy, A numerical approach to the testing of the fission hypothesis, Astron. J. 82 (1977) 1013–1024.

[36] L. Brookshaw, A method of calculating radiative heat diffusion in particle simulations, in: Proceedings of the Astronomical Society of Australia, Vol. 6, 1985, pp. 207–210.

[37] L.P. da Silva, Verification of Numerical Solutions in Diffusive Problems Solved with the Smoothed Particle Hydrodynamics Method (in Portuguese) (Ph.D. thesis), Federal University of Paraná, Curitiba, Brazil, 2022.

[38] J.P. Morris, Analysis of Smoothed Particle Hydrodynamics with Applications (Ph.D. thesis), Monash University, 1996.

[39] L. Pereira da Silva, M. Meneguette Junior, C.H. Marchi, Numerical error analysis and heat diffusion models, in: Numerical Solutions Applied to Heat Transfer with the SPH Method: A Verification of Approximations for Speed and Accuracy, Springer International Publishing, Cham, 2023, pp. 51–75.

[40] A. Buluç, J.T. Fineman, M. Frigo, J.R. Gilbert, C.E. Leiserson, Parallel sparse matrix-vector and matrix-transpose-vector multiplication using compressed sparse blocks, in: Proceedings of the Twenty-First Annual Symposium on Parallelism in Algorithms and Architectures, 2009, pp. 233–244.

[41] R. Pozo, K.A. Remington, A. Lumsdaine, SparseLib++ v. 1.5, in: Sparse Matrix Class Library, Reference Guide, 1996.

[42] L. Pereira da Silva, M. Meneguette Junior, C.H. Marchi, Numerical modeling of heat diffusion, in: Numerical Solutions Applied to Heat Transfer with the SPH Method: A Verification of Approximations for Speed and Accuracy, Springer International Publishing, Cham, 2023, pp. 7–49.

[43] G.M. Amdahl, Validity of the single processor approach to achieving large scale computing capabilities, in: Proceedings of the April 18-20, 1967, Spring Joint Computer Conference, 1967, pp. 483–485.

[44] G. Galante, Métodos Multigrid Paralelos em Malhas Não Estruturadas Aplicados à Simulação de Problemas de Dinâmica de Fluidos Computacional e Transferência de Calor (in Portuguese) (Master's thesis), Universidade Federal do Rio Grande do Sul, Porto Alegre, 2006.

[45] X.-H. Sun, Y. Chen, Reevaluating Amdahl's law in the multicore era, J. Parallel Distrib. Comput. 70 (2) (2010) 183–188.

[46] C.H. Marchi, Verification of Unidimensional Numerical Solutions in Fluid Dynamics (in Portuguese) (Ph.D. thesis), Federal University of Santa Catarina, Florianópolis, Brazil, 2001.

[47] P.J. Roache, Verification and Validation in Computational Science and Engineering, Hermosa, 1998.

[48] R.L. Burden, J.D. Faires, A.M. Burden, Numerical Analysis, tenth ed., 2016.

[49] P. Wesseling, An Introduction to Multigrid Methods, Book News, Inc., 2004.

[50] R.D. Falgout, An Introduction to Algebraic Multigrid, Technical Report, 2006, pp. 24–33.

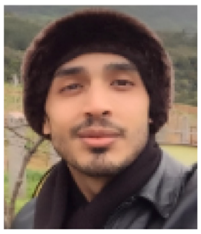
[51] W.L. Briggs, V.E. Henson, S.F. McCormick, A Multigrid Tutorial, second ed., SIAM, 2000.

[52] C. Iwamura, F.S. Costa, I. Sbarski, A. Easton, N. Li, An efficient algebraic multigrid preconditioned conjugate gradient solver, Comput. Methods Appl. Mech. Engrg. 192 (20–21) (2003) 2299–2318.

[53] G. Chaussonnet, S. Braun, L. Wieth, R. Koch, H.-J. Bauer, Influence of particle disorder and smoothing length on SPH operator accuracy, in: Conference Paper - 10th International SPHERIC Workshop, 2015.

[54] R. Suero, Parameter Optimization of Algebraic Multigrid Method for Two-Dimensional Diffusive Problems (in Portuguese) (Ph.D. thesis), Federal University of Paraná, Curitiba, Brazil, 2010.

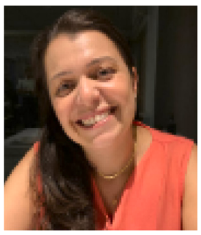
[55] L. Pereira da Silva, M. Meneguette Junior, C.H. Marchi, SPH applied to computational heat transfer problems, in: Numerical Solutions Applied to Heat Transfer with the SPH Method: A Verification of Approximations for Speed and Accuracy, Springer International Publishing, Cham, 2023, pp. 77–115.



Luciano Pereira da Silva http://lattes.cnpq.br/1873355458944068

Carlos Henrique Marchi http://lattes.cnpq.br/8251643344377056

Messias Meneguette Junior http://lattes.cnpq.br/1531018187057108



Roberta Suero http://lattes.cnpq.br/6702579185974800