Applied Mathematical Modelling 36 (2012) 2996–3006

Contents lists available at SciVerse ScienceDirect

Applied Mathematical Modelling

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

Analysis of algebraic multigrid parameters for two-dimensional steady-state heat diffusion equations

R. Suero [a,b,*], M.A.V. Pinto [c], C.H. Marchi [c], L.K. Araki [c], A.C. Alves [d]

[a] Federal Institute of Paraná – Campus of Paranaguá, 453 Antonio Carlos Rodrigues Street, 83215-750 Paranaguá-PR, Brazil
[b] Federal University of Paraná, Post-Graduate Course of Numerical Methods in Engineering, Centro Politécnico, 81531-980 Curitiba-PR, Brazil
[c] Federal University of Paraná, Department of Mechanical Engineering, Centro Politécnico, Block IV, 81531-980 Curitiba-PR, Brazil
[d] Positivo University, Sector of Exact Sciences and Technology, 81280-330 Curitiba-PR, Brazil

A R T I C L E I N F O

Article history:
Received 13 April 2011
Received in revised form 21 September 2011
Accepted 29 September 2011
Available online 10 October 2011

Keywords:
Parameters optimization
Algebraic multigrid
Square grids
Triangular grids

A B S T R A C T

In this work, it is provided a comparison for the algebraic multigrid (AMG) and the geometric multigrid (GMG) parameters, for Laplace and Poisson two-dimensional equations in square and triangular grids. The analyzed parameters are the number of: inner iterations in the solver, grids and unknowns. For the AMG, the effects of the grid reduction factor and the strong dependence factor in the coarse grid on the necessary CPU time are studied. For square grids the finite difference method is used, and for the triangular grids, the finite volume one. The results are obtained with the use of an adapted AMG1R6 code of Ruge and Stüben. For the AMG the following components are used: standard coarsening, standard interpolation, correction scheme (CS), lexicographic Gauss–Seidel and *V*-cycle. Comparative studies among the CPU time of the GMG, AMG and singlegrid are made. It was verified that: (1) the optimum inner iterations is independent of the multigrid, however it is dependent on the grid; (2) the optimum number of grids is the maximum number; (3) AMG was shown to be sensitive to both the variation of the grid reduction factor and the strong dependence factor in the coarse grid; (4) in square grids, the GMG CPU time is 20% of the AMG one.

1. Introduction

The multigrid method [1,2] belongs to a group of iterative solvers, and it is one of the most efficient and widespread methods to solve large systems of linear equations [3]. Its efficiency is based on the fact that the multigrid method presents a potential to solve $N \times N$ linear systems with only $O(N)$ computational effort [4]. Two different approaches can be accomplished employing the multigrid method according to the kind of data and information employed and also how the operators deal with them: the geometric multigrid (GMG) and the algebraic multigrid (AMG). The main difference between AMG and GMG is related to the manner of constructing the coarser grids [5]: the AMG method requires no knowledge of the geometry of the problem [6]. The GMG method [1,2] employ fixed grid hierarchies and, therefore, an efficient interplay between smoothing and coarse-grid correction has to be ensured by selecting appropriate smoothing processes [1]. On the other hand, the AMG method [5–8] fixes the smoother to some simple relaxation scheme and enforces an efficient interplay with coarse-grid correction by choosing the suitable coarser levels and the interpolation [1]. The grid hierarchies in the AMG is

* Corresponding author at: Federal Institute of Paraná – Campus of Paranaguá, 453 Antonio Carlos Rodrigues Street, 83215-750 Paranaguá-PR, Brazil. Tel.: +55 41 3721 8303; fax: +55 41 3721 8308.

E-mail addresses: roberta.suero@ifpr.edu.br (R. Suero), marcio_villela@ufpr.br (M.A.V. Pinto), marchi@ufpr.br (C.H. Marchi), lucaraki@ufpr.br (L.K. Araki), aalves@up.com.br (A.C. Alves).

0307-904X/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.apm.2011.09.088

generated in the setup phase (which is an initial or start-up phase), by considering the coefficient matrix, as well as the build interpolation, restriction and coarse-grid operators [6,9].

The application of the AMG method includes problems in which the use of the GMG method is difficult or even impracticable [7], such as: unstructured grids, large matrix equations which are not at all derived from continuous problems, extreme anisotropic equations and so on. A remarkable use of the AMG method takes place when there is none information about the problem geometry [9]. Table 1, adapted from Chang et al. [10], provides a comparison between the AMG and the GMG methods.

Both strategies, GMG and AMG, present similar steps. The first one is the generation of the coarser grids (levels), which is followed by the transfer of information among different grids (restriction and prolongation operators). After that comes the solution of the linear system in each grid using an iterative smoother algorithm (solver), and the choice of a multigrid cycle (*F*-cycle, *V*-cycle, *W*-cycle, among other ones) [1,2]. The decision about which iterative method to use as smoother (solver), how to coarsen the problem, or even how to transfer information between the grids, often involve considerable algorithmic research [4].

According to Trottenberg et al. [1], a single modification in the algorithm might result in a significant reduction of the CPU time requirements. The efficiency of the multigrid methods is also related to the adaptations of the multigrid components, which should be made properly according to the underlying physical problem and the variational formulation [11]. Unfortunately, the works available in the literature do not present deep studies about the components of the AMG algorithm and their optimization. More precisely, in such works, new coarsening algorithms and/or new interpolation operators are introduced, like the work of Xiao et al. [12], or to papers in which the AMG performance is compared to the GMG one [11,13]. In the latter case, such comparisons are limited to the CPU time, the number of cycles and the study of the multigrid efficiency provided by the speedup value.

The CPU time and its growth, according to the number of unknowns, were studied by Watanabe et al. [13], for both AMG and GMG. Both multigrid methods were also studied by Langer and Pusch [11], where comparisons for the number of cycles spent by the AMG and by the GMG, as well as the time requirements for the auxiliary grids generation were presented. The number of cycles was also revised by Wu and Elman [14], by using as stop criterion a given tolerance value; it was seen that the GMG convergence was slower than the AMG for convection–diffusion problems. Campos et al. [15] made a comparison between the performances of the AMG and the GMG, both with parallelized and preconditioned algorithms which are suitable for a non-linear system of differential equations. Additionally, simulations were performed varying the number of grids, for both the AMG and the GMG, and the grid reduction factor for the AMG, executed by 1, 2, 4 or 6 processors; the performance of the AMG algorithm was better than the one obtained by the GMG for both CPU time and memory.

Systematic studies about the multigrid parameters where found only for the GMG method. Gaspar et al. [16,17] presented theoretical and numerical results for the GMG with triangular grids, by applying distinct multigrid cycles, different numbers of inner iterations and proposing a new smoother (solver). On the other hand, Oliveira et al. [18] computed optimum values for the inner iterations and the levels used in heat diffusion problems with structured square grids.

Motivated by all these previous works, the purpose of this paper is to study the components and the parameters associated to the AMG method, by applying the numerical AMG1R6 code implemented by Ruge and Stüben [7], in order to find the optimum parameters for the AMG. The meaning of optimum values for each AMG parameter, considered here, is associated to the minimization of the CPU time, i.e., the values for each AMG parameter which corresponds to the smallest CPU time in each simulation, keeping fixed the values for the other parameters.

Two-dimensional Laplace and Poisson type equations are employed, using both square structured (both equations) and triangular grids (Laplace equation), in order to investigate the influence of the number of inner iterations (ν) and the number of levels (L) for both AMG and GMG methods, providing comparisons on the performances of such methods. Moreover, in the case of AMG, two parameters (which influence the coarser grids generation) are also investigated: the grid reduction factor (θ) and the strong dependence factor in the coarse grid (ε).

Although most works present the values used for each AMG parameter, or at least, some of them [1,2,6,7,14,19,20], none of these papers explain clearly why such values are employed; in most cases, it is only said that the used value is a universal practice, without giving more details. Based on these facts, this work tries to fill this lack of information providing the optimized values for the AMG parameters, compared to the standard ones.

As usual, the standard AMG algorithm adopts the following values for its parameters: $\nu = 1$ [1,2,6,7,19,20], $L = L_{maximum}$ [14,19,20], $\theta = 0.25$ [1,7,10,19,20] and $\varepsilon = 0.35$ [7,19]. In some works available in the literature [1,2,6], the authors employ values other than the ones listed here. However, they do not clearly explain the reasons which take them to use such values.

Table 1
Comparison between the GMG and the AMG methods (adapted from Chang et al. [10]).

Features	GMG	AMG
Solved problem	Continuous problem	Linear systems of algebraic equations
Used information	Geometrical structure of the problem	Only entries of the matrix
Program	Necessity of composing a program for each problem	Only one program for different problems
Efficiency	Very good	Good

For the values of L, some works do not even mention the number of grids employed [6,7,11]. The values listed above are the most utilized in literature and will be used in this paper to compare the optimized parameters presented in this work. A paper with preliminary results contained in this work was published at CONEM/2010 [21].

2. Mathematical and numerical models

Three two-dimensional heat transfer problems, with Dirichlet boundary conditions, are investigated in this work, governed by the following differential equation in the Cartesian coordinate system [22]:

$$\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}=S, \tag{1}$$

where x and y are the coordinate directions, T denotes the temperature, and S is a source term (which can be null, providing the Laplace equation, or either a constant or a function of x and/or y, providing a Poisson type equation). While the Laplace equation is employed twice, for two different sets of boundary conditions, listed in Table 2, the Poisson type equation has as source term $S=-2[(1-6x^2)y^2(1-y^2)+(1-6y^2)x^2(1-x^2)]$. In order to distinguish the two sets of boundary conditions employed for the Laplace equation, the labels Laplace 1 and Laplace 2 are used in this work, according to the boundary conditions listed in Table 2.

When square-structured grids are employed, both the Laplace as well as Poisson type equations are discretized by applying the finite difference method (FDM) [23]. On the other hand, for the triangular grids, the Laplace equation is discretized by employing the finite volume method (FVM) [24], applying the boundary conditions with ghost cells. For both grid types, a unitary square domain is employed and the second order central differencing scheme (CDS) is used to the derivative approximations. Whatever the discretization method used, the discretization process for the whole domain provides a system of linear equations of the type $\mathbf{AT} = \mathbf{b}$, where the coefficients matrix $\mathbf{A}$ is pentadiagonal, symmetric and positive definite, $\mathbf{T}$ is the unknowns vector and $\mathbf{b}$ is the source term vector.

3. The algebraic multigrid (AMG) method

The AMG method is composed by two steps [1,5–7]. The first one, called setup phase, is responsible for the grid generation and the construction of the transfer operators (restriction and prolongation) among the grids involved in this process. The second step is the solution phase, which is direct and employs the operators previously defined in the setup phase. In the solution phase, the coarser grids are visited according to a previously proposed cycle. In the construction of the coarser grids, it is necessary to make a partition of the original grid nodes, $i \in \Omega^h$, where Ω^h denotes the indices set $\{1,2,\ldots,n\}$. This partition gives rise to two disjoint subsets: the C^h, representing the variables which should be contained in the coarse level (C-variables), and the F^h, the complementary set, in which are placed the nodes not included in the coarse grid. This partition is based on the strong algebraic connections; moreover, in order to determine it, some other sets must be defined as [9]:

$$N_i=\left\{j\in\Omega^h : j\neq i, a_{ij}\neq 0\right\}, \tag{2}$$

$$S_i=\left\{j\in N_i : -a_{ij}\geq \theta \max_{a_{ik}<0}|a_{ik}| \,\text{with fixed}\, \theta,\, 0<\theta<1\right\}, \tag{3}$$

$$S_i^T=\{j\in\Omega : i\in S_j\}, \tag{4}$$

where: N_i is the neighborhood of the given point i, S_i determines the set of points which strongly influences i and S_i^T is the set of points which are strongly dependent of i. The parameter θ ($0 < \theta < 1$) which denotes the grid reduction factor is a constant which quantifies how much a given point is strongly connected to the other ones. The influence of this parameter on the CPU time is also investigated in this work.

After providing the partition, for each point i, other three subsets are obtained: C_i, D_i^S and D_i^w, necessary to the grid coarsening. The subset $C_i = C^h \cap S_i$ selects the neighbor points in the coarse grid which strongly influence the point i, while the

Table 2
Problem labels, boundary conditions and analytical solutions for Eq. (1).

Problem label	Boundary conditions	Analytical solution
Laplace 1	$T(0,y)=T(x,0)=0$ $T(x,1)=x, T(1,y)=y$	$T(x,y)=xy$
Laplace 2	$T(0,y)=T(x,0)=T(1,y)=0$ $T(x,1)=\sin(\pi x)$	$T(x,y)=\sin(\pi x)\frac{\sinh(\pi y)}{\sinh(\pi)}$
Poisson	$T(0,y)=T(x,0)=0$ $T(1,y)=T(x,1)=0$	$T(x,y)=(x^2-x^4)(y^4-y^2)$

subset $D_i^S = D_i \cap S_i$, where $D_i = N_i - C_i$, is composed by all the points in the neighborhood of F^h which strongly influence the point i, and the subset $D_i^w = D_i - S_i$ contains the weakly connected points. The interpolator operator (responsible for the transfer of information from a coarser to a finer grid) is given by [2]

$$\left(I_H^h e^H\right) = \begin{cases} e_i^H, & \text{if } i \in C^h \\ \sum_{j\in C_i} w_{ij} e_j^H, & \text{if } i \in F^h, \end{cases} \tag{5}$$

where

$$w_{ij} = -\frac{a_{ij} + \sum\limits_{m\in D_i^s} \left(\frac{a_{im}a_{mj}}{\sum\limits_{k\in C_i} a_{mk}}\right)}{a_{ii} + \sum\limits_{n\in D_i^w} a_{in}}. \tag{6}$$

The influence of the parameter $\varepsilon > 0$, the strong dependence factor in the coarse grid, on the CPU time is also studied in this work. This parameter is employed in the judgment if a point $j \in D_i^S$ strongly influences a given point in C, in such way that the interpolation is also influenced. So, it is needed to define a data set by Iwamura et al. [19]:

$$S_i^D = \left\{ j \in D_i^s : \sum_{l\in C_i} |a_{ij}| > \varepsilon\left(\frac{|a_{ij}|}{\max|a_{ik}|}\right)\max|a_{jl}| \text{ with fixed } \varepsilon > 0 \right\}. \tag{7}$$

More details about the adopted algorithm employed for the interpolation procedure can be found in Iwamura et al. [19].

4. Numerical results

Around 700 numerical simulations were done, in order to study the influence of the following parameters on the CPU time: the number of inner iterations (ν) in the smoothing algorithm (solver), the number of grids (L), the grid reduction factor (θ) and the strong dependence factor in the coarse grid (ε). For all the performed investigations, three different problem sizes were considered: for square grids (FDM), $257 \times 257 = 66{,}049$, $1025 \times 1025 = 1{,}050{,}625$ and $4097 \times 4097 = 16{,}785{,}409$ nodes; for triangular grids (FVM), $2^{18} = 262{,}144$, $2^{20} = 1{,}048{,}576$ and $2^{24} = 16{,}777{,}216$ volumes. In order to provide a complete overview of the effects of optimized parameters on the CPU time after all the optimizations performed here, for square grids (FDM) were studied grids as coarse as $5 \times 5 = 25$ up to $4097 \times 4097 = 16{,}785{,}409$ nodes (each node corresponding to one unknown). Otherwise, for triangular grids, the number of volumes varies from $2^2 = 4$ up to $2^{24} = 16{,}777{,}216$; in this case, also the influence of the number of unknowns (N) is studied in this work. All the simulations were done in order to compute the optimum value for each one of the tested parameters. The choice of these values is made in such a way that they offer the minimum CPU time for the AMG algorithm, while the values of the other parameters are fixed.

The computational code employed in this work is based on the AMG1R6 code, implemented by Ruge and Stüben [7]. In this code, the setup phase is based on the standard coarsening (based on the strong negative connections) and standard interpolation (chosen by the use of the standard coarsening). Otherwise, in the solution phase one has: V-cycle, null initial guess, lexicographic Gauss–Seidel as smoother (solver), non-dimensional residual $\overline{l_2}$-norm (based on the initial guess norm) and tolerance of 10^{-8}. The original code of Ruge and Stüben was changed in order to allow the evaluation of the implemented modifications on the CPU time performance. The input data is also modified for each grid type (square or triangular), to allow the use of a specific matrix generator, coupled to the main program.

4.1. Inner iterations (ν)

The determination of the optimum number of inner iterations was done based on three sets of grids, for each one of the grid geometries, as previously described. The range of values considered was $\nu = 1$–20. In the cases in which the CPU time was smaller than 10 s, the measured value corresponds to the arithmetic average among three simulations. For both grid geometries, it was employed as many levels as possible, while the values of the grid reduction factor (θ) and the strong dependence factor in the coarse grid (ε) were left constant and equal to 0.25 and 0.35, respectively, since these are the most utilized in the literature.

Fig. 1 presents numerical results for each one of the three problems listed in Table 2. Both square and triangular grids were employed: for square grids (FDM), Fig. 1 shows results for the grid with 4097×4097 nodes (totalizing 16,785,409 unknowns), while for triangular grids (FVM), the results are from a grid with 2^{24} volumes (16,777,216 unknowns). Qualitatively, the results for the other grids are analogous to the ones presented in Fig. 1 and, because of this, are not represented. According to the numerical results, the optimum value for the inner iterations (ν) equals 2 for square grids, independent on the solved problem. Otherwise, for triangular grids, the optimum value ν equals 1. Consequently, an influence of the grid type (triangular or square grids) on the optimum number of inner iterations was observed.

The CPU time reduction, for square grids, obtained with the use of the $\nu_{optimum} = 2$, in comparison to the current adopted value ($\nu = 1$) is about 7.5%, 9.9% and 1.1% for the Laplace 1, Laplace 2 and Poisson problems, respectively, in the considered grid ($4097 \times 4097 = 16{,}785{,}409$ nodes). The numerical results obtained in this work for the optimum value of ν agree to the

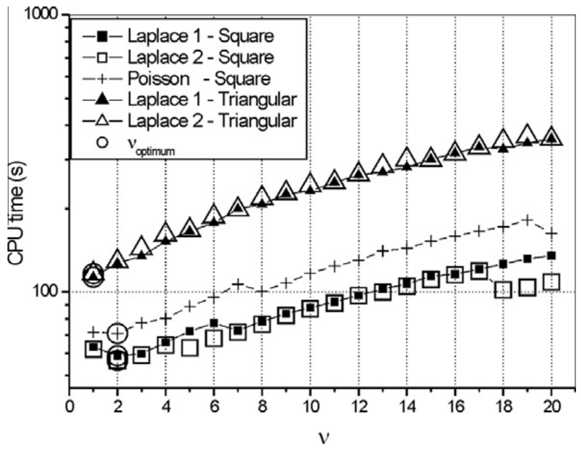

Fig. 1. CPU time *versus* inner iterations (ν) for square ($N = 4097 \times 4097 = 16{,}785{,}409$ nodes) and triangular ($N = 2^{24} = 16{,}777{,}216$) grids.

ones presented by Gaspar et al. [16] for triangular grids, and Oliveira et al. [18] for square grids. However, it must be noticed that these two works employed the GMG method.

4.2. Number of grids (L)

The influence of the number of grids on the CPU time is evaluated by using the optimum value of inner iterations obtained in the previous subsection: $\nu = 2$ for square grids and $\nu = 1$ for triangular grids. For both grid geometries, the values of the grid reduction factor (θ) and the strong dependence factor in the coarse grid (ε) were left constant and equal to 0.25 and 0.35, respectively.

In the numerical simulations performed in this paper, the number of grid levels (L) was considered in such a way that $1 \leqslant L \leqslant L_{maximum}$, where $L_{maximum}$ is the maximum number of different grids which can be employed in the multigrid cycle, for a given grid spacing ratio (GMG) or for a given value of θ (AMG). For example, in GMG, if $E = 512 \times 512$ elements grid is employed, with a ratio $r = 2$, the auxiliary grids are 256×256, 128×128, 64×64, 32×32, 16×16, 8×8, 4×4 and 2×2, which corresponds to the coarsest grid, with only one inner node, totalizing 9 grids (that is, $L_{maximum} = 9$). In the AMG, for $\theta = 0.25$, the problems for triangular grids were investigated in such a way that the number of unknowns (inner nodes) was the nearest possible when compared to square grids. In the AMG1R6 code [7], there is an option to solve a given problem using all the grid levels, that is, by using the grid with the smallest number of inner nodes. The number of grid levels employed, for triangular grids, was printed for each size problem and was used for the comparisons shown in Fig. 2. This figure presents the behavior of the CPU time according to the number of grid levels employed, for the finest grid considered ($4097 \times 4097 = 16{,}785{,}409$ nodes for the square grid and $2^{24} = 16{,}777{,}216$ volumes for the triangular one).

As can be seen, for both grid types, the number of grids influences the CPU time: the smaller the number of grids employed (L), the higher the CPU time requirement is. Based on this, the optimum number of grids (L), which should be used for all the numerical simulations, is the highest one, in other words, $L_{optimum} = L_{maximum}$.

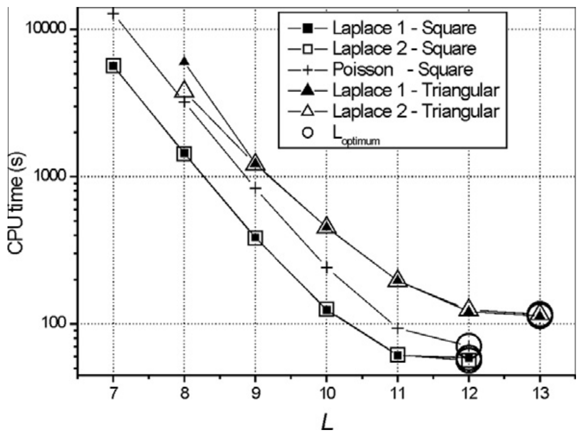

Fig. 2. CPU time *versus* number of levels (L) for square ($N = 4097 \times 4097 = 16{,}785{,}409$ nodes) and triangular ($N = 2^{24} = 16{,}777{,}216$) grids.

These results performed here agree with the ones obtained for the GMG method, presented by Oliveira et al. [18] and Gaspar et al. [16], who also employed all the grid levels for the GMG method, like Wu and Elman [14], Iwamura et al. [19] and Krechel and Stüben [20] for the AMG method.

4.3. Grid reduction factor (θ)

In order to investigate the effects of the grid reduction factor (θ) on the CPU time, the optimum values obtained in previous subsections were left constant: $\nu = 1$ for triangular grids and $\nu = 2$ for square ones and $L = L_{maximum}$. The strong dependence factor in the coarse grid (ε), whose optimization is presented in the next subsection, was left constant and equal to 0.35. The value of θ quantifies how much a given point is strongly connected to another one: depending on this relation, a given point is considered or not in the coarse grid.

Most authors [1,7,10,19,20] have utilized the value of θ equals to 0.25; this, however, is not a universal practice. For instance, Briggs et al. [2] used $\theta = 0.20$, while Falgout et al. [6] apply $\theta = 0.40$ for their numerical simulations. If an analogy to the GMG is done, it can be seen that $\theta = 0.25 = 1/4$ (in AMG) corresponds to the interpolation of four points in a finer grid, e.g., it is analogous to the grid spacing ratio $r = 2$ (in GMG). Because of this assertion, in order to keep an analogy to the GMG, besides the values of $\theta = 0.20$, 0.25 and 0.40 (found in literature), the values of $\theta = 0.11$ and $\theta = 0.0625$ were also studied. These latter values correspond to the grid spacing ratio of $r = 3$ and $r = 4$, respectively, for the GMG.

The influence of the values of θ on the CPU time is shown in Fig. 3, for both square and triangular grids. In this case, the number of variables is concerned to the most refined grid ($N = 4097 \times 4097 = 16{,}785{,}409$ nodes in the square grid and $2^{24} = 16{,}777{,}216$ volume in the triangular one), once numerical results for the other grids present an analogous behavior. The optimum values for θ are shown for each grid type and for each studied problem and, in general, diverge from the common value of $\theta = 0.25$ used as standard.

According to Fig. 3, the optimum values of θ vary with the problem solved. Using square grids: $\theta = 0.40$ for Laplace 1, $\theta = 0.25$ for Laplace 2 and $\theta = 0.20$ for Poisson. However, considering only the values $\theta = 0.20$, 0.25 and 0.40, the maximum increase in the CPU time was about 5.7% for all the problems. As can be seen, the influence of this parameter on CPU time in square grids is limited. If all the nodes satisfy Eq. (3), as observed for all the numerical cases investigated here, all the nodes are strongly connected to their respective neighbors and the influence on the CPU time is small.

The previous situation, however, does not apply to triangular grids: in this case, each node is connected to three other ones, satisfying or not Eq. (3). Because of this assertion, the CPU time is much more influenced by the variations on the values of the grid reduction factor (θ). Basically, the use of smaller values of θ implies in a smaller number of nodes employed in the auxiliary grids and, consequently, it implies in a small quantity of grid levels, improving the performance of the adopted algorithm. Based on these results, some other numerical simulations were performed in order to verify the influence of smaller values of θ on the CPU time. However, values smaller than $\theta = 0.0625$ cause, in most cases, the divergence of the algorithm. It seems that this can be caused by the strong lost of information between finer and coarser grids, induced by the use of a strongly reduced number of nodes in the coarse grids.

4.4. Strong dependence factor in the coarse grid (ε)

The study of the strong dependence factor in the coarse grid (ε) on the CPU time is presented in this subsection. The values employed for the other parameters are left constant and equal to the optimum ones, obtained in the previous subsections, for each grid type and each solved problem. The value of ε defines the strong dependence in the node set of the coarse grid C_i,

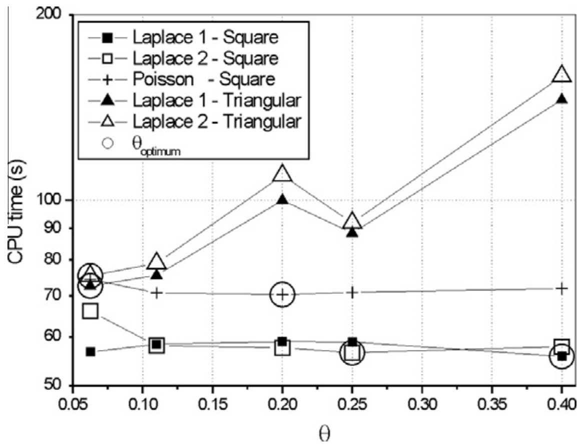

Fig. 3. CPU time *versus* grid reduction factor (θ) for square ($N = 4097 \times 4097 = 16{,}785{,}409$ nodes) and triangular ($N = 2^{24} = 16{,}777{,}216$) grids.

according to Eq. (7), and influences the construction of the auxiliary grids in the AMG. However, various works in the literature [6,11,13,15,20] do not mention the values of ε used for the AMG. Ruge and Stüben [7] employ $\varepsilon = 0.35$, while Iwamura et al. [19] use different values of ε according to the constructed grid level: $\varepsilon = 0.35$ is considered when the number of levels is smaller than or equal to 3, and $\varepsilon = 0.45$ is utilized when the number of levels is greater than 3. On the other hand, Trottenberg et al. [1] employ $\varepsilon = 0.20$ for the numerical solution of the Poisson equation.

The values of ε employed in this work varies from 0.20 to 0.50, with a step of 0.05, which covers all the values available in the literature [1,7,19], providing different ways to evaluate the influence of this parameter on the CPU time. Fig. 4 presents numerical results for the most refined grid using both the square ($N = 4097 \times 4097 = 16{,}785{,}409$ nodes) and the triangular grids ($N = 2^{24} = 16{,}777{,}216$ volumes); the other grids present similar behavior. As can be observed, in all the cases, the best results were obtained by using $\varepsilon = 0.35$ – the current value employed by Ruge and Stüben [7] and Iwamura et al. [19]; the exception is provided by the Poisson equation, in which the optimum value was 0.20 – exactly the one employed by Trottenberg et al. [1] for the same equation type. The use of $\varepsilon = 0.35$ for the Poisson equation, however, increases the CPU time in only 0.9%.

4.5. Number of unknowns (N)

In order to evaluate the efficiency of the adopted AMG compared to the theoretical one, which is of $O(N)$ [4], this subsection provides the CPU time needed to solve different problems with different grid sizes. All the optimum parameters obtained in the previous subsections, for different grid types and different problems, are employed in this section. For both, triangular and square grids, the AMG results are compared to the SG (singlegrid) ones; the SG method corresponds to the solution of the system of linear equations in only one grid (the basis grid), without using of auxiliary ones. In addition, for square grids, the GMG could be employed and their results are also presented. The complexity order of the AMG and GMG multigrid algorithms is evaluated by the computation of the CPU time (t_{CPU}) equation, which has the form

$$t_{CPU} = cN^P, \tag{8}$$

where c is a constant, N denotes the number of equations (number of nodes or volumes) and P is the exponent, which for the multigrid method should be approximately unitary, and represents the order of complexity of the given algorithm.

Numerical results for square grids are presented in Fig. 5, while Fig. 6 provides the results for triangular grids. Graphically, it is observed from Figs. 5 and 6 that the multigrid methods (AMG and GMG for square grids, and AMG for triangular ones) present similar values of P: in all cases, it is close of unity. The results presented in Table 3 confirm this fact, in which the values of P were obtained by the least square fitting, considering the results obtained in the four most refined grids for each problem/grid type. As can be seen, the performances of both AMG and GMG are really closer to the theoretical one, for the studied problems. The SG results are also closer to the theoretical ones, once the Gauss–Seidel algorithm was employed to solve the system of linear equations obtained by the discretization procedures, and, theoretically, the order of complexity of such algorithm is $O(N^2)$.

Although the values of P for both AMG and GMG methods using square grids are basically the same, it is possible to see from Fig. 5 that the CPU time using AMG is always greater than the result provided by the GMG algorithm when it is considered the same N. This is an effect of the complexity of the AMG method, when compared to the GMG: while the AMG method is more flexible, being applied in unstructured grids once it deals with only the matrices information, the GMG is faster by the use of the geometrical structure of the problem, as can be seen at Table 1. Overall, comparing the CPU time

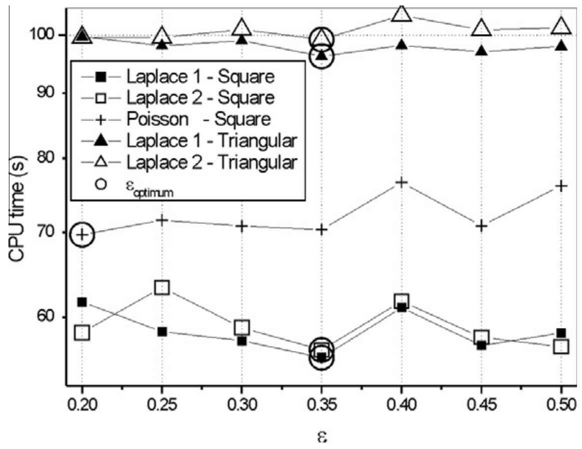

Fig. 4. CPU time *versus* strong dependence factor in the coarse grid (ε) for square ($N = 4097 \times 4097 = 16{,}785{,}409$ nodes) and triangular ($N = 2^{24} = 16{,}777{,}216$) grids.

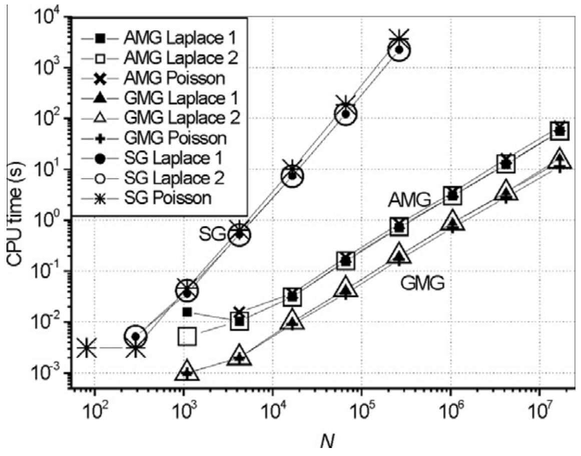

Fig. 5. CPU time *versus* number of unknowns (N) for the AMG, the GMG and the SG methods using square grids.

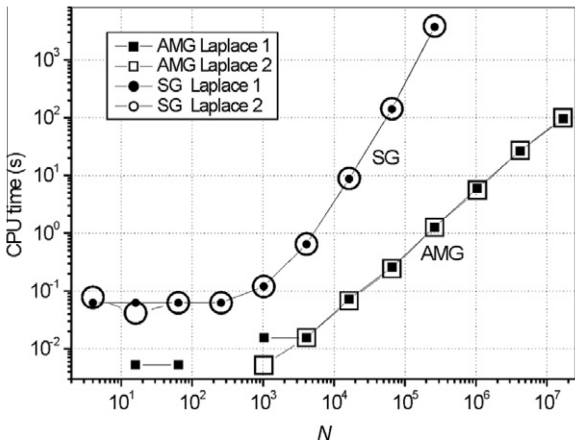

Fig. 6. CPU time *versus* number of unknowns (N) for the AMG and the SG methods using triangular grids.

Table 3
Values of P obtained for Eq. (8).

Problem label	Grid type	AMG	GMG	SG
Laplace 1	Square	1.05	1.09	1.92
Laplace 2	Square	1.00	1.07	1.91
Poisson	Square	1.04	1.04	2.04
Laplace 1	Triangular	1.05	–	1.88
Laplace 2	Triangular	1.04	–	1.88

of the GMG and the AMG algorithms, for the same problem and the same value of N, the former is about 5 times faster than the latter. This behavior is in agreement to the ones previously obtained by Watanabe et al. [13]. The behavior of the CPU time, for both SG and AMG methods, presented in Fig. 6, is also in concordance to other works available in the literature for triangular grids [16].

4.6. Standard versus optimized AMG algorithms

The last subsections were dedicated to the achievement of the optimum parameters of the AMG algorithm, which minimize the CPU time. The basis numerical code applied was the AMG1R6 one, implemented by Ruge and Stüben [7]. In each subsection, only one parameter was investigated, keeping the other ones fixed.

The values listed in Table 4 are the most common ones found in literature, although most works present the values used for each AMG parameter, none explains clearly why such values are employed. This work tries to fill this lack of information

Table 4
Standard and optimum values for the parameters of the AMG algorithm.

Problem label	Grid type	ν^*	ν	L^* and L	θ^*	θ	ε^*	ε
Laplace 1	Square	1	2	$L_{maximum}$	0.25	0.40	0.35	0.35
Laplace 2	Square	1	2	$L_{maximum}$	0.25	0.25	0.35	0.35
Poisson	Square	1	2	$L_{maximum}$	0.25	0.20	0.35	0.20
Laplace 1	Triangular	1	1	$L_{maximum}$	0.25	0.0625	0.35	0.35
Laplace 2	Triangular	1	1	$L_{maximum}$	0.25	0.0625	0.35	0.35

* Standard values found in literature.

Table 5
Comparison of CPU time between the standard and the optimized algorithms, for grids with $N = 4097 \times 4097 = 16{,}785{,}409$ nodes (square grid) and $N = 2^{24} = 16{,}777{,}216$ nodes (triangular grid).

Problem label	Grid type	$t_{CPU}{}^*$ [sec]	t_{CPU} [sec]	$\frac{t_{CPU}}{t_{CPU^*}}$
Laplace 1	Square	62.2	55.8	0.90
Laplace 2	Square	63.1	56.6	0.90
Poisson	Square	75.6	69.7	0.92
Laplace 1	Triangular	112.8	96.2	0.85
Laplace 2	Triangular	115.1	99.2	0.86

* Standard algorithm.

Table 6
Values of P obtained for Eq. (8) and the number of V-cycles needed for the convergence achievement.

Problem label	Grid type	P^*	P	V-cycles*	V-cycles
Laplace 1	Square	1.06	1.05	10	6
Laplace 2	Square	1.05	1.00	10	6
Poisson	Square	1.12	1.04	13	9
Laplace 1	Triangular	1.08	1.05	29	25
Laplace 2	Triangular	1.07	1.04	30	25

* Standard algorithm.

and provides the optimized values for the AMG parameters, compared to the standard ones, as presented in this section. Table 4 also presents the optimized parameters found in this work.

The comparison of the CPU time between the standard and its counterpart optimized algorithm is presented in Table 5. In the simulations performed here, the same microcomputer and the same compiler options were employed for both the standard and the optimized algorithms. As can be seen in Table 5, the CPU time reduction obtained with the optimum values of the parameters varies from 8% to 15%. Triangular grids, which are related to unstructured grids, were the most benefited by the parameters optimization: the CPU time reduction for both studied problems varies from 14% to 15%.

Another way to compare the performance of the standard and the optimized algorithms is verifying the values of P in the Eq. (8) $t_{CPU} = cN^P$, obtained by least square fitting, for both cases, and the number of V-cycles which are needed for the convergence achievement; these results are presented in Table 6. The optimization of AMG parameters not only reduces the CPU time requirements, but it also reduces the values of P in all the cases. Although the previous results presented by the standard algorithm are close to the theoretical value of $P = 1$, the performance of the optimized algorithm is better, with values of P closer to the unity: these improvement is easier observed for more refined grids. Another consequence of the parameters optimization is the reduction of the V-cycles employed for the algorithm convergence. In all the cases, the number of V-cycles is smaller for the optimized algorithm and its reduction achieves 40% to the Laplace equation, using square grids. Such reduction can be attributed to an association of the increasing of the ν from unity to 2 and the CPU time decreasing, which did not show the expected behavior when compared to the one verified for the number of V-cycles.

5. Conclusion

The optimum parameters for the AMG method, which lead to the CPU time reduction, were determined in this paper. For this purpose, systematical simulations were performed, for each one of the following parameters: the number of inner iterations (ν), the number of grids (L), the grid reduction factor (θ) and the strong dependence factor in the coarse grid (ε). The influence of the optimization of such parameters on the CPU time was analyzed by the use of different grids, which is related to different number of unknowns (N). The basis numerical code applied was the AMG1R6 one, implemented by Ruge and Stüben [7]. The finite difference method was employed to the discretization of the differential equations in square grids, while the finite volume method was used for triangular grids. In all the cases, simulations were up to tolerance (10^{-8}), applied to the non-dimensional residual $\overline{l_2}$-norm (based on the initial guess norm).

Based on the numerical results, the most interesting remarks are given below:

(1) The number of inner iterations (v) strongly influences the CPU time. The optimum values of v are 2 for square grids, and 1 for triangular grids.
(2) The CPU time is strongly dependent of the number of levels (L). For all the problems and both grid types, the optimum value is $L = L_{maximum}$. This result is independent of the grid type (triangular or square grid), or even about the multigrid approach (AMG or GMG).
(3) There is not a standard value for the grid reduction factor (θ) when square grids are employed: the optimum values were equal to 0.20, 0.25 and 0.40 for the Laplace 1, Laplace 2 and Poisson problems, respectively. For triangular grids, otherwise, the optimum value was equal to 0.0625 for both the Laplace 1 and the Laplace 2 problems.
(4) For all cases investigated in this paper, except by the Poisson problem (using square grids), the strong dependence factor in coarse grid (ε) presented as optimum value 0.35. In the exception case, the optimum value equals 0.20.
(5) The optimization of the AMG parameters improves the algorithm performance to values closer to the theoretical ones, reducing the values of P to values closer to the unity in Eq. (8). This optimization is also responsible for a reduction in the CPU time, which can achieve up to 15%, and the decreasing of the number of V-cycles up to 40%, considering the same stop criterion.
(6) This work presents results for grids more refined than the ones available in the literature, for both triangular and square grids.
(7) The performance of the GMG approach is better than the AMG approach: the former takes only 20% of the CPU time spent when compared to the latter methodology.

Acknowledgements

The first author would like to acknowledge the infrastructural support provided by the Laboratory of Numerical Experimentation (LENA) of the Department of Mechanical Engineering (DEMEC), Federal University of Paraná (UFPR), her friends, by all the provided support, and CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, Brazil), by her scholarship. The third author is scholarship of CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico, Brazil). The authors would like also to acknowledge the financial support provided by CNPq, CAPES, Fundação Araucária (Paraná, Brazil) and the Brazilian Space Agency (AEB), by the UNIESPAÇO Program. The authors are also grateful to Dr. Stüben, by providing his AMG1R6 code, which was used as basis for all the simulations and optimizations presented in this work.

References

[1] U. Trottenberg, C. Oosterlee, A. Schüller, Multigrid, Academic Press, San Diego, 2001.
[2] W.L. Briggs, V.E. Henson, S.F. Mccormick, A Multigrid Tutorial, second ed., SIAM, Philadelphia, 2000.
[3] A. Thekale, T. Gradl, K. Klamroth, U. Rüde, Optimizing the number of multigrid cycles in the full multigrid algorithm, Numer. Linear Algebra Appl. 17 (2010) 199–210.
[4] J. Brannick, M. Brezina, R. Falgout, T. Manteuffel, S. McCormick, J. Ruge, B. Sheehan, J. Xu, L. Zikatanov, Extending the applicability of multigrid methods, J. Phys.: Conf. Ser. 46 (2006) 443–452.
[5] A. Brandt, Algebraic multigrid theory: the symmetric case, Appl. Math. Comput. 19 (1986) 23–56.
[6] R.D. Falgout, An introduction to algebraic multigrid, Comput. Sci. Eng. 8 (2006) 24–33.
[7] J.W. Ruge, K. Stüben, Algebraic Multigrid, in: S.F. McCormick (Ed.), Frontiers of Applied Mathematics 3: Multigrid Methods, SIAM, Philadelphia, 1987, pp. 73–132.
[8] A. Bourloux, M.J. Gander, Modern Methods in Scientific Computing and Applications, Kluwer Academic Press, Dordrecht, 2002.
[9] K. Stüben, A review of algebraic multigrid, J. Comput. Appl. Math. 128 (2001) 281–309.
[10] Q. Chang, Y.S. Wong, H. Fu, On the algebraic multigrid, J. Comput. Phys. 125 (1996) 279–292.
[11] U. Langer, D. Pusch, Comparison of geometrical and algebraic multigrid preconditioners for data-sparse boundary element matrices, Lect. Notes Comput. Sci. 3743 (2006) 130–137.
[12] Y. Xiao, S. Shu, P. Zhang, M. Tan, An algebraic multigrid method for isotropic linear elasticity problems on anisotropic meshes, Int. J. Numer. Biomed. Eng. 26 (2010) 534–553.
[13] K. Watanabe, H. Igarashi, T. Honma, Comparison of geometric and algebraic multigrid methods in edge-based finite element analysis, IEEE Trans. Magn. 41 (2005) 1672–1675.
[14] C.-T. Wu, H.C. Elman, Analysis and comparison of geometric and algebraic multigrid for convection–diffusion equations, SIAM J. Sci. Comp. 28 (2006) 2208–2228.
[15] F.O. Campos, R.S. Oliveira, R.W. Santos, Performance comparison of parallel geometric and algebraic multigrid preconditioners for the bidomain equations, Lect. Notes Comput. Sci. 3991 (2006) 76–83.
[16] F.J. Gaspar, J.L. Gracia, F.J. Lisbona, C. Rodrigo, On geometric multigrid methods for triangular grids using three-coarsening strategy, Appl. Numer. Math. 59 (2009) 1693–1708.
[17] F.J. Gaspar, J.L. Gracia, F.J. Lisbona, C. Rodrigo, Efficient geometric multigrid implementation for triangular grids, J. Comput. Appl. Math. 234 (2010) 1027–1035.
[18] F. Oliveira, M.A.V. Pinto, C.H. Marchi, Effect of multigrid method roads on the CPU time for the two-dimensional Laplace equation [in Portuguese], Proc. XXIX Iberian-Latin-American Congress on Computational Methods in Engineering, Maceió-Brazil (2008) pp. 1–21.
[19] C. Iwamura, F.S. Costa, I. Sbarski, A. Easton, N. Li, An efficient algebraic multigrid preconditioned conjugate gradient solver, Comput. Methods Appl. Mech. Eng. 192 (2003) 2299–2318.
[20] A. Krechel, K. Stüben, Operator dependent interpolation in algebraic multigrid, Lect. Notes Comput. Sc. Eng. 3 (1998) 189–211.

[21] R. Suero, M.A.V. Pinto, C.H. Marchi, L.K. Araki, A. C. Alves, Optimization of Algebraic Multigrid Method to Two-dimensionals Laplace and Poisson Equations [in Portuguese], Proc. VI National Congress of Mechanical Engineering (CONEM), Campina Grande-Brazil (2010) pp. 1–10.
[22] F.P. Incropera, D.P. DeWitt, T.L. Bergman, A.S. Lavine, Fundamentals of Heat and Mass Transfer, sixth ed., John Wiley & Sons, 2007.
[23] J.C. Tannehill, D.A. Anderson, R.H. Fletcher, Computational Fluid Mechanics and Heat Transfer, second ed., Taylor & Francis, Washington, 1997.
[24] H. Versteeg, W. Malalasekera, An Introduction to Computational Fluid Dynamics: The Finite Volume Method, second ed., Pearson Education Limited, Harlow, 2007.