Wavelet-Based Algebraic Multigrid Method Using the Lifting Technique

Abstract —
The multigrid method has been applied to electromagnetic field problems yielding a significant reduction in computational time in comparison with conventional linear solvers. Although the algebraic approach of the method is very appropriate when the problem involves both irregular domains and unstructured meshes, it presents some difficulties dealing with complex systems, in which the coefficient matrix violates the M-matrix property, and parallelism. Trying to overcome these problems, a new wavelet-based algebraic multigrid method was proposed recently which proved to be more efficient than the standard approach in some cases. This paper presents a progress in the development of this recent method, using the lifting technique for creating an algorithm with smaller memory requirement and a reduced number of floating point operations. The approach is tested in the TEAM 28 Problem and the Incomplete Cholesky and Incomplete LU preconditioners are used for comparison.

I. Introduction

The multigrid method (MG) is a well-grounded numerical technique for solving large sparse linear systems of algebraic equations. The method can be used as an iterative solver or as a preconditioner for one appropriate iterative method. This technique explores the use of some levels to eliminate the high-frequency or oscillatory components of the error, which are not efficiently removed by basic iterative methods, as Jacobi or Gauss-Seidel [1].

In contrast to the standard geometric multigrid method, which demands explicitly a hierarchy of meshes, the algebraic multigrid (AMG) constructs a hierarchy of matrices and transfer operators just based on the information from the original matrix [2]. Thus, when the problem involves irregular domains and unstructured meshes, or when the interest is focused on the so called black-box solver which requires only information contained in the matrix of the system, or even when the matrix is very large and ill-conditioned as in 3-D electromagnetic problems, the AMG method is very appropriate. However, the use of AMG presents some difficulties, especially regarding to the choice of the strength threshold in the coarsening process.

The coarsening scheme, for traditional AMG, can lead to computational complexity growth as the problem size increases, resulting in an elevated memory use and execution time, and in a reduced scalability [3]. Moreover, the coarsening process is inherently sequential in nature that makes difficult the implementation on distributed memory machines [4]. On the other hand, classical AMG cannot directly be applied to problems in which the coefficient matrix violates the M-matrix property like, for example, ungauged edge-based finite-element (FE) analysis. In this context, simple approaches such as the use of a shifted coefficient matrix have a limited efficiency [5].

The combination of wavelet and the algebraic multigrid has been explored to overcome these difficulties. The similarities between the multigrid methods and wavelets arising from multiresolution analysis were first brought out by Briggs and Henson [6]. They observed that the space of fine grid vectors in multigrid scheme and the space of highest resolution in multiresolution are correlated.

Over the last few years, several works are exploiting these similarities [7–14]. In [7], it is made an exploration of the idea of Briggs to develop wavelet based interpolation and restriction operators in the geometric multigrid context. It compares the results with the geometric multigrid schemes using the conventional intergrid operators, and obtains results similar to the ones obtained with the standard procedure.

The use of the discrete wavelet transform (DWT) in the construction of the matrices hierarchy and the transfer operators in the AMG method was proposed by these authors in [11], producing a new method called WAMG. In that approach a modified DWT, using only low-pass filter bank, was applied to produce an approximation of the original matrix in each level of the multiresolution process, eliminating the standard coarsening process.

The WAMG has revealed to be a very efficient and promising method for several problems related to the computation of electromagnetic fields, in both serial and parallel computation [11], [13, 14]. The method can be either used as an iterative solver or as a preconditioning technique, presenting in many cases a better performance than some of the most advanced and current AMG algorithms [14].

Due to the WAMG efficiency and potentiality, further researches have been carried out for its improvement [15], [16]. This paper is part of this effort.

In order to accomplish this task this work build a modified discrete wavelet transform using the tool called the lifting scheme [17]. The lifting technique is a method introduced by W. Sweldens [18], which allows some improvements on the properties of existing wavelet transforms. The technique has some numerical advantages as a reduced number of floating point operations which are fundamental in the context of the iterative solvers.

II. The Algebraic Multigrid Method

Denoting for V_k the space of the variables of the system in a level k, and for V_{k+1} one subspace of V_k ($V_{k+1} \subset V_k$), the algorithm of the standard AMG consists of the following components [1, 2]:

- *Coarsening process*: define the splitting $\omega_k = \omega_C \cup \omega_F$ where ω_k is the set of all the variables of the system in level k, ω_C and ω_F are, respectively, the set of the variables that will and will not form the subspace V_{k+1}, all represented by their respective indexes;

- *Transfer Operators*:

prolongation $\quad \mathrm{P}_{k+1}^{k}: V_{k+1} \rightarrow V_k \quad$ (1)

restriction $\quad \mathrm{P}_{k}^{k+1} = [\mathrm{P}_{k+1}^{k}]^{\mathrm{T}} \quad$ (2)

- *Definition of the hierarchy of matrices*:

$$\mathrm{A}^{k+1} = \mathrm{P}_{k}^{k+1} \mathrm{A}^{k} \mathrm{P}_{k+1}^{k} \tag{3}$$

- *Appropriate smoother*: basic iterative Jacobi-type method that eliminates the oscillatory components of the error.

Overall, the first two components described above, are the most important of the AMG setup phase [1, 2]. These stages determine the subspace dimension, and, consequently, the matrix dimension in the corresponding level.

The splitting defined in the coarsening process is based on a concept of influence and dependence, formalized in the following definitions [1]:

Definition 1. Given a threshold value $0 < \alpha < 1$, the variable (point) u_i *strongly depends* on the variable (point) u_j if

$$\left|a_{ij}\right| \geq \alpha \cdot \max_{k \neq i} \left|a_{ik}\right| \tag{4}$$

that is, the grid point *i* strongly depends on grid point *j* if the coefficient a_{ij} is comparable in magnitude to the largest off-diagonal coefficient in the *i*th equation.

Definition 2. If the variable u_i strongly depends on the variable u_j, then the variable u_j *strongly influences* the variable u_i.

This standard procedure of the AMG presents two great difficulties in relation to the coarsening process, that is, the stage of selection of the subsystem variables. The first great difficulty is the choice of the parameter α, present in the equation (4). The value of this parameter strongly influences the quality of the process. Usually, this parameter is fixed in 0.25, but this choice not always produces the best results. The definition of adaptive value to the strength threshold α, as done in [19], can overcome this difficulty in some cases.

The second difficulty is related to the parallelization of the AMG method in distributed memory computers. The coarsening process is included in the core of the AMG setup phase and is inherently sequential in nature. Moreover, The coarsening scheme can lead to computational complexity growth as the problem size increases, resulting in an elevated memory use and execution time, and in a reduced scalability Some works have exclusively been looking for the solution of this important problem [20-22].

III. The Wavelet-based Algebraic Multigrid

The key point of the WAMG is the application of a modified (incomplete) DWT, as a filter bank with only low-pass filters, to generate the hierarchy of matrices in the AMG method.

The standard DWT corresponds to the application of low-pass and high-pass filters, followed by the elimination of one out of two samples (decimation or sub sampling). The discrete signal, which in one dimension is represented by a vector of values, is filtered by a set of digital filters that are associated to the wavelet adopted in the analysis. Then, starting from a vector $y(N)$ at level 0, two sets of coefficients are generated in each level *l* of the process: a set d_l of wavelets coefficients (detail coefficients) and a set c_l of coefficients called "approximation coefficients." This procedure can be applied again, now using c_l as an input vector to create new coefficients c_{l+1} and d_{l+1}, and successively. This procedure is illustrated in Fig. 1. The symbol $\downarrow 2$ represents the operation of decimation by 2. This operation keeps unaltered the vector dimension at the end of the process. Blocks H and G represent, respectively, the low-pass and high-pass filters.

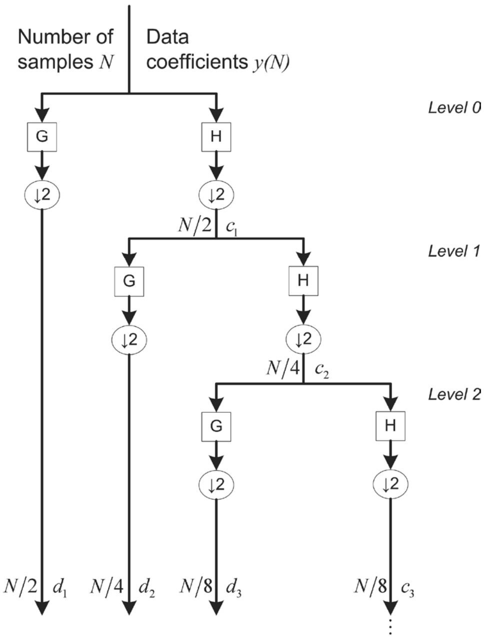


Fig. 1. Three levels of the one-dimensional discrete wavelet transform

In the 2-D case, in which the discrete signal is represented by a matrix, the DWT is obtained through the application of successive steps of 1-D transform into the rows and columns of the matrix. This process generates a matrix formed by four types of coefficients: the approximation coefficients and the detail coefficients (horizontal, vertical, and diagonal), as illustrated in Fig. 2.

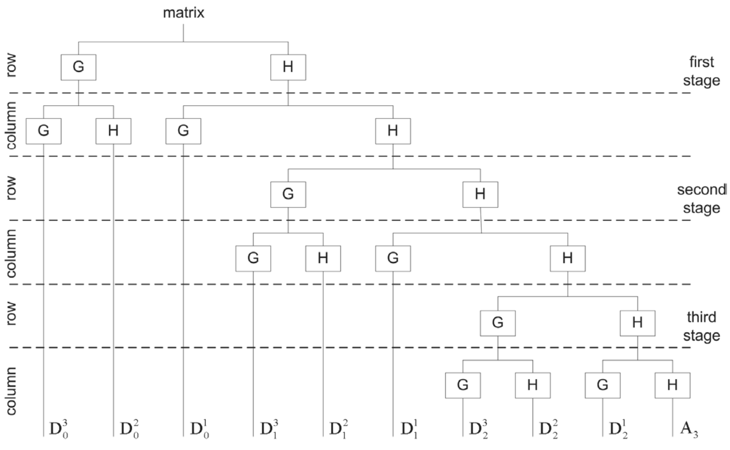


Fig. 2. The two-dimensional DWT: one-dimensional transform in the rows and columns of the matrix

In both cases, one and two dimensions, the approximation coefficients keep the most important information of the discrete signal, whereas the detail coefficients possess very small values, next to zero. These approximation coefficients will contain low-pass information, which is essentially a low resolution version of the signal and represent a coarse version of the original data.

This wavelet property is explored by WAMG for creating the hierarchy of matrices. In this approach the low-pass filters coefficients h_i are used in the construction of a matrix *R*, which is used as a restriction operator for the WAMG. If third order filters are used, for example, the restriction operator *R* from level *l* to level *l*+1 takes the following form:

$$R_l^{l+1} = \begin{pmatrix} h_3 & h_2 & h_1 & h_0 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & 0 & h_3 & h_2 & h_1 & h_0 & 0 & \cdots & 0 \\ \vdots & & & & \vdots & & & & \vdots \\ h_1 & h_0 & 0 & \cdots & \cdots & \cdots & 0 & h_3 & h_2 \end{pmatrix} \qquad (1)$$

The prolongation operator *P*, in the corresponding level, is defined in the usual form [5],

$$P_{l+1}^{l} = (R_l^{l+1})^{\mathrm{T}} \,. \qquad (2)$$

Therefore, the WAMG setup phase depends only of the choice of the filters coefficients. This approach is very interesting mainly because it avoids the coarsening process and the heuristic parameters present in the standard AMG, simplifying the use of the method as well as its parallel implementation in distributed memory computers [11, 14]. Moreover, the WAMG setup time is strongly reduced, since it is not necessary to compute the full wavelet transform in each level and the DWT is limited to the computation of a sequence of approximation coefficients.

A. The choice of the wavelet filters

Very important classes of filters are those of Finite Impulse Response (FIR). The main characteristic of these filters is the convenient time-localization properties. These filters are originated from wavelets with compact support and are such that,

$$h_n = 0 \quad \text{for} \quad n < 0 \ \text{ and } \ n > L. \qquad (3)$$

in which *L* is the length of the filter.

Minimal requirements for these compact FIR filters are:

1. The length of the scaling filter h_n must be even.
2. $\sum_n h_n = \sqrt{2}$
3. $\sum_n (h_n \cdot h_{n-2k}) = \delta(k)$,

in which $\delta(k)$ is the Kronecker delta, such that is equal to 1 when *k*=0 or 0 when *k*=1.

These filters are calculated analytically and some of them are described as follows:

i. Length 2 scaling filter: Haar or Daubechies-2 coefficients

$$h_{\mathrm{D2}} = [h_0, h_1] = \left[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right] \qquad (4)$$

ii. Length 4 scaling filters: Daubechies-4 coefficients

$$h_{\mathrm{D4}} = \left[\frac{1+\sqrt{3}}{4\sqrt{2}}, \frac{3+\sqrt{3}}{4\sqrt{2}}, \frac{3-\sqrt{3}}{4\sqrt{2}}, \frac{1-\sqrt{3}}{4\sqrt{2}}\right] \qquad (5)$$

For more details about compact FIR filters see [19].

A common problem on the choice of the filters is to decide between the fill-in control and the wavelet compression properties. As the WAMG often deals with sparse matrices, the control of the nonzero number is a very important task. In this case, if the WAMG level *l* matrix A^l is sparse, then the number of nonzero elements in the next level matrix $A^{l+1} = R_l^{l+1} A^l P_{l+1}^l$ will depend on the order of the filter used in the restriction and prolongation operators. In fact, the longer the used filter, the larger will be the number of nonzero entries in the next computed matrix. Consequently, most of the wavelet based multigrid methods use shorter filters such as Haar or Daubechies-2 coefficients in its approaches [11-14]. This is also the case in this paper.

IV. The implementation using the lifting technique

In [24] Daubechies and Sweldens have shown that every wavelet filter can be decomposed into lifting steps. Therefore, all discrete wavelet transforms used for WAMG can be implemented with the lifting scheme. The main advantages of this technique over the classical wavelet transform are:

a) Smaller memory requirement – the calculations can be performed in-place;
b) Efficiency: reduced number of floating point operations;
c) Parallelism –inherently parallel feature;
d) Easier to understand - not introduced using the Fourier transform;
e) Easier to implement;
f) The inverse transform is easier to find – it has exactly the same complexity as the forward transform;
g) Transforms signals with an arbitrary length (need not be 2^n) – very appropriate in the WAMG context;
h) Transforms signals with a finite length (without extension of the signal).

More details about these advantages as well as other important structural advantages of the lifting can be found in [17, 18].

In fact, there are basically three forms for representing a wavelet transform: equation form (lifting), filter form (filter bank) and matrix form. However, only the first two are appropriate in the multigrid implementation.

The representations of the Daubechies 2 and Daubechies 4 wavelets in the lifting form are presented, respectively, in equations (6) and (7). These representations were extracted from [17], which shows the details for converting between lifting and filters forms (one can also see [24]).

$$
\begin{aligned}
& d_{l+1}^{(1)}[n] = c_l[2n+1] - c_l[2n], \\
& c_{l+1}^{(1)}[n] = c_l[2n] + \frac{1}{2} \cdot d_{l+1}^{(1)}[n], \\
& c_{l+1}[n] = \sqrt{2} \cdot c_{l+1}^{(1)}[n], \\
& d_{l+1}[n] = \frac{1}{\sqrt{2}} \cdot d_{l+1}^{(1)}[n].
\end{aligned}
\tag{6}
$$

$$
\begin{aligned}
& c_{l+1}^{(1)}[n] = c_l[2n] + \sqrt{3} c_l[2n+1], \\
& d_{l+1}^{(1)}[n] = c_l[2n+1] - \frac{1}{4}\sqrt{3} c_{l+1}^{(1)}[n] - \frac{1}{4}(\sqrt{3}-2) c_{l+1}^{(1)}[n-1], \\
& c_{l+1}^{(2)}[n] = c_{l+1}^{(1)}[n] - d_{l+1}^{(1)}[n+1], \\
& c_{l+1}[n] = \frac{\sqrt{3}-1}{\sqrt{2}} c_{l+1}^{(2)}[n], \\
& d_{l+1}[n] = \frac{\sqrt{3}+1}{\sqrt{2}} d_{l+1}^{(1)}[n].
\end{aligned}
\tag{7}
$$

The coefficients $c_{l+1}[n]$ and d_{l+1} $[n]$ in (6) and (7) are, respectively, the approximation and the detail coefficients, at level l+1, of the input signal. Therefore, for the multigrid implementation the last equations in (6) and (7) are not necessary.

In this work, the WAMG method was implemented using the lifting technique, with length 2 Daubechies filters, producing a method called LAMG. The new approach was tested in some numerical problems and the results are shown in sections 5 and 6.

A. The cost of the lifting algorithm

The use of the lifting technique for the wavelet-based multigrid allows creating an algorithm with a reduced number of floating point operations. It is known that the lifting algorithm is asymptotically twice as fast as the standard algorithm for long filters [17]. A comparison between the cost of the lifting algorithm (*L*) and the cost of the standard algorithm (*T*) and the relative speedup (*T*/*L*-1) for the Daubechies filters are presented in Table I.

TABLE I. LIFTING (*L*) VERSUS THE STANDARD ALGORITHM (*T*)

Wavelet	*T*	*L*	*Speedup*
Daubechies 2	3	3	0 %
Daubechies 4	14	9	56 %
Daubechies 6	22	14	57 %

The values of the cost presented in Table I are measured in number of performed multiplications and additions.

V. BENCHMARK

The performance of the proposed approach is verified by solving two different numerical test problems. In the first test, the LAMG is applied as a preconditioner for the iterative methods Bi-Conjugate Gradient Stabilized (BiCGStab) and Generalized Minimum Residual method with restart parameter m = 20 (GMRes-20) to solve two sparse linear systems issued from University of Florida Sparse Matrix Collection, FEMLAB group [25]. These test problems arise in real 3D computational fluid dynamics problems and its matrix properties are presented in Table II.

The results in Table III give us the setup and solver times (in seconds) spent by the methods and number of iterations necessary to reduce the Euclidean norm of the residual vector to the order of 10^{-6}. The Incomplete LU (ILU) preconditioner was used for comparison.

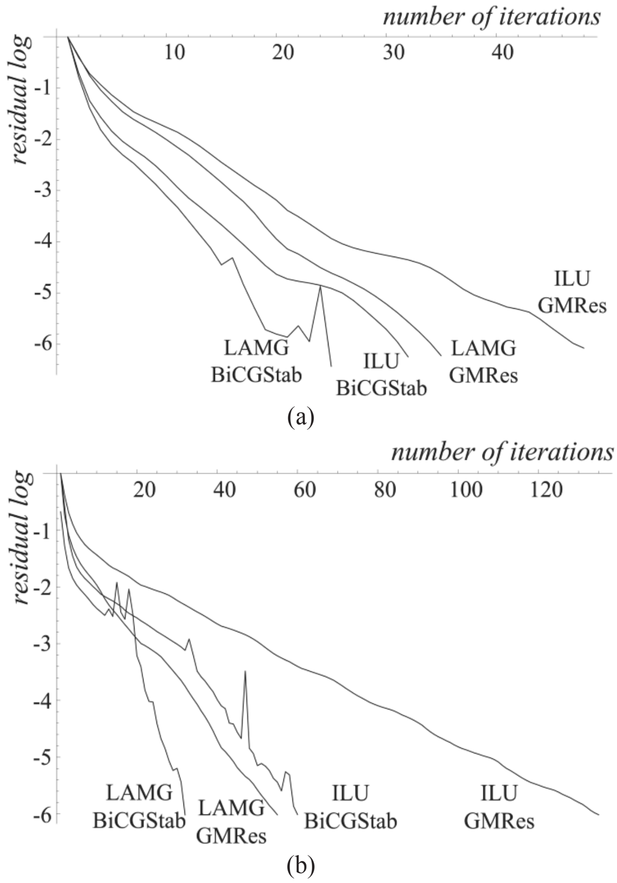
TABLE II. MATRIX PROPERTIES FOR THE FIRST TEST

Properties	*Matrices*	
	poisson3Da	*poisson3Db*
number of rows	13514	85623
number of columns	13514	85623
nonzeros	352762	2374949
Type	real unsymmetric	real unsymmetric
Cholesky candidate?	No	No
positive definite?	No	No

TABLE III. RESULTS FOR THE FIRST TEST PROBLEM

Method	*poisson3Da*			*Poisson3Db*		
	Setup	*Solver*	*Iterations*	*Setup*	*Solver*	*Iterations*
ILU BiCGStab	1.22	9.60	31	9.28	133.04	59
ILU GMRes		8.06	49		163.70	140
LAMG BiCGStab	2.73	7.00	24	34.35	83.54	32
LAMG GMRes		5.49	35		77.72	56

The convergence history for this first case is presented in Fig. 3.



(a)

(b)

Fig. 3. Convergence history for first test problem – poisson3Da (a) and poisson3Db (b).

The second test problem in related to the three-dimensional Poisson equation defined in a uniform grid on the cube $[-1,1]^3$. The spatial discretization of the Poisson equation uses Lagrange finite-element functions and second order elements, both implemented in the C++ Finite Element Library - LibMesh [26]. The resulting symmetric positive definite matrix has 29791 rows and 1771561 nonzero entries.

In this case, the Algebraic Multigrid method with the lifting implementation was used as a stand-alone solver and as a preconditioner for the iterative Conjugate Gradient (CG) method and for BiCGStab. The results are shown in Table IV. For comparison, the standard Incomplete Cholesky (IC) preconditioner was used in this case.

TABLE IV. NUMERICAL RESULTS FOR THE SECOND TEST PROBLEM

Method	*Number of steps*	*Time in seconds*		
		Setup	*Solver*	*Total*
LAMG	5	3.65	3.85	7.50
LAMG-BiCGStab	3		4.58	8.23
LAMG-CG	3		4.04	7.69
IC-CG	10	6.78	2.59	9.37
IC-BiCGStab	5		2.08	8.86

VI. APPLICATION IN THE TEAM 28 PROBLEM

The performance of the proposed approach is also verified in the steady-state analysis of Compumag TEAM 28 Problem [27]. This problem relates to the modeling of an electrodynamic device which consists of two stationary concentric exciting coils interacting with a moveable round conducting plate. The used model, which was created using the Finite Element Method Magnetic (FEMM) [28], applies the Kelvin transformation [29] for investigating the steady-state levitation height of the plate. The resulting complex symmetric matrix has 54723 rows and 380957 nonzero entries.

The LAMG method was used as a preconditioner for BiCGStab. Again, the results were compared with those obtained by the incomplete decompositions preconditioners IC and ILU. The results are shown in Table V.

The magnetic flux density B (module) for this problem and the convergence history of the iterative methods are presented in Fig. 4 and 5, respectively. The stopping criterion for this case was to reduce the residual norm by 4 orders of magnitude.

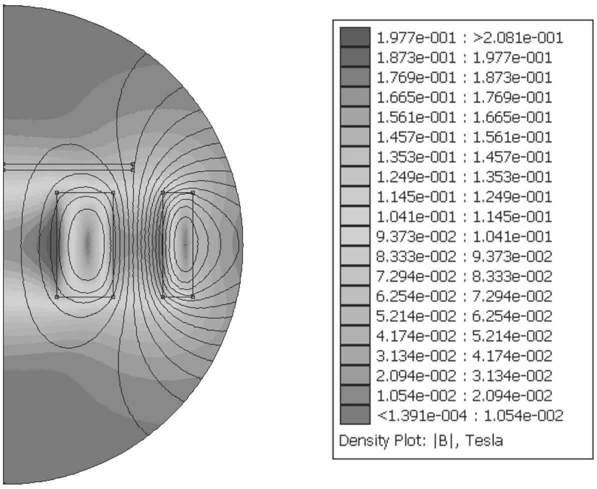


Fig. 4. Magnetic flux density B (module)

TABLE V. NUMERICAL RESULTS FOR TEAM 28 PROBLEM.

Method	*Number of steps*	*Time in seconds*		
		Setup	*Solver*	*Total*
LAMG BiCGStab	42	1.11	15.92	17.03
ILU-BiCGStab	244	0.36	81.17	81.53
IC-BiCGStab	223	0.58	33.99	34.57

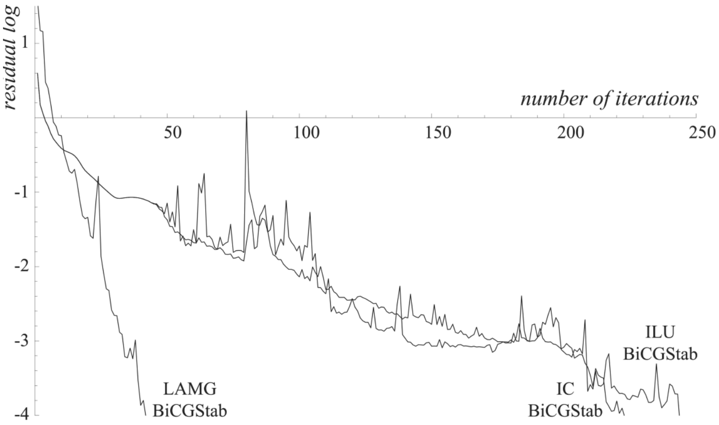


Fig. 5. Convergence performance for TEAM 28 Problem

VII. A PARALLEL WAVELET-BASED ALGEBRAIC MULTIGRID

Especially in the context of large scale problems and massively parallel computing, the most desirable property of the multigrid approaches is its potential for algorithmic scalability: in the ideal case, for a matrix problem with n unknowns, the number of iterative V-cycles required for convergence is independent of the problem size n and the work in the setup phase and in each V-cycle is linearly proportional to the problem size [3, 4]. For all this, the need to solve linear systems arising from problems posed on extremely large, unstructured grids has been generating great interest in parallelizing algebraic multigrid.

This section presents a parallel algorithm for Wavelet-based algebraic multigrid (PWAMG) using a variation of the parallel implementation of discrete wavelet transforms. As stated, this approach eliminates the grid coarsening process present at standard setup phase, simplifying significantly the implementation on distributed memory machines and allowing the use of PWAMG as a parallel "black box" solver and preconditioner.

The first stage of the multiresolution process is calculated using a parallel multiple 1D DWT. It means that rows of the matrix are distributed among the processors in such a way that the 1D DWT may be calculated efficiently in the directions of the columns. Thus the resulting matrix will be assembled in a way that the second stage is calculated entirely locally [30].

In order to ensure no communication among processors in the setup phase, we distributed the rows of the matrix in blocks of 2^k , $k \in \mathbb{Z}$.

In the solver phase the communications among processors are necessary only for operations involving matrices. In a matrix vector product $A \cdot b = c$, for example, the matrix A is distributed by rows, the vector b is shared by all processors and the vector c is calculated in parallel as illustrated in Fig. 6, for 4 processors. Then the resulting vector c is updated by the processors through MPI message passing. This task is accomplished by using the MPI collective communication function MPI_Allgather [31]. The MPI_Allgather function effect is show in Fig. 7.

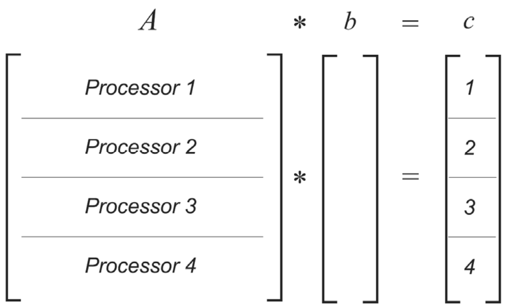


Fig. 6. The proposed parallel matrix-vector product

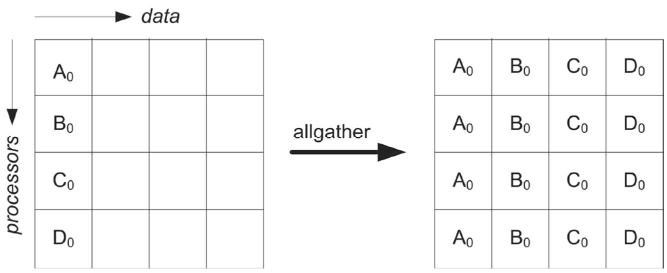


Fig. 7. The effect of the `MPI_Allgather` function

VIII. Description of the Parallel Test Problems

In this section, we present some numerical examples concerning the solution of the Poisson equation in two dimensions (2D). Owing to its scientific importance and challenge, the development of new and efficient methods to solve the Poisson equation is always welcome.

The Poisson equation describes the electrostatic potential caused by a fixed charge distribution and plays a fundamental role in many physical processes.

In the first test problem, we consider the Poisson equation on a unit square with Dirichlet boundary conditions. In this case, the Poisson operator is discretized by the standard 5-point difference stencil (8)

$$L_h = \frac{1}{h^2}\begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix}_h \qquad (8)$$

where h is the internodal spacing.

In the next test, the same Poisson problem is solved with the high-order 9-point difference stencil (9)

$$L_h^{(9)} = \frac{1}{6h^h}\begin{bmatrix} -1 & -4 & -1 \\ -4 & 20 & -4 \\ -1 & -4 & -1 \end{bmatrix}_h . \qquad (9)$$

Finally, we solved the anisotropic equation (10) in the same unit square domain with Dirichlet boundary conditions. This is an important test because many physical problems are strongly anisotropic.

$$-\varepsilon u_{xx} - u_{yy} = f \qquad (10)$$

The discrete operator for this problem was obtained by the following standard 5-Point difference stencil

$$L_h^{(5)}(\varepsilon) = \frac{1}{h^2}\begin{bmatrix} & -1 & \\ -\varepsilon & 2(1+\varepsilon) & -\varepsilon \\ & -1 & \end{bmatrix} \qquad (11)$$

with $\varepsilon = 0.01$. This same operator in (11) can be gotten using the standard 5-point stencil for the pure Poisson operator on a stretched grid with $h_x = h_y/\sqrt{\varepsilon}$, where h_x and h_y are respectively the internodal spacing in x and y directions.

All these problems were presented by Chang e Huang in [32]. The authors proposed several new approaches for improving the AMG method and they used these problems to evaluate the performances of their AMG algorithms. The numerical results presented by them will be literally transcribed here for comparison.

IX. Parallel Numerical Results

The parallel algorithm uses the version two of the Message Passing Interface (MPI-2) that provides a standard for message passing for parallel computers and workstation clusters.

The parallel implementation is based on the Discrete Wavelet Transform with filters of Daubechies of length 2, avoiding any communication among processors in the setup phase. A hybrid Jacobi-Gauss method was used as smoother and the V-cycle for the resolution scheme.

The PWAMG has been implemented using in C++ and tested in a homogeneous Beowulf cluster with 4 machine nodes connected to the switch with Gigabit network, as illustrated in Fig. 8. The characteristics of the machines are detailed in Table VI.

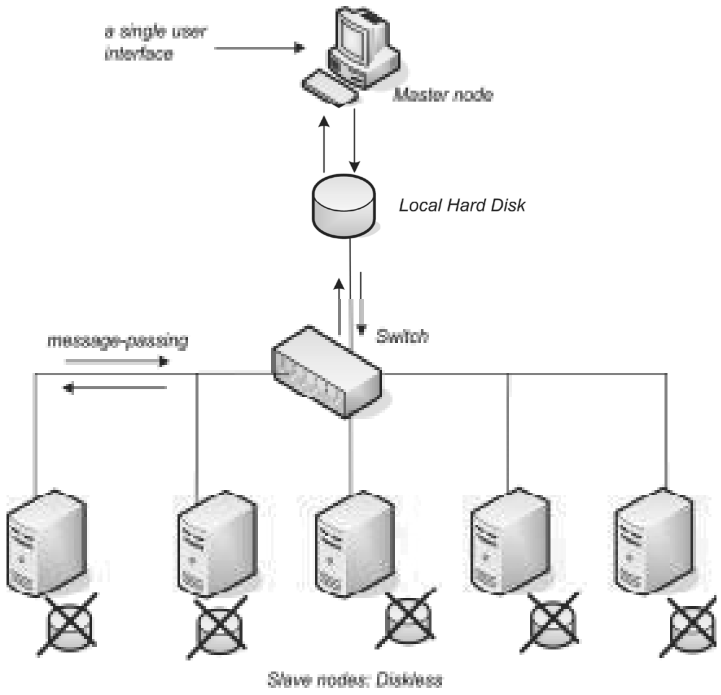


Fig. 8. The Beowulf LINUX Cluster architecture

TABLE VI. Cluster Machines Characteristics

processor	*Machine characteristics*
1	AMD Athlon(tm) XP 2400+ 1.99 GHz 1 GB RAM
2	AMD Athlon(tm) XP 2400+ 1.99 GHz 1 GB RAM
3	AMD Athlon(tm) XP 2400+ 1.99 GHz 1 GB RAM
4	AMD Athlon(tm) XP 2400+ 1.99 GHz 1 GB RAM

The PWAMG method has been tested in the problems described in the last section, for different numbers of finite difference mesh nodes. Comparisons are made with the sequential version of this method and with some of the most advanced and current AMG algorithms presented in [32]. So, the results reported here use the following notation that is the same notation used in that work:

ρ asymptotic convergence factor
np number of processors
t_s computing time for the solution phase
t_m computing time for the setup phase
N number of iterations for convergence
EQ total number of matrix equations

In all cases, the convergence is defined by $\| r^N \| / \| b \| \leq 10^{-6}$, where r^N is the residual vector at the *N*th iteration and the right hand side vector *b* is chosen so that the solution is a vector with all the elements equal to 1.0.

Like in [32], only one smoothing step is applied before and after coarse-grid correction steps.

Table VII gives the numerical results for the first test problem. In this case, finite difference meshes with 4096 and 16384 nodes have been used and the resulting linear systems solved with a sequential version of PWAMG method. The results are compared with those achieved by the single processor algorithms presented in [32] (methods I-VIII).

TABLE VII. Numerical Results For L_h

Method	***EQ***	ρ	***N***	t_m	t_s
I	4096	0.021	4	0.17	0.12
II		0.017	4	0.16	0.10
III		0.062	5	0.05	0.16
IV		0.078	6	0.06	0.16
V		0.020	4	0.11	0.11
VI		0.017	4	0.11	0.17
VII		0.018	4	0.11	0.11
VIII		0.018	4	0.11	0.11
WAMG		0.010	3	0.04	0.16
I	16384	0.022	4	0.61	0.61
II		0.017	4	0.71	0.40
III		0.079	6	0.28	0.50
IV		0.087	6	0.45	0.55
V		0.020	4	0.44	0.43
VI		0.017	4	0.61	0.50
VII		0.017	4	0.44	0.49
VIII		0.017	4	0.44	0.54
WAMG		0.010	3	0.15	0.65

The computational results obtained by PWAMG for this same problem in meshes with larger number of nodes are given in Table VIII.

The results for time presented in all tables are given separately for setup phase and solver phase. These times are in seconds and have been presented for PWAMG, in tables III, IV and V, in the form t_1/t_2, where t_1 is the processing time returned by the C `clock()` function and t_2 is the total time spent in the corresponding phase, which includes the MPI communication time and is measured using the MPI function `MPI_Wtime()`. Of course, for single processor algorithms the times t_1 and t_2 are the same.

Table VIII shows that the setup time results from `MPI_Wtime()` and `clock()` functions are very similar. This indicates the absence of communication in the setup phase that represents a great advantage of the approach proposed here.

TABLE VIII. Pwamg Method For The First Test Problem (8)

EQ	***np***	***N***	t_m	t_s
4096	1	3	0.04	0.15
	2	3	0.02/0.03	0.06/0.14
			0.02/0.02	0.02/0.14
	4	3	0.01/0.01	0.04/0.17
			0.01/0.01	0.01/0.17
			0.01/0.01	0.02/0.17
			0.01/0.01	0.01/0.17
16384	1	3	0.15	0.65
	2	3	0.08/0.11	0.23/0.35
			0.08/0.08	0.14/0.35
	4	3	0.04/0.09	0.20/0.33
			0.03/0.03	0.11/0.32
			0.03/0.03	0.10/0.32
			0.03/0.03	0.08/0.32
65536	1	3	0.63	2.70
	2	3	0.32/0.36	1.09/1.46
			0.30/0.30	0.77/1.45
	4	3	0.17/0.24	0.90/1.36
			0.16/0.16	0.49/1.36
			0.14/0.14	0.55/1.36
			0.13/0.14	0.52/1.35
262144	1	3	2.57	10.76
	2	3	1.32/1.46	4.58/5.17
			1.25/1.25	3.61/5.13
	4	3	0.73/0.88	3.42/5.07
			0.69/0.69	2.28/5.01
			0.59/0.59	2.36/4.99
			0.60/0.60	2.27/4.98
1048576	1	3	10.54	43.68
	2	3	5.38/5.70	18.05/20.71
			5.12/5.12	13.76/20.56
	4	3	2.95/3.22	13.23/19.83
			2.83/2.83	9.60/19.73
			2.40/2.40	8.34/19.65
			2.40/2.40	9.24/19.73

The PWAMG method was also applied to the other two problems. A finite difference mesh with 16384 nodes was used for the Poisson equation discretized with the 9-point stencil (9) and another mesh with 4096 nodes for the anisotropic problem. In both cases, the results obtained are compared again with those results of the methods in [32] and with the WAMG. These results are given in Tables IX and X.

From the numerical results, we can observe the efficiency of the PWAMG to solve the Poisson equation. It is interesting because the solution of the Poisson equation is a nontrivial task, since it involves large linear matrix equations.

TABLE IX. NUMERICAL RESULTS FOR THE SECOND TEST PROBLEM

Method	*np*	*N*	t_m	t_s
I	1	6	0.55	0.60
II	1	5	0.66	0.44
III	1	6	0.44	.055
IV	1	6	0.50	0.52
V	1	5	0.60	0.51
VI	1	5	0.72	0.51
VII	1	5	0.60	0.50
VIII	1	5	0.52	0.50
PWAMG	1	3	0.16	0.72
	2	3	0.08/0.12	0.24/0.38
			0.09/0.09	0.16/0.38
	4	3	0.05/0.06	0.22/0.35
			0.04/0.04	0.14/0.34
			0.04/0.04	0.12/0.34
			0.04/0.04	0.10/0.34

TABLE X. NUMERICAL RESULTS FOR ANISOTROPIC PROBLEM

Method	*np*	*N*	t_m	t_s
I	1	4	0.16	0.16
II	1	4	0.16	0.11
III	1	4	0.06	0.11
IV	1	5	0.11	0.05
V	1	4	0.11	0.16
VI	1	3	0.11	0.10
VII	1	3	0.11	0.13
VIII	1	3	0.11	0.16
PWAMG	1	3	0.05	0.23
	2	3	0.02/0.03	0.09/0.22
			0.02/0.02	0.03/0.22
	4	3	0.01/0.01	0.06/0.26
			0.01/0.01	0.01/0.26
			0.01/0.01	0.02/0.26
			0.01/0.01	0.01/0.26

As usual, the parallel performance analysis for the results presented in tables above was characterized using the absolute speedup which is defined as

$$S_p = \text{speedup}(p) = \mathbf{time}(1)/\mathbf{time}(p) \qquad (11)$$

in which **time**(1) is the time spent by the best sequential WAMG algorithm and **time**(*p*) is the time required by the parallel method executing on *p* processors [33].

The speedup for setup and solver times and the speedup total are illustrated in Fig. 9 for the results presented in Table VIII. The efficiency of the parallel method for these cases is reported in Table XI. Efficiency is a performance metric which estimates how well-utilized the processors are in solving the problem and it is defined as follow [33]:

$$E_p = S_p/p \qquad (10)$$

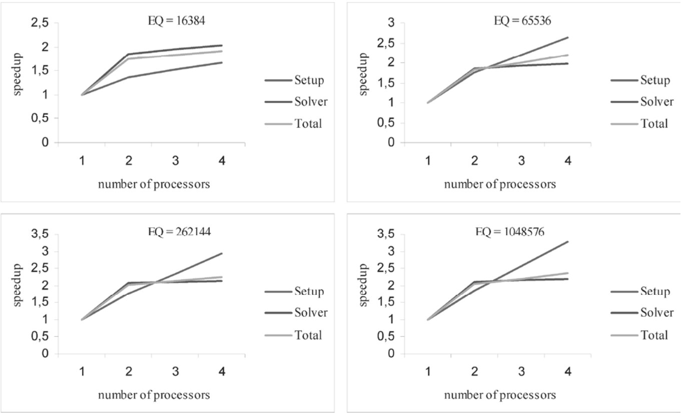


Fig. 9. Parallel speedup for first test problem

TABLE XI. EFFICIENCY OF THE PARALLEL METHOD FOR THE FIRST PROBLEM

np	*Number of equations*			
	16384	65536	262144	1048576
2	0,87	0,91	1,00	1,03
4	0,48	0,55	0,56	0,59

According to Table XI, the parallel method was more efficient for bigger problems, as is characteristic of the multilevel approaches. For these bigger problems the parallel efficiency was about 60%. When 2 processors were used the method reached a little more than the linear speedup. However, for 4 processors, the speedup was less than the linear as can be seen in Fig. 9.

X. CONCLUSIONS

The proposed approach seems to be very promising. The use of the lifting technique allows creating an efficient preconditioner, as can be seen from the number of LAMG steps in Tables III, IV and V. It is important to mention that the LAMG can be applied as a preconditioner for both symmetric and unsymmetric systems as presented in the two first test problems. Moreover, the method also can be used as a stand-alone solver.

Overall, in spite of the relatively large setup times compared to standard preconditioners IC and ILU, the LAMG preconditioner accelerates the iteration process and it gave a lower number of iterations and the smaller total time, even for the unsymmetric systems from real 3D computational fluid dynamics problems (Table III) and for the complex valued symmetric system arising from the steady-state analysis of Compumag TEAM 28 Problem (Table V), which, in general, can be very difficult to solve by iterative methods [30]. In fact, the approach based on a combination of the discrete wavelet transform and the algebraic multigrid method, which has been improved in this paper, has already proved to be efficient for other complex and critical 3D problems as, for example, the problem of time-harmonic electromagnetic behavior of a substation grounding system formulated in terms of ungauged *AV* edge finite-element analysis [13].

The significantly better performance of the LAMG for the complex problem can be partially credited to the difficult of the preconditioning techniques based on incomplete factorizations, as ILU and IC, to solve complex valued linear systems due to the appearance of unstable pivots during the incomplete factorization process [34, 35]. This LAMG performance for this kind of problem is especially interesting because, curiously, there is not much literature available on iterative solvers and preconditioners for complex symmetric problems, given the number of diverse applications in which these problems arise. However, in order to explain correctly the differences of the convergence between the different problems a theoretical spectral analysis should to be accomplished, but such an analysis is out of the scope of this paper.

It is important to highlight that as the LAMG algorithm implemented in this work uses the Daubechies 2 filters its efficiency is similar to the standard wavelet-based algebraic multigrid (Table I). However, the other advantages presented in section III are kept, especially, the inherently parallel feature and the capacity to transform signals with an arbitrary length (need not be 2^n), which has allowed to develop very appropriate algorithms in the Finite Element context.

As for the parallel method, the proposed algorithm has been applied as a black-box solver in some numerical Poisson problems with good results. The obtained results for different problem sizes are compared with those achieved by some of the most advanced and current single processor AMG algorithms, demonstrating the efficiency of the new approach.

Like the standard AMG, this method has the very important scalability property that can be observed in table VIII, which shows the number of iterations required for PWAMG convergence for different numbers of matrix equations.

Other important characteristic of the PWAMG method is its small demand for interprocessors communication. Actually, no communication is required in setup phase when first order filters are used. This characteristic is confirmed by the results for setup time presented in table VIII, only observing that the times measured by `MPI_Wtime()` and `clock()` functions are practically the same ones.

Additionally, the results have shown that the parallel method was more efficient for large problems, as is characteristic of the multilevel approaches. The parallel efficiency for the bigger Poisson problems tested was about 60%. When 2 processors were used the method reached a little more than the linear speedup and for 4 processors the speedup was less than the linear. In spite of this fact, it is worth pointing out some important aspects of this application: in the context of large sparse linear system of equations where this paper is inserted, the problems have large memory requirements. In these cases, as presented in [33], the speedup necessary to be cost-effective can be much less than linear. The parallel program does not need *p* times memory of the unit processor, since parallelizing a job rarely multiplies its memory requirements by *p*.

XI. References

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

[2] Chang, Q., Wong, Y.S., and Fu, H., “On the algebraic multigrid method”, *J. Comput. Phys.* 125, 1996, 279–292.

[3] Haase, G., Kuhn, M., and Reitzinger, S., “Parallel Algebraic Multigrid Methods on Distributed Memory Computers”, *SIAM J. Sci. Comput.*, 24, 2002, 410-427.

[4] Henson, V.E. and Yang, U.M., “BoomerAMG: a Parallel Algebraic Multigrid Solver and Preconditioner”, *Appl. Num. Math.*, 41(1), 2002, 155−177.

[5] Mifune, T., Iwashita, T. and Shimasaki, M., “New algebraic multigrid preconditioning for iterative solvers in electromagnetic finite edge-element analyses,” *IEEE Trans. Magn.*, 39(3), 2003, 677–1680.

[6] Briggs, W.L. and Henson, V.E., “Wavelets and multigrid”, *SIAM J. Sci. Comput.*, 14, 1993, 506–510.

[7] Avudainayagam, A. and Vani, C., “Wavelet based multigrid methods for linear and nonlinear elliptic partial differential equations”, *Appl. Math. Comput.* 148, 2004, 307–320.

[8] De Leon, D., *Wavelet techniques applied to multigrid methods*, Department of Mathematics UCLA, UCLA Mathematics Department CAM Report 00-42, December 2000.

[9] Wang, G., Dutton, R.W. and Hou, J. “A fast wavelet multigrid algorithm for solution of electromagnetic integral equations”, *Microwave and Optical Technology Letters*, 24(2), 2000, 86–91.

[10] Chen, R.S., Fang, D.G., Tsang, K.F., and Yung, E.K.N., “Analysis of millimeter wave scattering by an electrically large metallic grating using wavelet-based algebraic multigrid preconditioned CG method”, *Int. J. Infrared Millimeter Waves*, 21(9), 2000, 1541–1560.

[11] Pereira, F.H., Verardi, S.L.L., Nabeta, S.I., “A Wavelet-based Algebraic Multigrid preconditioner for sparse linear systems”, *Appl. Math. Comput.*, 182, 2006, 1098-1107.

[12] Garcıa, V.M., Acevedo, L., and Vidal, A.M., “Variants of algebraic wavelet-based multigrid methods: Application to shifted linear systems,” *Appl. Math. Comput.*, 202(1), 2008, 287–299.

[13] Pereira, F.H. *et al.*, “A Wavelet-based Algebraic Multigrid Preconditioning for Iterative Solvers in Finite Element Analysis”. *IEEE Trans. on Magn.*, 43(4), 2007, 1553-1556.

[14] Pereira, F.H., Palin, M.F., Verardi, S.L.L., Nabeta. S.I., “A Parallel Wavelet-based Algebraic Multigrid black-box Solver and Preconditioner”, *16th Compumag - Conference on the Computation of Electromagnetic Fields*, 2007, Aachen.

[15] Pereira, F.H., Nabeta. S.I.,, “Wavelet-Based Algebraic Multigrid Method Using the Lifting Technique”, *J. of Microwaves, Optoelectronics and Electromagnetic Applications*, 9(1), 2010, 1-9.

[16] Pereira, F.H., Afonso, M.M., Vasconcelos, J.A., and Nabeta, S.I., “An Efficient Two-Level Preconditioner based on Lifting for FEM-BEM Equations”, *J. of Microwaves, Optoelectronics and Electromagnetic Applications* , 9(2), 2010, 78-88.

[17] Jensen, A. and la Cour-Harbo, A., *The Discrete Wavelet Transform*, Ripples in Mathematics, Springer, Berlin, 2001.

[18] Sweldens, W., “The lifting scheme: A custom-design construction of biorthogonal wavelets”, *Appl. Comput. Harmon. Analysis*, 3(2), 1996, 186-200.

[19] Pereira, F.H., Verardi, S.L.L., Nabeta, S.I., “A fast algebraic multigrid preconditioned conjugate gradient solver”, *Applied Math. and Comp.*, 179, 2006, 344-351.

[20] Griebel, M., Metsch, B., Oeltz, D., Alex, M., Schweitzer, E., “Coarse grid classification: a parallel coarsening

scheme for algebraic multigrid methods", *Numerical Linear Algebra with Applications*, Vol. 13(2–3), 2006, 193-214.

[21] Cleary, A.J., Falgout, R.D. Henson, V.E. Jones, J.E., *Coarse-grid Selection for Parallel Algebraic Multigrid*, Lawrence Livermore National Laboratory, Livermore, 2000.

[22] Sterck, H.D., Yang, U.M., and Heys, J.J. 2005. "Reducing Complexity in Parallel Algebraic Multigrid Preconditioners", *SIAM J. Matrix Anal. Appl*. 27(4), 2005, 1019-1039.

[23] Sarkar, T.K., Magdalena, S.P. and Michael, C.W., *Wavelet Applications in Engineering Electromagnetics*, Artech House, Boston, 2002.

[24] Daubechies, I. and Sweldens, W., "Factoring Wavelet Transforms into Lifting Steps", *J. Fourier Anal. Appl.*, 4(3), 1998, 247-269.

[25] Davis, T., *University of Florida Sparse Matrix Collection*, NA Digest, 1997.

[26] Kirk, B. *et al.*, LibMesh: "A C++ Library for Parallel Adaptive Mesh Refinement/Coarsening Simulations", *Engineering with Computers*, 22(3-4), 2006, 237-254.

[27] Karl, H., Fetzer, J., Kurz, S., Lehner, G., Rucker, W.M., "Description of TEAM workshop problem 28: An electromagnetic levitation device", *Proceedings of the TEAM Workshop in the Sixth Round*, Rio de Janeiro, 1997, 48-51.

[28] Meeker, D., *Finite element method magnetic: User's manual*, Massachusetts, USA, FEMM Version 4.2, 2003.

[29] Chen, Q. and Konrad, A., "A review of finite element open boundary techniques for static and quasistatic electromagnetic field problems", *IEEE Trans. on Magn.*, 33(1), 1997, 663-676.

[30] Ford, J.M., Chen, K., and Ford, N.J. "Parallel implementation of fast wavelet transforms", *Numerical Analysis*, 39, Manchester University, 2001.

[31] Sips, H.J., and Lin, H.X., *High Performance Computing Course MPI Tutorial*, Delft University of Technology Information Technology and Systems, September 16, 2002.

[32] Chang, Q., and Huang, Z., "Efficient Algebraic Multigrid Algorithms and their Convergence", *SIAM J. Sci. Comput.*, vol. 24(2), 2002, 597-618.

[33] Wood, D.A., and Hill, M.D., "Cost-effective parallel computing", *Computer*, vol. 28 (2), 1995, 69-72.

[34] Osei-Kuffuor, D. and Saad, Y., "Preconditioning Helmholtz linear systems", *Appl. Num. Math.*, 60(4), 2010, 420-431.

[35] Benzi, M., "Preconditioning Techniques for Large Linear Systems: A Survey", *J. of Comput. Physics*, 182, 2002, 418–477.

AUTHORS NAME AND AFFILIATION

Fabio Henrique Pereira, Industrial Engineering Post Graduation Program, Nove de Julho University (PMEP/Uninove), Francisco Matarazzo, Av. 612, 05001100,

Água Branca, São Paulo, Brazil, Tel.: +55 11 36639355, e-mail: fabiohp@uninove.br

Silvio Ikuyo Nabeta, Electrical Machine and Drives Lab of São Paulo University (GMAcq/USP), Luciano Gualberto Av., 158, Cidade Universitária, São Paulo, Brazil, Tel.: +55 11 30919809, e-mail: nabeta@pea.usp.br