Computer Modeling in Engineering & Sciences
DOI:10.32604/cmes.2020.012634
Article

Tech Science Press

Performance of Geometric Multigrid Method for Two-Dimensional Burgers' Equations with Non-Orthogonal, Structured Curvilinear Grids

Daiane Cristina Zanatta[1,*], Luciano Kiyoshi Araki[2], Marcio Augusto Villela Pinto[2] and Diego Fernando Moro[3]

[1]State University of Centro-Oeste, Irati, Brazil
[2]Federal University of Parana, Department of Mechanical Engineering, Curitiba, Brazil
[3]Positivo University, Curitiba, Brazil
*Corresponding Author: Daiane Cristina Zanatta. Email: daiane@unicentro.br
Received: 07 July 2020; Accepted: 02 September 2020

Abstract: This paper seeks to develop an efficient multigrid algorithm for solving the Burgers problem with the use of non-orthogonal structured curvilinear grids in L-shaped geometry. For this, the differential equations were discretized by Finite Volume Method (FVM) with second-order approximation scheme and deferred correction. Moreover, the algebraic method and the differential method were used to generate the non-orthogonal structured curvilinear grids. Furthermore, the influence of some parameters of geometric multigrid method, as well as lexicographical Gauss–Seidel (Lex-GS), η-line Gauss–Seidel (η-line-GS), Modified Strongly Implicit (MSI) and modified incomplete LU decomposition (MILU) solvers on the Central Processing Unit (CPU) time was investigated. Therefore, several parameters of multigrid method and solvers were tested for the problem, with the use of non-orthogonal structured curvilinear grids and multigrid method, resulting in an algorithm with the combination that achieved the best results and CPU time. The geometric multigrid method with Full Approximation Scheme (FAS), V-cycle and standard coarsening ratio for this problem were utilized. This article shows how to calculate the coordinates transformation metrics in the coarser grids. Results show that the MSI and MILU solvers are the most efficient. Moreover, the MSI solver is faster than MILU for both grids generators; and the solutions are more accurate for the Burgers problem with grids generated using elliptic equations.

Keywords: Computational fluid dynamics; finite volume method; Burgers' equation; geometric multigrid; non-orthogonal curvilinear grids

1 Introduction

Many Engineering problems concern complex geometries, in which the use of Cartesian, cylindrical or spherical coordinate systems is not practical or appropriate, and discretization on structured curvilinear or unstructured grids is preferred.

Structured curvilinear grids are based on the mapping of the physical domain over a simple shape computational domain, for instance, a rectangle [1]. The use of such grids simplifies the application of boundary conditions, since the conditions existing on the boundaries of the physical plane are transferred exactly to the boundaries of the modified plane, and thus do not need approximation. With this option, a system of global equations is achieved and it is possible to write and discretize differential equations in this new system.

The discretization of mathematical models by the Finite Volume Method (FVM) [2] approximates a system of differential equations through a system of algebraic equations of the form

$$A\phi = \mathbf{b}, \tag{1}$$

where A represents the matrix of coefficients, ϕ is the vector of unknowns and $\mathbf{b}$ is the vector of independent terms. The FVM can be applied to any type of grid, in complex geometries and in different coordinate systems [3]. The attractive feature of this method is that the method satisfies the integral conservation of the quantities, such as mass, amount of linear movement and energy for any group of control volumes and, consequently, for the entire calculation domain [2].

The numerical solution methods of the system of algebraic equations given by Eq. (1) can be classified as direct or iterative [4]. Since the generated system of algebraic equation is sparse and large-scale, direct methods are not suitable due to their high computational cost. For this reason, we chose to work with iterative methods (solvers). Solvers are efficient in relaxing high-frequency (oscillatory) components during the first few iterations in refined grids. However, after some iterations, the process becomes slow, what means there is a predominance of low-frequency (smooth) modes [5,6].

Multigrid is an efficient method to accelerate solvers' convergence rate [5–7]. This method sweeps, during the iterative process, several grids of different refinement levels. According to Trottenberg et al. [6], smooth modes of the error become more oscillatory in coarser grids. Because of this, the different components of the error are efficiently smoothed, what accelerates the convergence of the iterative process used to solve the system given by Eq. (1).

Several studies concern the use of the multigrid method (MG) in problems involving a system of curvilinear coordinates. Smith et al. [8] presented a MG algorithm in order to accelerate a three-dimensional Navier–Stokes equations on general curvilinear grids. The authors calculated the metrics of the coarser grid while restricting the metrics of the adjacent finer grid. Moreover, they presented the convergence rate, computational times and performance indexes of the MG for several Reynolds numbers in highly curved ducts, with Full Approximation Scheme (FAS) combined with Full Multigrid (FMG).

Oosterlee et al. [9] found benchmark solutions for Navier–Stokes equations for incompressible flows, in steady state, in L-shaped domain, with systems of general curvilinear coordinates. The discretized system was solved using the MG with FAS, F-cycle, number of relaxations performed at the restriction and prolongation step (pre- and post-smoothing) equals to 1 and maximum number of levels.

Trottenberg et al. [6] stated that the choice of the components of the MG, such as solver, number of relaxations and type of cycle as well as restriction and prolongation operators, can strongly influence the convergence rate of the algorithm. According to Ferziger et al. [3], in the general context of the MG, many parameters can be chosen more or less arbitrarily, and the convergence rate and CPU time depend on these choices.

Li et al. [10] studied Navier–Stokes equations for two-dimensional flow of incompressible fluid as well as heat transfer in parallelogrammic, trapezoidal and sine-shaped cavities. The authors used FVM to discretize equations with collocated grid arrangement of the variables to solve physical problems in grids with 1024×1024 volumes. The system of algebraic equations was solved using the MG with FAS, V-cycle, and Strongly Implicit Procedure (SIP) solver as well as the SIMPLE method to calculate the pressure-velocity coupling. They presented numerical results for the problem of lid-driven cavity in parallelogrammic cavity and Reynolds numbers of 100 and 5000, more accurate than the solutions presented by [11]. Moreover, they presented benchmark solutions for natural and mixed-convection problems.

Two-dimensional Burgers' equations are regarded as one of the important models in the study of fluid flows. These equations have many applications, which go from cosmology to traffic modeling [12]. Ferm et al. [13], Zhang et al. [14], Neveu et al. [15] and Santiago et al. [16] studied the problem of Burgers' equations with the aim of improving the convergence of the MG. All of the authors used orthogonal structured grids. Among the studies found on MG applied to Burgers' equations, only a few aim at improving the convergence of the method by means of reducing the CPU time, and even less are systematic studies of the parameters of the MG.

In this paper, our goal is to systematically analyze some parameters of the geometric MG aimed at developing an efficient code for the problem of two-dimensional Burgers' equations, using systems of general curvilinear coordinates. In order to generate the non-orthogonal structured curvilinear grids, algebraic and differential methods were used [17]. Equations were discretized by FVM with Central Difference Scheme (CDS). The system of algebraic equations was solved with the solvers Lex-GS, η-line-GS, MILU and MSI. The metrics of the transformation and of the residual were restricted in order to solve the equations with curvilinear grids [8].

This paper is organized as follow: Section 2 presents the mathematical model in Cartesian and curvilinear coordinates; the generation of the coordinate system is shown in Section 3; Section 4 details the numerical model; Section 5 depicts the resolution methods for the system of linear equations; the MG is presented in Section 6; Section 7 presents the calculation of transformation metrics; Section 8 shows the results and discussion; and, finally, the conclusion is presented in Section 9.

2 Mathematical Model

Considering constant properties, steady state and Cartesian coordinates, the two-dimensional Burgers' equations are written dimensionless as

$$\frac{\partial u^2}{\partial x} + \frac{\partial uv}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{\mathrm{Re}}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{2}$$

and

$$\frac{\partial uv}{\partial x} + \frac{\partial v^2}{\partial y} = -\frac{\partial p}{\partial y} + \frac{1}{\mathrm{Re}}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) - S\left(x, y, \mathrm{Re}\right), \tag{3}$$

where p stands as the static pressure, u and v are the velocity components on the coordinate directions x and y, respectively; and Re is the Reynolds number.

The source term $S(x,\ y,\ \mathrm{Re})$ and the field pressure p are, respectively,

$$S(x,y,\mathrm{Re}=1)=\frac{1}{2}f^2(x)\left[g(y)g'''(y)-g'(y)g''(y)\right]+\left[f'(x)\right]^2 g(y)g'(y)-f''(x)f(x)g(y)g'(y)$$
$$+2f'(x)g''(y)+g^{iv}(y)F(x)+f'''(x)g(y),$$

and

$$p(x,y,\mathrm{Re}=1)=-\frac{1}{2}f^2(x)\left[g'(y)\right]^2+\frac{1}{2}f^2(x)g(y)g''(y)+f'(x)g'(y)+F(x)g'''(y),$$

where $f(x)=x^5-\frac{5}{2}x^4+\frac{35}{16}x^3-\frac{25}{32}x^2+\frac{3}{32}x$, $g(y)=y^5-\frac{5}{2}y^4+\frac{35}{16}y^3-\frac{25}{32}y^2+\frac{3}{32}y$ and $F(x)=\int f(x)dx$.

In this study, we consider $\mathrm{Re}=1$. The analytical solutions, obtained by manufactured solution [18], are given by the expressions $u(x,y)=f(x)\ g'(y)$ and $v(x,y)=-f'(x)g(y)$. Thus

$$u(x,y)=\left(x^5-\frac{5}{2}x^4+\frac{35}{16}x^3-\frac{25}{32}x^2+\frac{3}{32}x\right)\left(5y^4-10y^3+\frac{105}{16}y^2-\frac{25}{16}y+\frac{3}{32}\right) \tag{4}$$

and

$$v(x,y)=\left(-5x^4+10x^3-\frac{105}{16}x^2+\frac{25}{16}x-\frac{3}{32}\right)\left(y^5-\frac{5}{2}y^4+\frac{35}{16}y^3-\frac{25}{32}y^2+\frac{3}{32}y\right). \tag{5}$$

The computational domain of this problem, depicted in Fig. 1, is defined by

$$\begin{cases}0\le x\le 1, & \text{if}\quad 0\le y\le 0.5\\ 0\le x\le 0.5, & \text{if}\quad 0.5<y\le 1\end{cases}\quad\text{and}\quad\begin{cases}0\le y\le 1, & \text{if}\quad 0\le x\le 0.5\\ 0\le y\le 0.5, & \text{if}\quad 0.5<x\le 1.\end{cases}$$

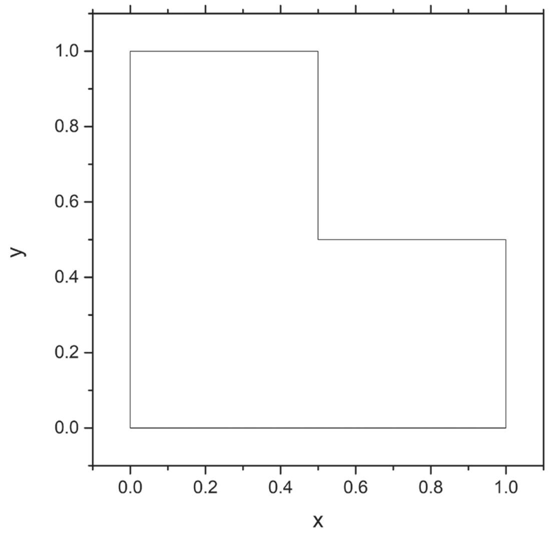


Figure 1: Computational domain

Dirichlet boundary conditions are obtained by replacing the boundary locations in the expressions given by Eqs. (4) and (5).

In general, Eqs. (2) and (3) can be written as

$$\frac{\partial}{\partial x}(u\phi)+\frac{\partial}{\partial y}(v\phi)=p^{\phi}+\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}-S^{\phi}, \tag{6}$$

where ϕ (being u or v), $p^u=-\frac{\partial p}{\partial x}$ and $p^v=-\frac{\partial p}{\partial y}$.

In order to transform Eq. (6), written on the physical domain (x,y), to the computational domain (ξ,η), the transformation of coordinates that is needed for two-dimensional problems is given by

$$\begin{cases}\xi=\xi(x,y)\\ \eta=\eta(x,y)\end{cases}.$$

The metrics of the transformation are given by

$$\xi_x=y_\eta J,\quad \eta_x=-y_\xi J,\quad \xi_y=-x_\eta J,\quad \eta_y=x_\xi J$$

and the jacobian is given by

$$J=\left(x_\xi y_\eta-y_\xi x_\eta\right)^{-1}.$$

Following the chain rule, the derivatives of the advective and diffusive flow of the generic variable φ and the terms of pressure, which appear in Eq. (6), result in

$$\frac{\partial}{\partial x}(u\phi)+\frac{\partial}{\partial y}(v\phi)=J\left[(u\phi)_\xi y_\eta-(u\phi)_\eta y_\xi-(v\phi)_\xi x_\eta+(v\phi)_\eta x_\xi\right],$$

$$\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}=J\left\{J\left[\alpha\phi_\xi-\beta\phi_\eta\right]\right\}_\xi+J\left\{J\left[\gamma\phi_\eta-\beta\phi_\xi\right]\right\}_\eta,$$

$-\frac{\partial p}{\partial x}=J\left[-\frac{\partial}{\partial \xi}\left(py_\eta\right)+\frac{\partial}{\partial \eta}\left(py_\xi\right)\right]$ and $-\frac{\partial p}{\partial y}=J\left[\frac{\partial}{\partial \xi}\left(px_\eta\right)-\frac{\partial}{\partial \eta}\left(px_\xi\right)\right]$, where $\alpha=x_\eta^2+y_\eta^2$, $\beta=x_\xi x_\eta+y_\xi y_\eta$ and $\gamma=x_\xi^2+y_\xi^2$ are the components of the metric tensor in two dimensions.

After defining the following variables $U=uy_\eta-vx_\eta$ and $V=vx_\xi-uy_\xi$, the advective flow can be rewritten as

$$\frac{\partial}{\partial x}(u\phi)+\frac{\partial}{\partial y}(v\phi)=(U\phi)_\xi+(V\phi)_\eta,$$

where U and V are the contravariant components of the velocity vector.

By grouping the transformed terms, we obtain

$$\frac{\partial}{\partial \xi}(U\phi)+\frac{\partial}{\partial \eta}(V\phi)=\frac{p^{\phi}}{J}+\frac{\partial}{\partial \xi}\left[J\left(\alpha\frac{\partial\phi}{\partial\xi}-\beta\frac{\partial\phi}{\partial\eta}\right)\right]+\frac{\partial}{\partial \eta}\left[J\left(\gamma\frac{\partial\phi}{\partial\eta}-\beta\frac{\partial\phi}{\partial\xi}\right)\right]-\frac{S^{\phi}}{J}. \tag{7}$$

Eq. (7) is the general equation transformed for a scalar ϕ.

3 Generation of the Coordinate System

The methods for grid generation are basically divided into algebraic and differential methods [17,19]. In this paper, we used an algebraic method that utilizes Lagrange interpolation and a differential method that uses the solution of elliptic equations for grid generation.

Fig. 2 presents a grid generated by means of Lagrange interpolation and another grid obtained by solving the system of elliptic differential equations.

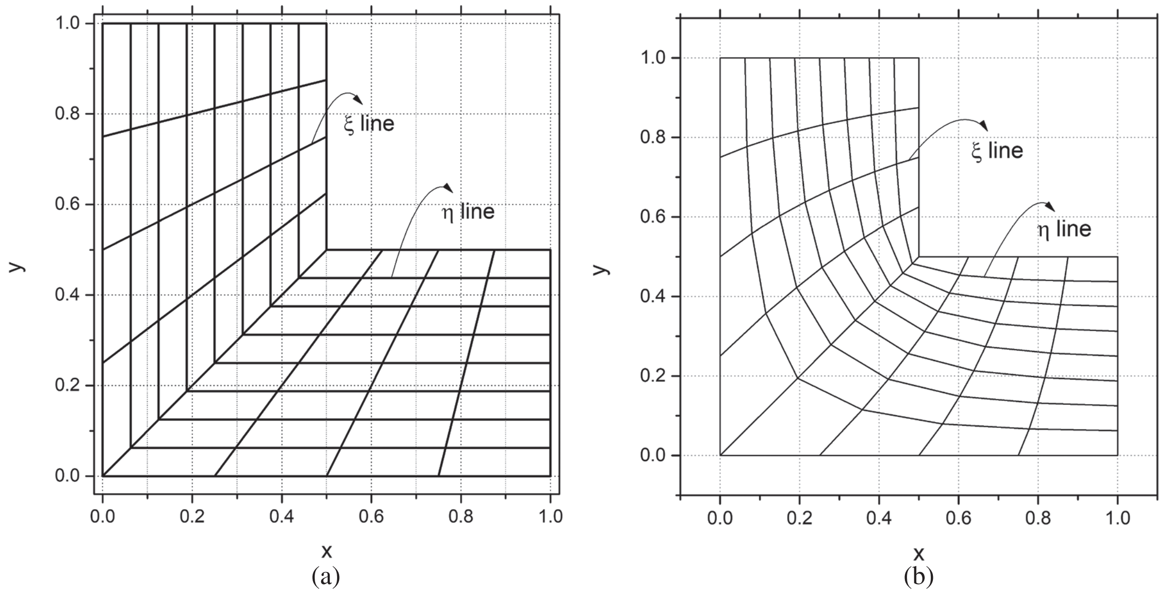


Figure 2: Grids generated by means of Lagrange interpolation and elliptic equations with 8 volumes in the directions ξ and η. (a) Lagrange interpolation. (b) Elliptic equations

4 Numerical Model

By integrating Eq. (7) over the control volume P in the transformed plane, as seen in Fig. 3a, results in

$$U_e\phi_e\Delta\eta - U_w\phi_w\Delta\eta + V_n\phi_n\Delta\xi - V_s\phi_s\Delta\xi = J_e\left\{\alpha_e\left[\frac{\partial\phi}{\partial\xi}\right]_e - \beta_e\left[\frac{\partial\phi}{\partial\eta}\right]_e\right\}\Delta\eta$$
$$-J_w\left\{\alpha_w\left[\frac{\partial\phi}{\partial\xi}\right]_w - \beta_w\left[\frac{\partial\phi}{\partial\eta}\right]_w\right\}\Delta\eta + J_n\left\{\gamma_n\left[\frac{\partial\phi}{\partial\eta}\right]_n - \beta_n\left[\frac{\partial\phi}{\partial\xi}\right]_n\right\}\Delta\xi$$
$$-J_s\left\{\gamma_s\left[\frac{\partial\phi}{\partial\eta}\right]_s - \beta_n\left[\frac{\partial\phi}{\partial\xi}\right]_s\right\}\Delta\xi - L\left[\frac{p^\phi}{J}\right]_P - \frac{S^\phi}{J}\Delta\xi\Delta\eta, \quad (8)$$

where $L\left[\frac{p^\phi}{J}\right]_P$ is the approximation of the integral in ξ and η of p^φ in the volume P, and w, e, s and n are the west, east, south and north faces of the volume P, respectively.

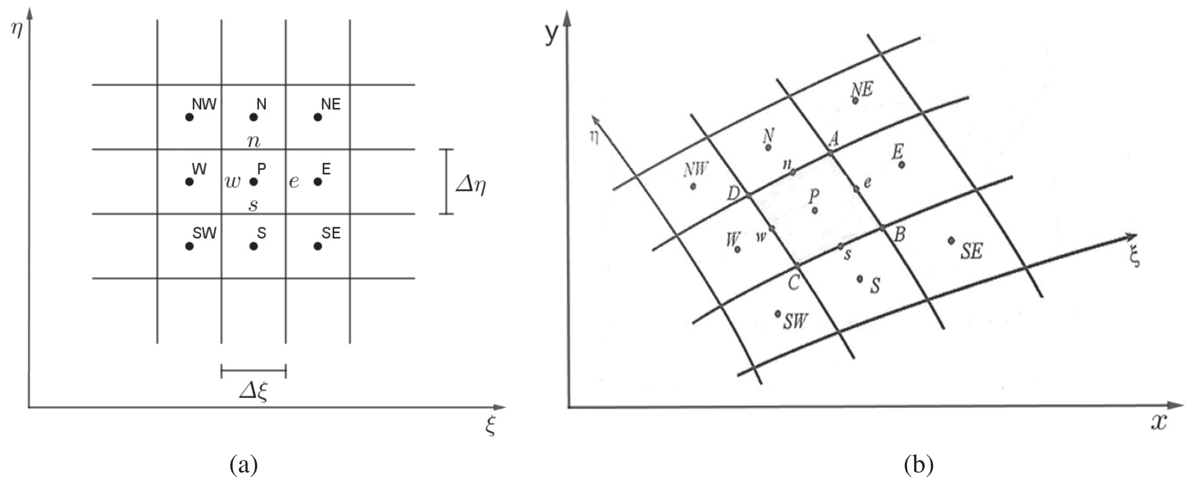


Figure 3: Control volume and neighbors. (a) In the transformed plane. (b) In the physical domain

By approximating the derivatives of the right side of Eq. (8) by CDS and the advective terms (left side of Eq. (8)) by Upstream Difference Scheme (UDS), we have

$$U_e\left[\left(\frac{1}{2}+\lambda_e\right)\phi_P+\left(\frac{1}{2}-\lambda_e\right)\phi_E\right]\Delta\eta-U_w\left[\left(\frac{1}{2}+\lambda_w\right)\phi_W+\left(\frac{1}{2}-\lambda_w\right)\phi_P\right]\Delta\eta$$

$$+V_n\left[\left(\frac{1}{2}+\lambda_n\right)\phi_P+\left(\frac{1}{2}-\lambda_n\right)\phi_N\right]\Delta\xi-V_s\left[\left(\frac{1}{2}+\lambda_s\right)\phi_S+\left(\frac{1}{2}-\lambda_s\right)\phi_P\right]\Delta\xi$$

$$=J_e\left[\alpha_e\frac{\phi_E-\phi_P}{\Delta\xi}-\beta_e\frac{\phi_N+\phi_{NE}-\phi_S-\phi_{SE}}{4\Delta\eta}\right]\Delta\eta-J_w\left[\alpha_w\frac{\phi_P-\phi_W}{\Delta\xi}-\beta_w\frac{\phi_N+\phi_{NW}-\phi_S-\phi_{SW}}{4\Delta\eta}\right]\Delta\eta$$

$$+J_n\left[\gamma_n\frac{\phi_N-\phi_P}{\Delta\eta}-\beta_n\frac{\phi_E+\phi_{NE}-\phi_W-\phi_{NW}}{4\Delta\xi}\right]\Delta\xi-J_s\left[\gamma_s\frac{\phi_P-\phi_S}{\Delta\eta}-\beta_n\frac{\phi_E+\phi_{NE}-\phi_W-\phi_{NW}}{4\Delta\xi}\right]\Delta\xi$$

$$+L\left[\frac{p^\phi}{J}\right]_P-\frac{S_P}{J}\Delta\xi\Delta\eta,$$

where $\lambda_e=\frac{1}{2}\text{sign}\,(U_e)$, $\lambda_w=\frac{1}{2}\text{sign}\,(U_w)$, $\lambda_n=\frac{1}{2}\text{sign}\,(V_n)$ and $\lambda_s=\frac{1}{2}\text{sign}\,(V_s)$, where sign stand as the sign function. This expression can be rewritten as

$$a_p\phi_p=a_w\phi_W+a_e\phi_E+a_s\phi_S+a_n\phi_N+a_{sw}\phi_{SW}+a_{se}\phi_{SE}+a_{nw}\phi_{NW}+a_{ne}\phi_{NE}+b_p=\sum_{nb}a_{nb}\phi_{NB}+b_P, \quad (9)$$

where NB stands as the 8 neighboring volumes of P.

In order to transform the conservation equations to the domain (ξ, $\boldsymbol{\eta}$), it is necessary to know the transformation metrics, that is, the variables $\left(x_\eta\right)_P^e$, $\left(y_\eta\right)_P^e$, $\left(x_\xi\right)_P^n$ and $\left(y_\xi\right)_P^n$, for the two-dimensional case. To make the notation easier, the face is used as superscript and the reference point as subscript. Fig. 3b shows a basic volume P, with points A, B, C and D in the respective vertex.

With these variables, it is possible to calculate the jacobian of the transformation. The metrics are numerically calculated by

$$(x_\eta)_P^e = \frac{x_A - x_B}{\Delta\eta}, \quad (y_\eta)_P^e = \frac{y_A - y_B}{\Delta\eta}, \quad (x_\xi)_P^n = \frac{x_A - x_D}{\Delta\xi} \quad \text{and} \quad (y_\xi)_P^n = \frac{y_A - y_D}{\Delta\xi}.$$

Thus, $\alpha_P^e = \left[(x_\eta)_P^e\right]^2 + \left[(y_\eta)_P^e\right]^2$, $\gamma_P^n = \left[(x_\xi)_P^n\right]^2 + \left[(y_\xi)_P^n\right]^2$ and $\beta_P^e = (x_\xi)_P^e(x_\eta)_P^e + (y_\xi)_P^e(y_\eta)_P^e$, with $(\psi_\xi)_P^e = \frac{1}{4}\left[(\psi_\xi)_P^n + (\psi_\xi)_E^n + (\psi_\xi)_S^n + (\psi_\xi)_{SE}^n\right]$, where $\psi = x$ or y, and $\beta_P^n = (x_\xi)_P^n(x_\eta)_P^n + (y_\xi)_P^n(y_\eta)_P^n$, with $(\psi_\eta)_P^n = \frac{1}{4}\left[(\psi_\eta)_P^e + (\psi_\eta)_N^e + (\psi_\eta)_W^e + (\psi_\eta)_{NW}^e\right]$.

Therefore, the jacobian is calculated as $J_P^c = \left[(x_\xi)_P^c(y_\eta)_P^c - (x_\eta)_P^c + (y_\xi)_P^c\right]^{-1}$, where c is the center of the control volume and $(\psi_\xi)_P^c = \frac{1}{2}\left[(\psi_\xi)_P^n + (\psi_\xi)_S^n\right]$ and $(\psi_\eta)_P^c = \frac{1}{2}\left[(\psi_\eta)_P^e + (\psi_\eta)_W^e\right]$. Moreover, we obtain $(J)_P^e = \frac{1}{2}\left[(J)_P^c + (J)_E^c\right]$ and $(J)_P^n = \frac{1}{2}\left[(J)_P^c + (J)_N^c\right]$.

5 Iterative Methods and Multigrid Details

A solution for Eq. (9) can be obtained by iterative method. In this section we will briefly describe those used in this article.

5.1 Gauss–Seidel Method and Its Variations

If the diagonal entries of A are non-zeros, the unknown corresponding value can be isolated in each equation, resulting in the lexicographic Gauss–Seidel method (Lex-GS) [4,20] and its variations.

It can be classified as a point-wise or block solver. In point-wise methods, each variable is updated individually. Fig. 4a depicts the lexicographic order, which was used in this study. By combining the Gauss–Seidel method with this order, we have the Lex-GS.

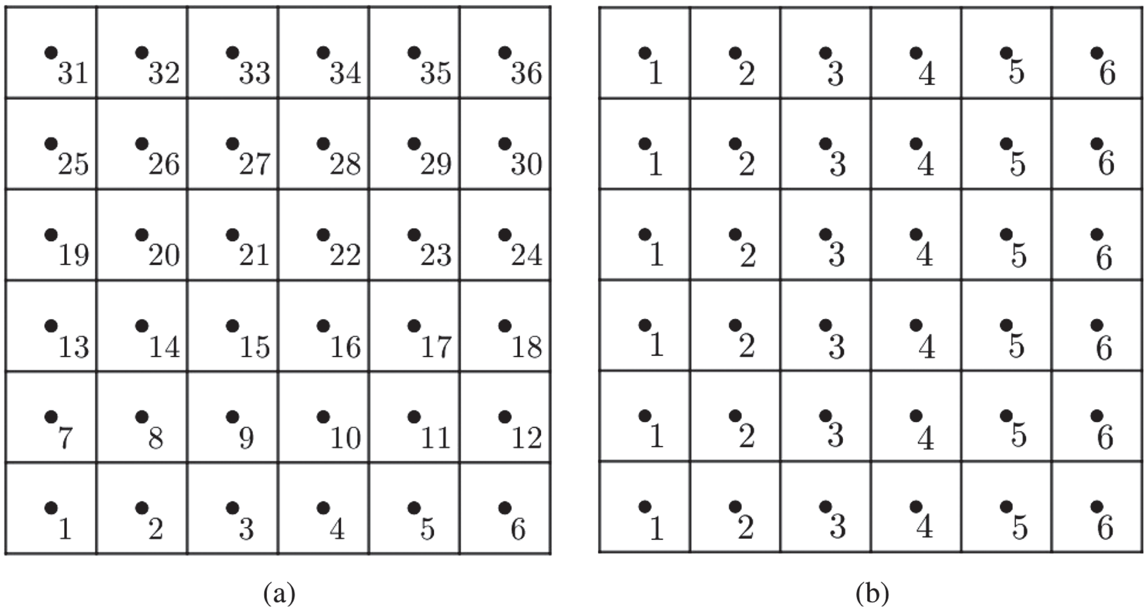


Figure 4: Lexicographic ordering and η-line ordering in a two-dimensional grid. (a) Lexicographic order. (b) Line order in the direction $\boldsymbol{\eta}$

For Eq. (9), Lex-GS has the form of

$$\phi_P^{(m+1)} = \left(\sum_i a_i \phi_i^{(m+1)} + \sum_j a_j \phi_j^{(m)} + b_p \right) \Big/ a_P,$$

where the superscript m represents the mth iteration, the subscript the position on the grid, $i = W, S, SW, SE$ and $j = E, N, NW, NE$. Notice that the nine closest neighbors were required in order to approximate φ_p.

On block methods (columns or lines, for instance), each block is updated at once. Fig. 4b shows grid points ordered in lines in the direction η. By using the Gauss–Seidel method with this order, we have the η-line Gauss–Seidel (η-line-GS), which was used in this study.

5.2 Modified Incomplete LU Decomposition

Incomplete LU decomposition (ILU) of the A [20,21] consists in decomposing a matrix A in an incomplete manner. Such decomposition is of the $A = LU - R$ form, where, in this subsection, L and U represent lower and upper triangular matrices and R represents the residue or decomposition error.

By doing the nine-point ILU decomposition of the coefficient matrix given by Eq. (9), we obtain L and U matrices with the same sparsity of A [20]. This case is called ILU(0). Fig. 5 presents the scheme $A = LU - R$ for ILU(0). Matrix R contains the four additional non-null diagonals (continuous lines), in which the hatched lines show the original position of the non-null diagonals of A.

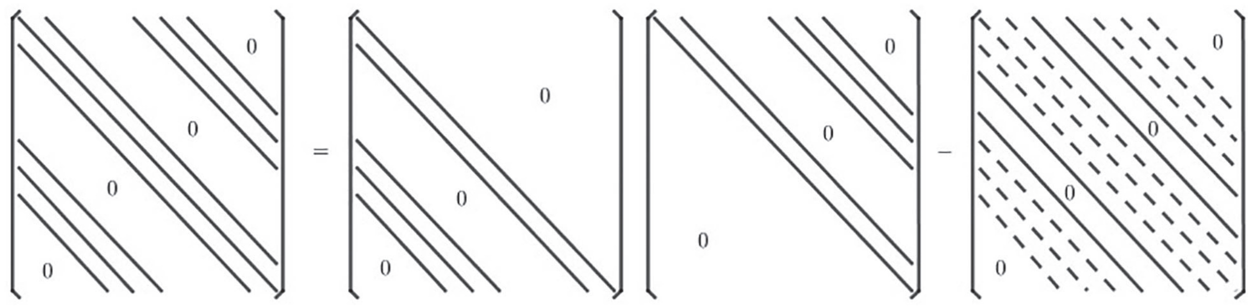


Figure 5: ILU(0) factorization for a 9-diagonal matrix

The main diagonal of R and the elements u_{kk} of the matrix U can be modified in such a way that

$$r_{kk} \leftarrow \sigma \sum_{j \neq k} |r_{kj}| \text{ and } u_{kk} \leftarrow u_{kk} + r_{kk}.$$

This method is named Modified ILU decomposition (MILU) [6,20,22].

Given the MILU decomposition, the system given by Eq. (9) is solved by the iterative process

(1) Solve $Ly^{(m)} = \mathbf{r}^{(m)}$ to obtain $y^{(m)}$, where $\mathbf{r}^{(m)}$ is the residual vector;
(2) Solve $U\boldsymbol{e}^{(m)} = \mathbf{y}^{(m)}$ to obtain $\boldsymbol{e}^{(m)}$.

Thus, the solution in the iteration $m+1$ is given by $\phi^{(m+1)} = \phi^{(m)} + \boldsymbol{e}^{(m)}$.

5.3 Modified Strongly Implicit

In the MSI method [16,23], a LU decomposition is proposed, such that, $M = LU$, where the matrices L and U are lower and upper triangular, respectively, and the main diagonal of U is unitary. By performing the LU decomposition, in M four diagonals are filled. These non-null additional diagonals, are denoted by $\phi^1_{i,j}$, $\phi^2_{i,j}$, $\phi^3_{i,j}$ and $\phi^4_{i,j}$. Therefore, the matrix M can be written as $M = A + N$, where A is the coefficient matrix of the equation system given by Eq. (9) and N consists only of diagonals $\phi^1_{i,j}$, $\phi^2_{i,j}$, $\phi^3_{i,j}$ and $\phi^4_{i,j}$.

A parameter σ was employed to partly cancel the influence of the additional terms that appear in M. By decomposing matrix A, the system is solved using the same iterative process shown in the Subsection 5.2.

6 Multigrid Details

Basic iterative methods are efficient in relaxing high-frequency components in refined grids during the first few iterations. However, after some iterations, the process becomes slow, what signals the predominance of low-frequency modes [5]. At this moment, it is recommended to transfer the information to the immediately coarser grid, where smooth error modes become more oscillatory and relaxation will be more efficient [6,22]. This happens because when the multigrid method (MG) is associated with an iterative method, it relaxes the error and corrects the solution in different grid sizes.

The MG sweeps a group of grids with different spacings. By means of a solver, iterations are performed at each level of the grid until the specified convergence criterion is reached for the system of equations of the finest grid. The sequence through which the grids are swept is denominated cycle. The coarsening ratio (r) of the grids is defined as $r = H/h$, where h is the size of the volumes of the finer grid and H represents the size of the volumes of the immediately coarser grid. The number of grids employed is called the number of levels (L). In case the highest possible number of grid levels is used, it will be denoted as L_{max}.

The multigrid method can be implemented with Correction Scheme (CS), more suitable for linear problems, or Full Approximation Scheme (FAS), more indicated for non-linear problems [5]. In the FAS, the residual and the approximation of the solution are transferred to the coarser grids [6].

Operators that transfer information from a fine grid, Ω^h, to an immediately coarser grid, Ω^{2h}, are called restriction operators (I^{2h}_h), the reverse are called prolongation operators (I^h_{2h}). The number of iteration used in the restriction and prolongation steps are called pre- and post-smoothing (ν_1 and ν_2, respectively).

7 Calculation of Coordinates Transformation Metrics in Coarser Grids

It is also necessary to restrict the transformation metrics for the solution of the equations in curvilinear grids. In this paper, the convergence of the MG was achieved by calculating the coordinate transformation metrics as $\left(\psi_\eta\right)^e_F = \left(\psi_\eta\right)^e_{16}$ and $\left(\psi_\xi\right)^n_F = \left(\psi_\xi\right)^n_{21}$, where $\psi = x$ or y, F is a coarse grid point and 16 and 21 are fine grid points, as seen in Fig. 6.

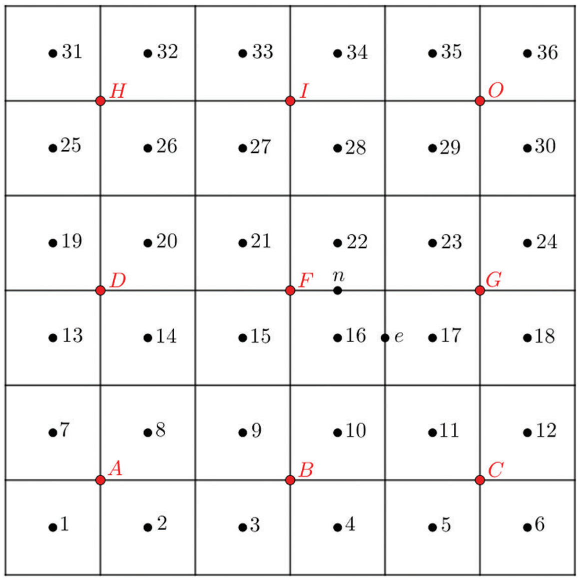


Figure 6: Grids scheme in the computational domain to restrict the coordinates transformation metrics of fine grid (points $1, 2, 3, \ldots, 36$) to the immediately coarser grid (points A, B, C, D, F, G, H, I and O)

The metric tensor components in the coarse grid are calculated using the following expressions $\alpha_F^e = \left[(x_\eta)_F^e\right]^2 + \left[(y_\eta)_F^e\right]^2$, $\gamma_F^n = \left[(x_\xi)_F^n\right]^2 + \left[(y_\xi)_F^n\right]^2$ and $\beta_F^e = (x_\xi)_F^e (x_\eta)_F^e + (y_\xi)_F^e (y_\eta)_F^e$ with

$$(\psi_\xi)_F^e = \frac{1}{4}\left[(\psi_\xi)_F^n + (\psi_\xi)_G^n + (\psi_\xi)_B^n + (\psi_\xi)_C^n\right] \text{ and } (\psi_\eta)_F^n = \frac{1}{4}\left[(\psi_\eta)_F^e + (\psi_\eta)_I^e + (\psi_\eta)_D^e + (\psi_\eta)_H^e\right],$$

where G, B, C, I, D and H coarse grid points.

The jacobian of the transformation on the coarse grid is calculated as

$$J_F^c = \left[(x_\xi)_F^c (y_\eta)_F^c - (x_\eta)_F^c + (y_\xi)_F^c\right]^{-1},$$

where c is the center of the control volume, $(\psi_\xi)_F^c = \frac{1}{2}\left[(\psi_\xi)_F^n + (\psi_\xi)_B^n\right]$ and $(\psi_\eta)_F^c = \frac{1}{2}\left[(\psi_\eta)_F^e + (\psi_\eta)_D^e\right]$. Moreover, $(J)_F^e = \frac{1}{2}\left[(J)_F^c + (J)_G^c\right]$ and $(J)_F^n = \frac{1}{2}\left[(J)_F^c + (J)_I^c\right]$.

8 Results and Discussion

8.1 Implementation Data

The algorithm was implemented in double precision FORTRAN 95 Intel 11.1 compiler. Simulations were performed in a 3.4 GHz Intel(R) Core(TM) i7-3770 microcomputer, 16 GB RAM, 64-bits Windows XP.

Geometric MG with FAS and V-cycle was used to solve the system of algebraic equations represented by Eq. (9) for the domain depicted in Fig. 1. The restriction operator of the approximations of the solution was obtained by arithmetic mean of the property values of the four volumes of the fine grid [6]. The restriction of the residual was done by adding the relative residuals of the control volumes of the fine grid that correspond to those of the coarser grid [24]. The correction prolongation was done by bilinear interpolation [5,6]. In this study, standard grid coarsening ratio was used ($r = 2$). Lex-GS, η-line-GS, MSI and MILU were used as solvers. The number of iterations in the pre- and post smoothing is the same, that is, $\nu = \nu_1 = \nu_2$. The number of unknowns is given by $N = N_\xi N_\eta$, where N_ξ and N_η stand for the number of volumes in the directions ξ and η, respectively. The stop criterion used is the l_1 norm of the residual in the current iteration $r^{(m)}$, non-dimensionalized by the residual in the initial estimate, $\mathrm{r}^{(0)}$, that is $||\mathbf{r}^{(m)}||_1/||\mathbf{r}^{(0)}||_1 \leq \mathrm{tol}$, where $\mathrm{tol} = 10^{-11}$ is adopted.

8.2 *Average Convergence Factor*

The empirical average convergence factor, which approximates the asymptotic convergence factor, is computed based on the residual. Such factor, as described in Trottenberg et al. [6], is given by

$$\rho_m = \sqrt[m]{\frac{||\mathbf{r}^{(m)}||}{||\mathbf{r}^{(0)}||}}, \tag{10}$$

where m represents the number of iterates or multigrid cycles performed.

In order to reduce the effect caused by the discard of elements during the ILU decomposition, a study of ρ_m for different values of σ, with $0 \leq \sigma \leq 0.95$ and $-0.25 \leq \sigma \leq 1$, for the MSI and MILU solvers, respectively, was carried out. For this, we considered $L = L_{max}$ and the number of inner iterations $\nu_1 = \nu_2 = 3$, for both solvers.

Figs. 7a and 7b depict the MSI method for the results of ρ_m *vs.* σ for the variables u and v and grids generated by Lagrange interpolation and elliptic equations. Figs. 7c and 7d depict the same to MILU. Notice that the smallest values of ρ_m, regardless of the size of the problem, happen when $\sigma \approx 0.9$ and $\sigma \approx -0.2$ for MSI and MILU solvers, respectively. Therefore, from now on, these values will be employed for σ in the formulations of MSI and MILU.

Fig. 8 shows the empirical average convergence factor for all solvers assessed. The presented results are for different grid sizes generated by Lagrange interpolation and elliptic equations. As shown in Fig. 8, the η-line-GS method present the better empirical average convergence factor when compared with the Lex-GS method. This happened as the equations involve a strong coupling of the unknowns in the direction η.

As Trottenberg et al. [6] stated, stretched volumes in the physical domain cause anisotropy in the equations. In anisotropic problems, the convergence of basic methods, such as pointwise solvers, decreases [9,25,26]. The MILU and MSI solvers present the best empirical average convergence factors, that is, present $\rho_m \sim 0$ and $\rho_m \ll 1$, which is a desirable property. For instance, in the $N = 4096 \times 4096$ grid, $\rho_m \approx 0.03$. Based on these results and aimed to develop an efficient algorithm to solve the problem in question, we will use the MILU and MSI solvers in our algorithm hereinafter as they have the best average convergence factors.

8.2.1 *Inner Iterations*

The effect of the number of inner iterations, denoted by $\nu = \nu_1 = \nu_2$, was assessed for grids sizes $N = 512 \times 512$, 1024×1024, 2048×2048 and 4096×4096, using the maximum number of levels ($L = L_{max}$) in every case. Fig. 9 illustrate the influence of the number of inner iterations in the sum of the CPU times (t_{CPU}) in determining the velocities u and v, using the MSI and MILU solvers.

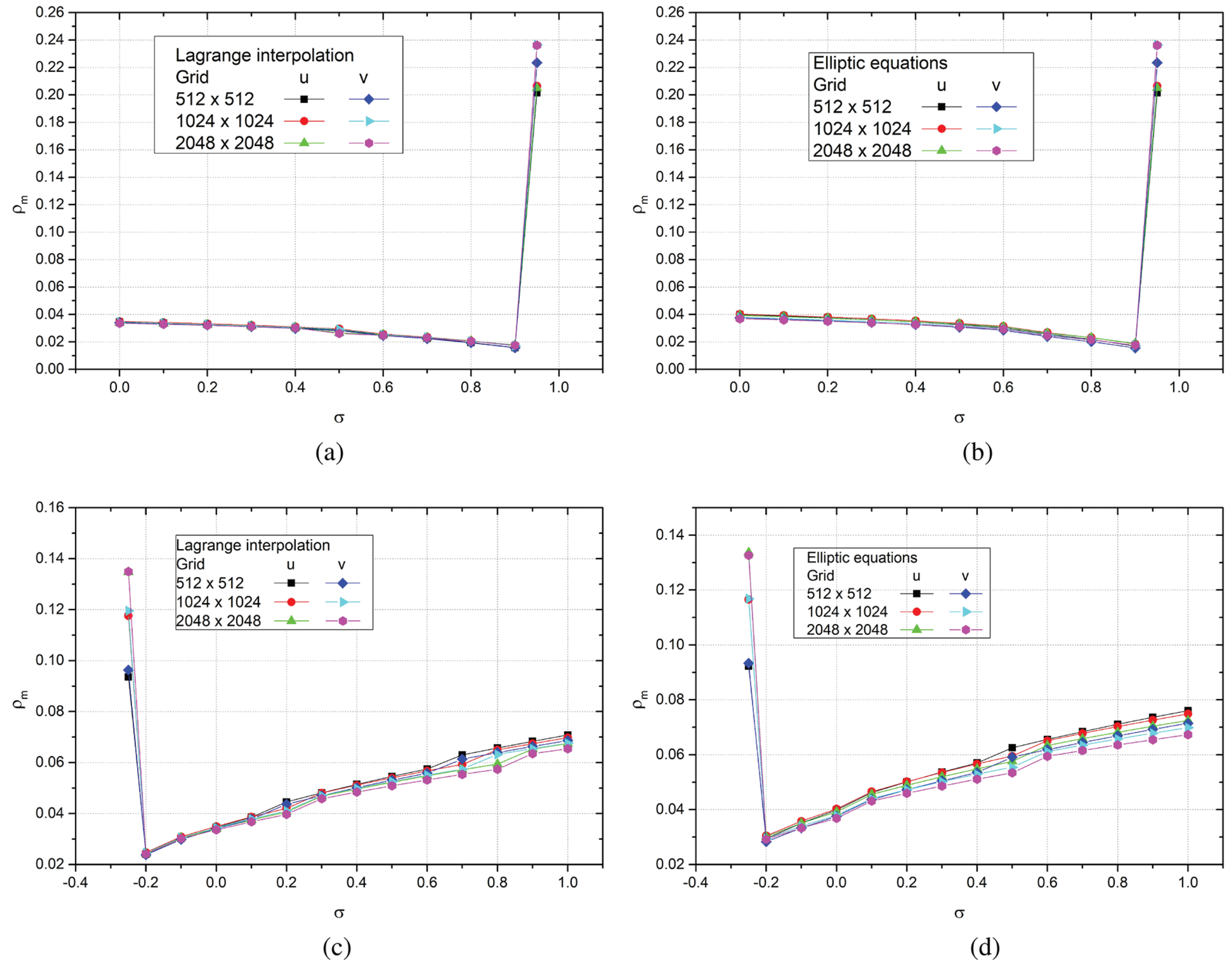


Figure 7: ρ_m *vs.* σ for the MSI and MILU solvers. (a) MSI, Lagrange interpolation. (b) MSI, Elliptic equation. (c) MILU, Lagrange interpolation. (d) MILU, Elliptic equation

In each curve, the value of ν that results in the lowest CPU time is indicated by the symbol □. Noticeably, the lowest CPU time, for the algorithm built with the MSI solver, was obtained with four iterations, except in the grid $N = 4096 \times 4096$, generated by Lagrange interpolation, and grid $N = 2048 \times 2048$, generated by elliptic equations. Calculating the weighted average in CPU time gains for $\nu = 2$ and $\nu = 4$, the number of inner iterations of the solver that obtained the best average performance was $\nu = 4$, for both grids generators. Thus, hereinafter, for the algorithm built with the MSI solver, the ν value adopted will be $\nu = 4$.

For the algorithm built using MILU as solver, the lowest CPU time was obtained with three iterations for any of the values of N used, for grids generated by means of Lagrange interpolation. For grids generated by means of elliptic equations, the lowest CPU time was obtained with two iterations, except for the finest grid. Evaluating the weighted average in the CPU time gains for $\nu = 2$ and 3, the number of inner iterations of the solver that obtained the best average performance was $\nu = 3$, is approximately 1% lower. For this reason, the value adopted in the following studie is $\nu = 3$, for the algorithm built using MILU as solver for both grids generators.

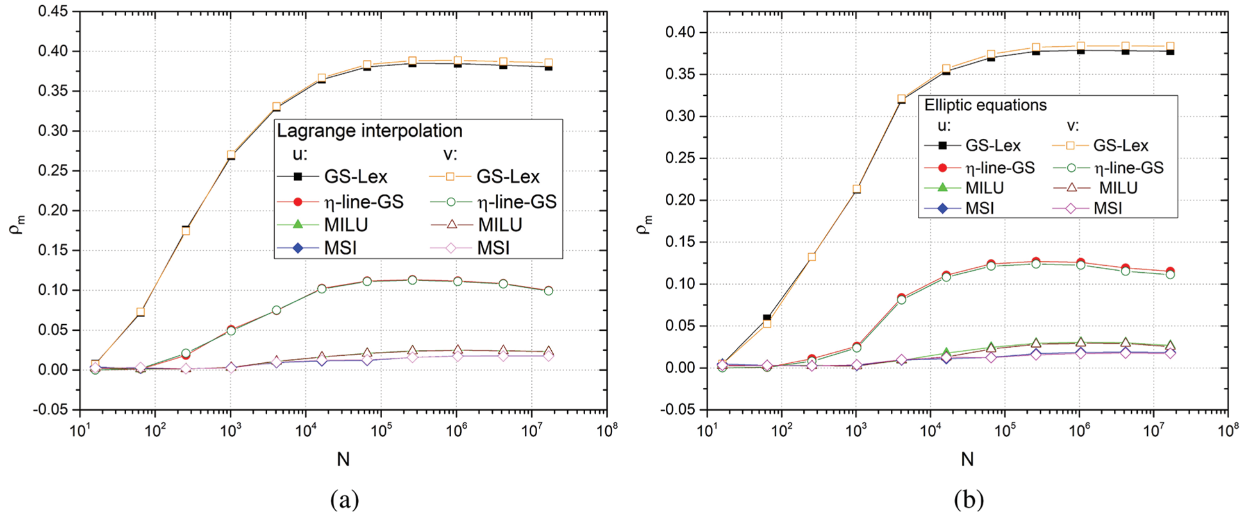


Figure 8: ρ_m *vs.* N. (a) Lagrange interpolation. (b) Elliptic equations

One must notice that when the value of ν increases or decreases in relation to the ν adopted, the CPU time increases. Depending on the value of ν, this increase can be significant. The analysis also showed that the increase in CPU time is the same, or very close, for both grid generators, as shown in Fig. 9. Santiago et al. [16] solved two-dimensional Burgers' equations by means of the finite difference method in Cartesian coordinates, on orthogonal grids, for the square cavity problem, and tested MSI as solver, which was also used in this work. The authors found $\nu_{optimum} = 5$ using the MG with FAS (the authors did not mention the value of σ used).

8.2.2 Number of Grid Levels

Another important parameter of MG is the number of grid levels. For instance, considering a problem with $r = 2$, the coarsening ratio used in this study, and $N = 128 \times 128$ unknowns, the highest possible number of grid levels is $L_{max} = 7$ grids, which are 2×2, 4×4, 8×8, 16×16, 32×32, 64×64 and 128×128.

For the study of influence of the number of grid levels of MG over CPU time, we consider the number of inner iterations suggested in the previous section. Fig. 10 depict the effect of the number of grid levels in the sum of the CPU time in order to determine the velocities u and v, with MSI and MILU as solvers. The symbol □ indicates the L that resulted on the lowest CPU time ($L_{optimum}$) in each curve.

Notice in Fig. 10a (Lagrange interpolation) that the number of grid levels in which the lowest CPU time was obtained is $L_{optimum} = L_{max} - (1, 2 \text{ or } 3)$. For grids generated by employing elliptic equations (Fig. 10b), $L_{optimum} = L_{max} - (1, 3 \text{ or } 4)$. In Fig. 10c (Lagrange interpolation), it is possible to observe that the number of grid levels in which the lowest CPU time was obtained is $L_{optimum} = L_{max} - (1, 2 \text{ or } 3)$. For grids generated by employing elliptic equations, Fig. 10d, is $L_{optimum} = L_{max} - (0 \text{ or } 1)$. Moreover, Fig. 10 show that for both grid generators, the CPU time can significantly increase according to the number of levels used. This is the case for small number of grid levels, that is, number of grid levels close to $L = 1$, which is the case of the singlegrid method. However, for $L \equiv L_{max}$, with MSI or MILU as solvers, the CPU time does not change significantly.

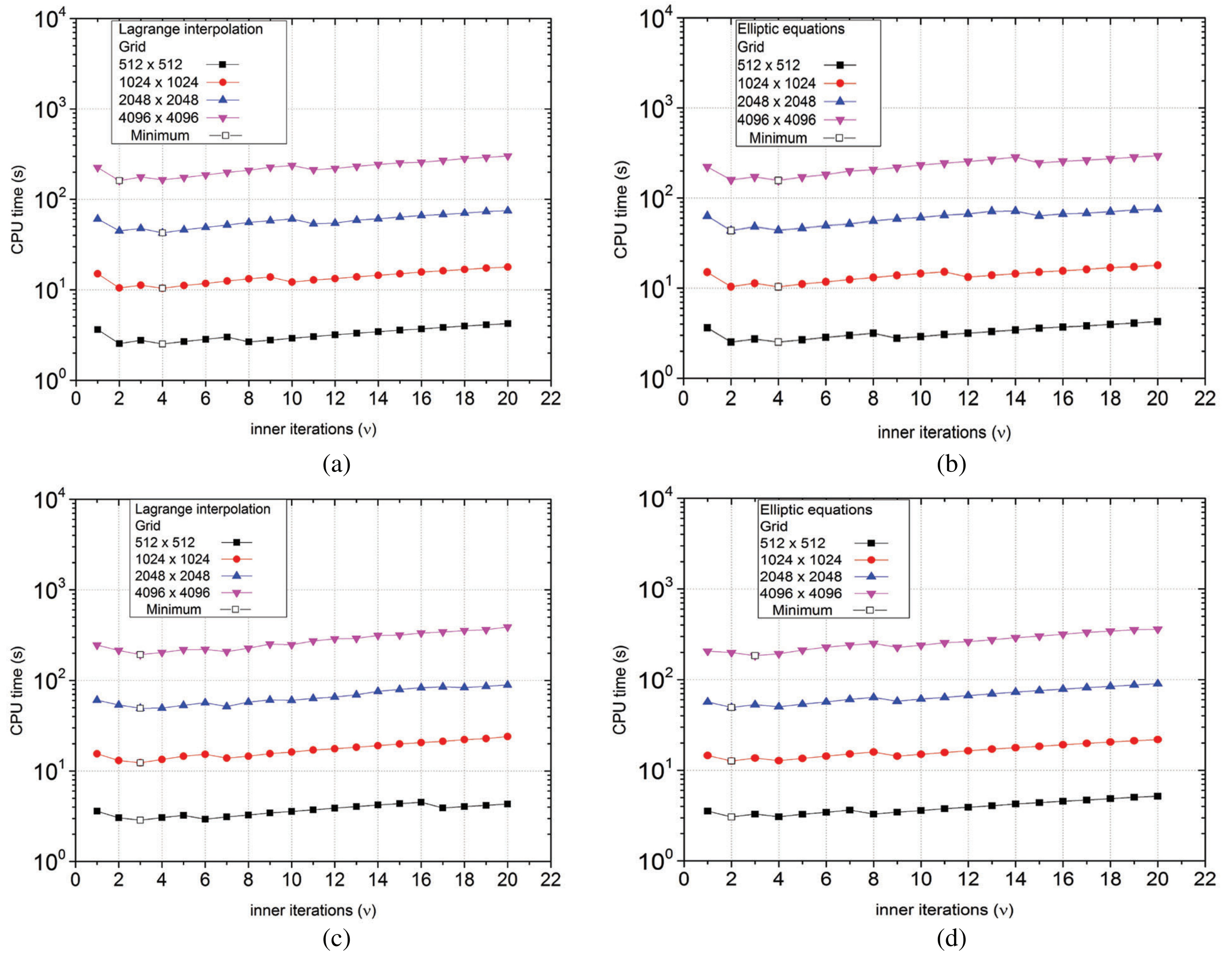


Figure 9: Effect of ν over CPU time using the MSI and MILU solvers. (a) MSI, Lagrange interpolation. (b) MSI, Elliptic equation. (c) MILU, Lagrange interpolation. (d) MILU, Elliptic equation

It was found that the number of grid levels what present the best average performance (weighted average of the percentages in gains in CPU time) was $L = L_{max} - 1$ for algorithms built using MSI as solver, for the four biggest grid sizes and for both grid generators used. For MILU solver, the best average performance was obtained for $L = L_{max} - 3$ for grids generated using Lagrange interpolation and $L = L_{max}$ for grids generated by employing elliptic equations, considering the four biggest grid sizes.

By means of the FVM, Rabi et al. [27] solved the conductive-convective problem with the multigrid method by employing orthogonal, structured grids for a problem with $N = 64 \times 64$ volumes and recommend using at least $L = 4$ for good error relaxation. Kumar et al. [24] state that there is no gain with L higher than 4 for the lid-driven cavity problem using orthogonal structured grids, FVM and $N = 513 \times 513$ volumes. For Burgers' equations using orthogonal structured grids and MG with FAS, Santiago et al. [16] reached $L_{optimum} = L_{max} - (0 \text{ to } 3)$ by employing FDM. The authors considered $L_{optimum} = L_{max}$, since for $L_{optimum} < L \leq L_{max}$ the CPU time was virtually the same.

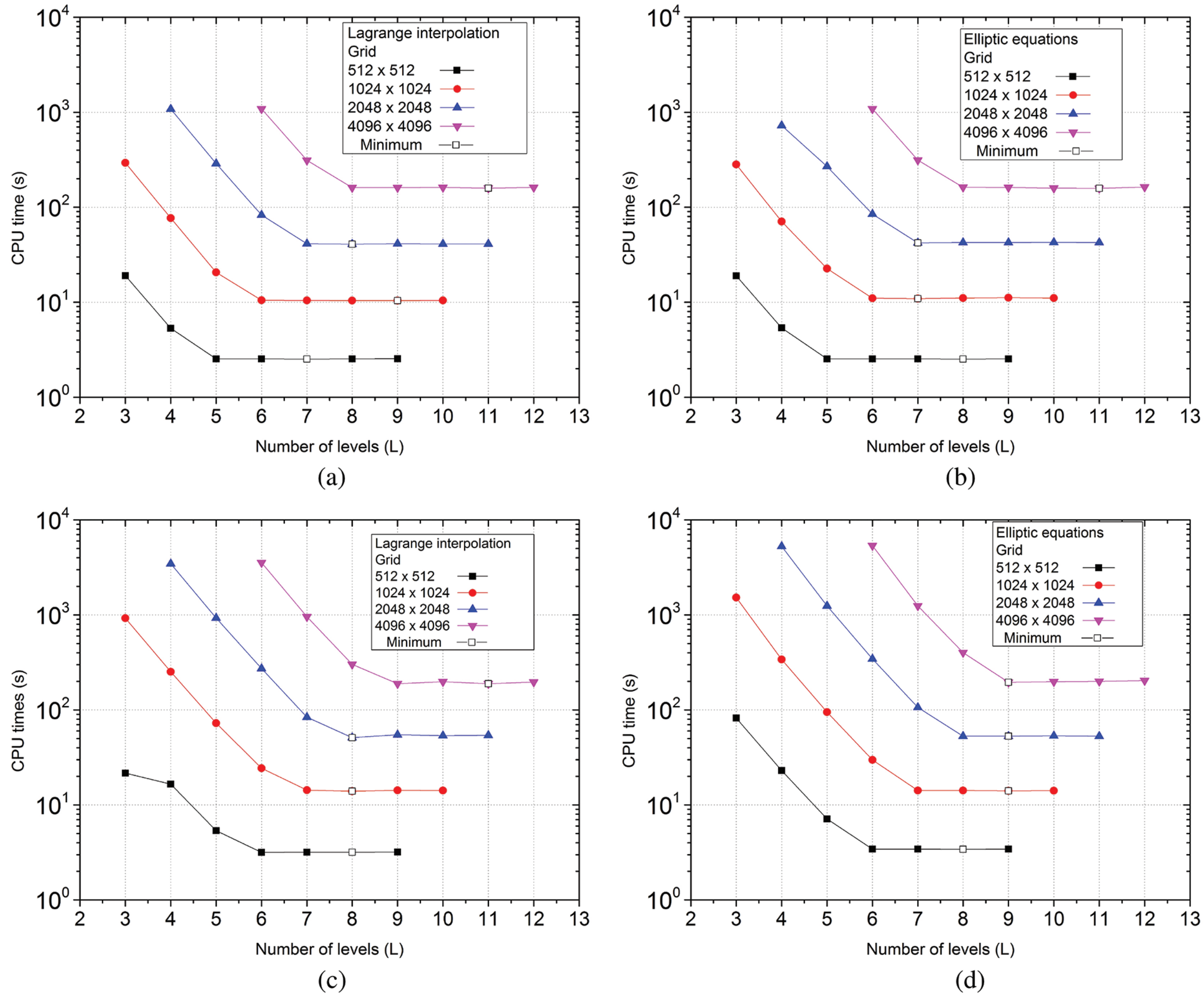
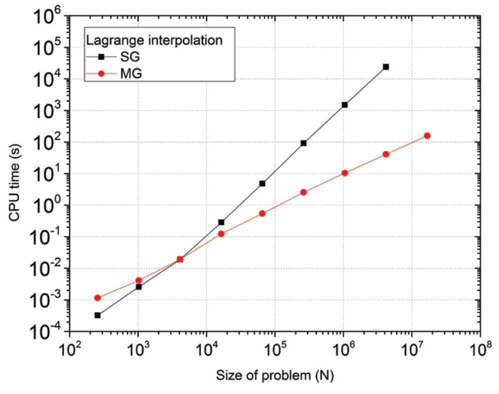


Figure 10: Effect of the number of grid levels over CPU time using the MSI and MILU solvers. (a) MSI, Lagrange interpolation. (b) MSI, Elliptic equation. (c) MILU, Lagrange interpolation. (d) MILU, Elliptic equation

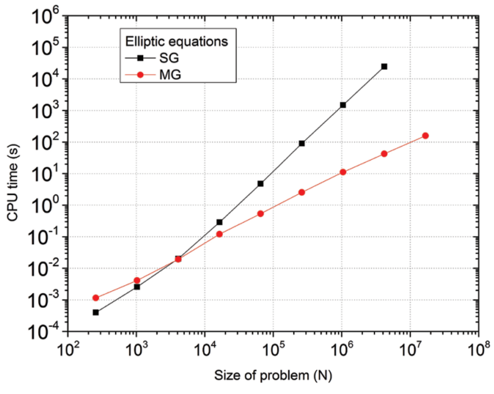
8.2.3 Computational Effort

On the analysis of influence of the number of unknowns of the system of equations over CPU time, the number of inner iterations and the number of grid levels are considered. Both numbers were recommended on the previous subsections. Problems of size $N = 16 \times 16$ up to the highest size of problem supported by the random access memory (RAM) of the machine used for simulations, $N = 4096 \times 4096$, were considered in the analysis for the MG with FAS. For the sake of comparison, results obtained with the singlegrid method (SG) for problems of size $N = 16 \times 16$ up to $N = 2048 \times 2048$ are also shown. Results are presented in Fig. 11 for the MSI and MILU solvers.

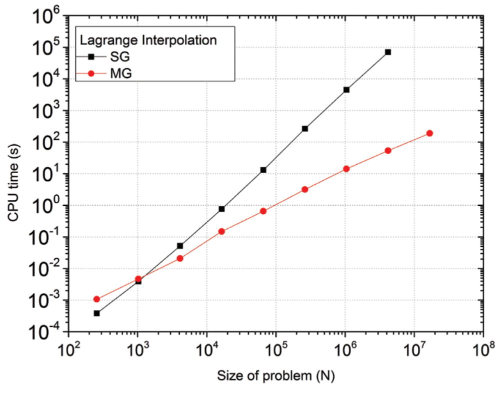
As shown in Fig. 11, the MG was extremely efficient. According to Ferziger et al. [3], the higher the number of unknowns N, the higher the advantage of MG over SG.



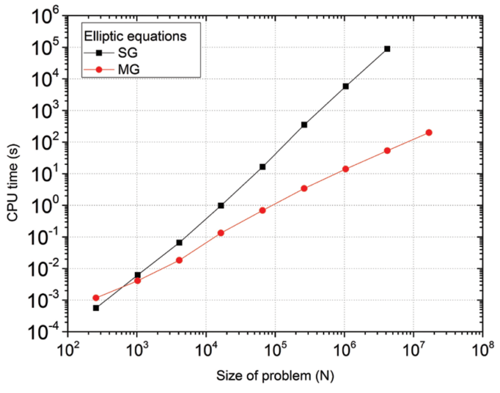
(a)



(b)



(c)



(d)

Figure 11: Effect of the number of unknowns over CPU time using MSI and MILU solvers. (a) MSI, Lagrange interpolation. (b) MSI, Elliptic equation. (c) MILU, Lagrange interpolation. (d) MILU, Elliptic equation

In order to determine the complexity (computational cost) of the solvers associated to the method, a geometrical adjustment of the type $t_{CPU}(N) = cN^p$ was made, where c stands as a relative constant of the method, p represents the algorithm's order or curve inclination, and N, the number of variables of the problem. Ideally, MG presents $p = 1$, what indicates that the CPU time increases proportionally to the number of unknowns (N) [6,22,28]. To calculate these coefficients, grids $N = 64 \times 64$ up to $N = 4096 \times 4096$ are considered for the multigrid method (MG), while grids $N = 64 \times 64$ up to $N = 2048 \times 2048$ are considered for the SG. Results of the geometrical adjustment are depicted in Tab. 1, as well as the values of c and p.

Notice that the theoretical performance of the MG for Burgers' equations using the general curvilinear coordinates system was confirmed, $p \approx 1$. Besides, $p \approx 2$ for the SG, what was also expected [4]. Furthermore, the CPU time and p values were very close for both methods, with a slight advantage of the MSI.

Tab. 2 shows execution time, in seconds, of the SG and MG with MSI and MILU as solvers as well as the grid generators using Lagrange interpolation and elliptic equations.

Table 1: Coefficient c and exponent p for the singlegrid and multigrid methods

Burgers equations	MG		SG	
	c	p	c	p
Lagrange interpolation—MSI	3.341E-06	1.072	7.709E-10	2.038
Elliptic equations—MSI	3.155E-06	1.078	8.328E-10	2.033
Lagrange interpolation—MILU	2.625E-06	1.106	2.605E-09	2.026
Elliptic equations—MILU	2.050E-06	1.125	4.259E-09	2.011

Based on the data presented in Tab. 2, considering MG with FAS, the CPU times for the problem with the MSI solver and grids generated using Lagrange interpolation, are smaller than those obtained with MILU, being approximately 19% faster in the most refined grid. For the problem with the MSI solver and grids generated using elliptic equations are also smaller than those obtained with MILU, being approximately 16% faster in the most refined grid. Comparing SG CPU times, in Tab. 2, MSI was also faster in all grids.

Table 2: Execution time in seconds

Grid	Generator	Solver	SG	MG
256×256	Lagrange interpolation	MSI	4.806	0.539
		MILU	13.278	0.651
	Elliptic equations	MSI	4.744	0.533
		MILU	13.059	0.645
512×512	Lagrange interpolation	MSI	91.447	2.535
		MILU	268.773	3.171
	Elliptic equations	MSI	91.026	2.527
		MILU	272.577	3.843
1024×1024	Lagrange interpolation	MSI	1500.109	10.404
		MILU	4370.962	14.325
	Elliptic equations	MSI	1491.337	11.154
		MILU	4378.430	15.731
2048×2048	Lagrange interpolation	MSI	24010.317	41.006
		MILU	70533.482	51.175
	Elliptic equations	MSI	24484.499	42.661
		MILU	69576.038	55.018
4096×4096	Lagrange interpolation	MSI	–	158.838
		MILU	–	189.018
	Elliptic equations	MSI	–	158.729
		MILU	–	184.816

Tab. 3 shows the discretization error in infinity norm of the SG and MG with MSI as solver as well as the grid generators using Lagrange interpolation and elliptic equations.

Considering the results of the Tab. 3, the solutions obtained for the problem with grids generated using elliptic equations are slightly more accurate than those using Lagrange interpolation.

Table 3: Infinity norm of error for the problem solved with the MSI solver and both grid generators

Grid	MSI	
	Lagrange interpolation	Elliptic equations
256×256	5.4838644067E-06	3.1500430985E-06
512×512	2.8784941319E-06	1.6042873321E-06
1024×1024	1.4741079725E-06	8.0633172263E-07
2048×2048	7.4539890321E-07	4.0348652942E-07
4096×4096	3.7419632012E-07	2.0167536730E-07

In this sense, the MG with MSI as solver and grids generated by employing Lagrange interpolation stood out as the best combination (lowest CPU time and discretization error) when compared to the other possible combinations studied.

9 Conclusions

In this paper, two-dimensional Burgers' equations in steady state with constant properties, using a system of curvilinear coordinates in an L-shaped domain was solved. The influence of some parameters of the geometric multigrid method as well as some solvers to solve this problem were assessed. In order to generate grids, Lagrange interpolation and systems of elliptic equations were used. Equations were discretized by FVM, generating systems of linear equations, which were solved by applying the Geometric Multigrid method and the GS-Lex, η-line-GS, MSI and MILU solvers. Convergence of the multigrid method was achieved by calculating the coordinate transformation metrics as presented in this paper. Based on the obtained results, it was mainly found that from the solvers employed, MSI and MILU are the most efficient for the problem assessed, with empirical average convergence factor close to 0.03; results show that the MSI solver, with $\nu = 4$ and $L = L_{max} - 1$, is faster than MILU for both grid generators, since it has the best complexity factors values. In the grid $N = 4096 \times 4096$, the multigrid method with MSI as solver is approximately 19% e 16% faster than the MILU solver, for grids generated by Lagrange interpolation and elliptic equations, respectively. The algorithm built using the MG with MSI as solver and grids generated by employing Lagrange interpolation stood out as the best combination (lowest CPU time and discretization error) when compared to the other possible combinations studied.

It is also necessary to restrict the coordinates transformation metrics to solve this problem in curvilinear grids. Therefore, it shows how to calculate the coordinates transformation metrics in the coarser grids.

Acknowledgement: The first author would like to thank the State University of Centro-Oeste (UNICENTRO) for granting a financial support and full leave of absence from teaching activities. The second author thanks CNPq for the financial resources for the project. The authors acknowledge the infrastructure of the Numerical Experimentation Laboratory (LENA) and the Graduate Program in Numerical Methods in Engineering of the Federal University of Parana (UFPR).

Funding Statement: The author(s) received no specific funding for this study.

Conflicts of Interest: The authors declares that they have no conflicts of interest to report regarding the present study.

References

1. Versteeg, H. K., Malalasekera, W. (2007). *An introduction to computational fluid dynamics: The finite volume method*. Harlow, England: Pearson Education.
2. Patankar, S. (1980). *Numerical heat transfer and fluid flow*. Washington: Hemisphere Publishing Corporation.
3. Ferziger, J. H., Peric, M. (2002). *Computational methods for fluid dynamics*, 3rd edition. New York, USA: Springer.
4. Burden, R., Faires, J. (2016). *Numerical analysis*, 10th edition. Boston: Cengage Learning.
5. Briggs, W. L., Henson, V. E., McCormick, S. F. (2000). *A multigrid tutorial*, 2nd edition. Philadelphia: SIAM.
6. Trottenberg, U., Oosterlee, C. W., Schuller, A. (2001). *Multigrid*. San Diego: Academic Press.
7. Hackbusch, W. (2003). *Multi-grid methods and applications*, vol. 4. Berlin: Springer Science & Business Media.
8. Smith, K. M., Cope, W. K., Vanka, S. P. (1993). A multigrid procedure for three-dimensional flows on non-orthogonal collocated grids. *International Journal for Numerical Methods in Fluids, 17(10),* 887–904. DOI 10.1002/fld.1650171005.
9. Oosterlee, C., Wesseling, P., Segal, A., Brakkee, E. (1993). Benchmark solutions for the incompressible Navier–Stokes equations in general co-ordinates on staggered grids. *International Journal for Numerical Methods in Fluids, 17(4),* 301–321. DOI 10.1002/fld.1650170404.
10. Li, J., Yu, B., Wang, M. (2015). Benchmark solutions for two-dimensional fluid flow and heat transfer problems in irregular regions using multigrid method. *Advances in Mechanical Engineering, 7(11),* 1–17.
11. Demirdžić, I., Lilek, Ž., Perić, M. (1992). Fluid flow and heat transfer test problems for non-orthogonal grids: Bench-mark solutions. *International Journal for Numerical Methods in Fluids, 15(3),* 329–354.
12. Zheng, Y. (2016). Stochastic stability of Burgers equation. *Acta Mathematica Sinica, English Series, 32(12),* 1509–1514.
13. Ferm, L., Lötstedt, P. (1997). Two-grid solution of shock problems. *SIAM Journal on Scientific Computing, 18(6),* 1533–1552.
14. Zhang, W., Zhang, C., Xi, G. (2010). An explicit chebyshev pseudospectral multigrid method for incompressible Navier–Stokes equations. *Computers & Fluids, 39(1),* 178–188.
15. Neveu, E., Debreu, L., Le Dimet, F. X. (2011). Multigrid methods and data assimilation-convergence study and first experiments on non-linear equations. *Revue Africaine de la Recherche en Informatique et Mathématiques Appliquées, 14,* 63–80.
16. Santiago, C. D., Marchi, C. H., Souza, L. F. (2015). Performance of geometric multigrid method for coupled two-dimensional systems in CFD. *Applied Mathematical Modelling, 39(9),* 2602–2616. DOI 10.1016/j.apm.2014.10.067.
17. Thompson, J. F., Warsi, Z. U., Mastin, C. W. (1985). *Numerical grid generation: Foundations and applications,* vol. 45. North-Holland, New York, USA: Elsevier.
18. Roache, P. J. (2002). Code verification by the method of manufactured solutions. *Journal of Fluids Engineering, 124(1),* 4–10. DOI 10.1115/1.1436090.

19. Anderson, D., Tannehill, J. C., Pletcher, R. H. (2016). *Computational fluid mechanics and heat transfer*, 3rd edition. Boca Ratón, USA: CRC Press.
20. Saad, Y. (2003). *Iterative methods for sparse linear systems,* vol. 82*,* 2nd edition. Philadelphia: Society for Industrial and Applied Mathematics—SIAM.
21. Anunciação, M., Pinto, M. A. V., Neundorf, R. L. A. (2020). Solution of the Navier–Stokes equations using projection method and preconditioned conjugated gradient with multigrid and ILU solver. *Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniera, 36(1),* 1–22.
22. Wesseling, P. (2004). *An introduction to multigrid methods*. Philadelphia: R. T. Edwards.
23. Schneider, G., Zedan, M. (1981). A modified strongly implicit procedure for the numerical solution of field problems. *Numerical Heat Transfer, 4(1),* 1–19. DOI 10.1080/01495728108961775.
24. Santhosh Kumar, D., Suresh Kumar, K., Kumar Das, M. (2009). A fine grid solution for a lid-driven cavity flow using multigrid method. *Engineering Applications of Computational Fluid Mechanics, 3(3),* 336–354. DOI 10.1080/19942060.2009.11015275.
25. Wienands, R., Joppich, W. (2005). *Practical Fourier analysis for multigrid methods*. Washington, USA: Chapman and Hall/CRC Press.
26. Oliveira, M. L., Pinto, M. A. V., Gonçalves, S. F. T., Rutz, G. V. (2018). On the robustness of the xy-zebra-Gauss–Seidel smoother on an anisotropic diffusion problem. *Computer Modeling in Engineering & Sciences, 117(2),* 251–270. DOI 10.31614/cmes.2018.04237.
27. Rabi, J. A., de Lemos, M. J., (2001). Optimization of convergence acceleration in multigrid numerical solutions of conductive–convective problems. *Applied Mathematics and Computation, 124(2),* 215–226. DOI 10.1016/S0096-3003(00)00088-6.
28. Wienands, R., Joppich, W. (2004). *Practical Fourier analysis for multigrid methods*. Boca Raton, USA: Chapman and Hall/CRC.