Non-nested and non-structured multigrid methods applied to elastic problems. Part I: The two-dimensional case

MARCO L. BITTENCOURT (*)
CRAIG C. DOUGLAS (**)
RAÚL A. FEIJÓO (***)

* Center for Computational Sciences
University of Kentucky
325 McVey Hall, Lexington, KY, 40506-0045, USA
e-mail: mlb@ccs.uky.edu

** Department of Mathematics
University of Kentucky
715 Patterson Office Tower, Lexington, KY, 40506-0027, USA
e-mail: douglas@ccs.uky.edu

*** Laboratório Nacional de Computação Científica (LNCC/CNPq)
Av. Getulio Vargas 333, CEP 25651-070, Petrópolis/RJ, Brazil
e-mail: feij@alpha.lncc.br

SUMMARY

This paper presents the application of non-nested and non-structured multigrid methods for two-dimensional elastic linear problems. Some basic aspects related to multigrid methods are discussed including nested iterations, coarse grid correction scheme, transfer operators, and multigrid strategies. A variational formulation for the transfer operators is also considered. The C++ implementation of the multigrid software is discussed and some examples are analyzed. The performance of multigrid strategies is compared with direct and pre-conditioned conjugate gradient algorithms.

1 Introduction

The equilibrium equation for a continuous media with domain Ω [15] is given by

$$\text{div}\mathbf{T} + \mathbf{f} = \mathbf{0}, \tag{1}$$

where $\mathbf{T}$ is the Cauchy stress tensor and $\mathbf{f}$ is the body force vector field. For a linear elastic problem the stress tensor is related to the infinitesimal strain tensor $\mathbf{E}(\mathbf{w}) = \frac{1}{2}\left(\nabla\mathbf{w} + \nabla\mathbf{w}^T\right)$ linearly as $\mathbf{T} = \mathsf{C}\left[\mathbf{E}\right] = 2\mu\mathbf{E} + \lambda(\text{tr }\mathbf{E})\mathbf{I}$. C is the elasticity tensor, $\mathbf{w}$ is the displacement vector field, and λ and μ are the Lamé's coefficients.

The essential and natural boundary conditions on $\partial\Omega$ are defined by $\mathbf{w} = \mathbf{0}$ for $\mathbf{x} \in \Gamma^1$ and $\mathbf{Tn} = \mathbf{\Phi}$ for $\mathbf{x} \in \Gamma^2$. Here, $\partial\Omega = \Gamma^0 \cup \Gamma^1 \cup \Gamma^2$ and $\Gamma^0 \cap \Gamma^1 \cap \Gamma^2 = \emptyset$. Γ^0 is a part of the boundary $\partial\Omega$ without any prescribed boundary condition and $\mathbf{\Phi}$ is the surface force vector field.

The weak or variational formulation for the elastic problem is given by

$$\int_\Omega \mathbf{T}\left(\mathbf{w}\right)\cdot\mathbf{E}\left(\mathbf{v}\right)\ dV = \int_{\Gamma^2} \mathbf{\Phi}\cdot\mathbf{v}\ dA + \int_\Omega \mathbf{f}\cdot\mathbf{v}\ dV, \qquad \forall\mathbf{v} \in \mathcal{V}, \tag{2}$$

where $\mathcal{V}$ is the vector space of admissible displacements. Specifically $\mathcal{V} = \{v \mid v \in H^1(\Omega), v = 0 \; on \; \Gamma^1\}$ and $H^1(\Omega)$ is a Hilbert space. The previous equation can be written as,

$$a(u, v) = l(v), \qquad \forall v \in \mathcal{V}, \tag{3}$$

where $a(\cdot, \cdot) : \mathcal{V} \times \mathcal{V} \to \Re$ is a bounded, coercive, symmetric bilinear form and $l(\cdot) : \mathcal{V} \to \Re$ is a bounded linear form. The Lax-Milgram lemma guarantees the existence and uniqueness of the solution of the weak form (3).

An approximate solution z for (3) can be obtained through a Ritz-Galerkin formulation [1]. For this purpose an N-finite dimensional subspace $\mathcal{V}_j \subset \mathcal{V}$ with basis functions $\{\phi\}_{i=1}^N$ is defined. The approximation z is given by the linear combination $z = \sum_{i=1}^N u_i \phi_i$. Substituting this linear combination into (3), the coefficients u_i are obtained from the solution of a system of linear equations

$$\mathbf{A}\mathbf{u} = \mathbf{b}, \tag{4}$$

where $\mathbf{A}$ is a symmetric matrix of order N and $\mathbf{u}$ and $\mathbf{b}$ are the vectors of unknown and independent terms. For the solution of (4), direct, iterative, and multigrid algorithms are usually applied [1, 2, 9].

Multigrid methods use several meshes for solving (4). Nested iterations, coarse grid correction, transfer operators, and relaxation schemes are applied [2]. Traditionally, multigrid methods have been used with nested meshes (see Figure 1). This simplifies transfer operators, mesh generation, definition of adaptive refinement criteria [9, 18, 23], and convergence for some classes of problems [10, 17, 20]. The main feature of interest is that some multigrid methods have an asymptotic linear cost $\mathcal{O}(N)$ for the solution of (4).

However, many engineering problems have complex geometries. This makes it difficult to generate a sequence of nested meshes. Therefore the use of non-nested approximation spaces is an interesting option [2, 4]. Non-nested multigrid was applied to fluid flow problems in [19, 21]. Assuming that the number of variables in the finest mesh is sufficiently large, the cost of mesh generation is negligible when compared to the solution of the problem. Appropriate data structures are used to transfer information among the meshes.

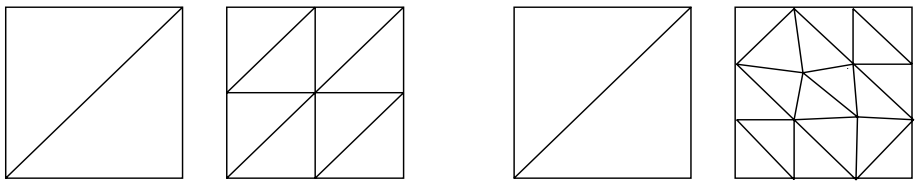

Figure 1: Nested and non-nested meshes.

Convergence rates for non-nested multigrid methods are proven in [8, 12, 24]. The operators and meshes used in this paper meet the requirements established in [12, 24]. Hence, the convergence of the multigrid strategies applied to the examples used in this paper is assured. The convergence theory in [8] depends on a subtle requirement on the smoothers that our quite standard relaxation method fails to meet. However, more complicated relaxation and Krylov subspace methods do meet the requirements in [8] and can be substituted.

In this paper concepts related to the application of non-nested multigrid methods to linear elastic problems using non-structured meshes are discussed. Concepts discussed include nested iterations, coarse grid correction, variational formulation for operators, and a C++ implementation [7]. Results for two-dimensional elastic examples are also presented.

In [5], some aspects related to operators implementation, expressions for calculating multigrid computational costs, and three-dimensional elastic examples are considered. For those examples a linear average cost for the solution of system of equations was obtained. Adaptive multigrid methods using the Zienkiewicz-Zhu error estimator [25] and adaptive refinement criteria are discussed in [3].

2 Multigrid Methods

The principal elements of multigrid methods (e.g., nested iterations, coarse-grid correction, and transfer operators) and some strategies will be discussed in this section.

2.1 One way or cascadic multigrid

The performance of an iterative procedure can be improved by using a better initial approximation obtained, for example, by relaxation on a coarser mesh. As the number of unknowns on the coarser mesh is smaller, the computational cost for one relaxation is also much smaller than one on the finer mesh. The following one-way or cascadic multigrid scheme obtains a better approximation for the finest mesh solution [10].

Relax on $\mathbf{Au} = \mathbf{b}$ on the coarsest mesh Ω^1.
$\vdots$
Relax on $\mathbf{Au} = \mathbf{b}$ on Ω^{k-2} to obtain an initial approximation for Ω^{k-1}.
Relax on $\mathbf{Au} = \mathbf{b}$ on Ω^{k-1} to obtain an initial approximation for Ω^k.
Relax on $\mathbf{Au} = \mathbf{b}$ on Ω^k to obtain a final approximation to the solution $\mathbf{u}$.

2.2 Coarse grid correction

Let $\mathbf{v}$ be an approximation for the solution $\mathbf{u}$ of (4). The residual equation $\mathbf{Ae} = \mathbf{r}$ defines a relation between the algebraic error $\mathbf{e} = \mathbf{u} - \mathbf{v}$ and the residual $\mathbf{r} = \mathbf{b} - \mathbf{Av}$ and allows iterations using the error $\mathbf{e}$. Relaxing on (4) with some approximation $\mathbf{v}$ is equivalent to iterating on the residual equation, with initial approximation $\mathbf{e} = \mathbf{0}$. The coarse grid correction scheme [10] is defined in the following way:

Relax on $\mathbf{Au} = \mathbf{b}$ on Ω^k to obtain an approximation $\mathbf{v}^k$.
Compute the residual $\mathbf{r} = \mathbf{b} - \mathbf{Av}^k$.
    Relax on $\mathbf{Ae} = \mathbf{r}$ on Ω^{k-1} to obtain an approximation to the error $\mathbf{e}^{k-1}$.
Correct the approximation obtained on Ω^k with the error estimate on Ω^{k-1} : $\mathbf{v}^k \leftarrow \mathbf{v}^k + \mathbf{e}^{k-1}$.

It is well known that the stationary iterative methods have the smoothing property, for which the higher frequency components in the algebraic error $\mathbf{e}$ of an approximation $\mathbf{v}$ are eliminated with only a few relaxations. After only a few relaxations on the fine mesh Ω^k the convergence rate deteriorates. The coarse grid correction scheme should then be used to get an approximation to the error $\mathbf{e}$, which corrects the fine grid approximation calculated previously.

When a smooth function on a finer mesh Ω^k is mapped onto a coarser mesh Ω^{k-1} it becomes more oscillatory. An oscillatory mode in Ω^k is transformed onto a smooth function on Ω^{k-1}. Therefore, when there is a stagnation in the rate of convergence of an iterative method, indicating that the oscillatory components have been eliminated, the approximation should be mapped onto a coarser mesh. As the mapped function is oscillatory, the higher frequency components can be smoothed by an iterative procedure (e.g., Gauss-Seidel).

Some common multigrid schemes are depicted pictorially in Figure 3.

2.3 Full multigrid or nested iteration

Combining the one-way and correction multigrid methods leads to what is known as full multigrid [9] or nested iteration [16]. Its prime advantage is that it is of optimal order in both time and space [12]. The algorithm is defined recursively as follows:

> If Ω^k is not the coarsest mesh, then $\mathbf{v}^k \leftarrow \mathbf{v}^{k-1}$.
> Correct the approximation on Ω^k using ν_k iterations of the correction multigrid scheme from §2.2.

Some common full multigrid schemes are depicted pictorially in Figure 3.

2.4 Transfer operators

The concepts discussed in the previous sections are based on transferring operators between coarse and fine meshes. The first operator, denoted by I_{k-1}^k, transforms functions from the coarse mesh Ω^{k-1} to the fine mesh Ω^k and is called an interpolation or prolongation operator. It is used to map the error $\mathbf{e}^{k-1}$ or the initial approximation $\mathbf{v}^{k-1}$ from the coarse mesh Ω^{k-1} onto the fine mesh Ω^k. The restriction operator I_k^{k-1} maps functions from the fine mesh Ω^k to the coarse mesh Ω^{k-1}, as in the case of the residual projection $\mathbf{r}^k$ onto Ω^{k-1}. For nested meshes, the simplest types of restriction are injection and weighting operators like the L_2 projection [10].

In this paper non-nested meshes are considered. Using a variational formulation this section presents how the restriction I_k^{k-1} and prolongation I_{k-1}^k operators are obtained when transferring information between two generic meshes $k-1$ and k. The approach used here provides the additional benefit of being valid for both nested and non-nested meshes. It also allows transfer operators to be calculated for problems with generalized degrees of freedom such as in plate and shell bending, as well as in mixed models. Its extension to non-linear problems is straightforward.

In order to simplify the presentation, the following variational problem is solved using a multigrid method: *find* $u \in H_0^1(\Omega)$ *such that*

$$a(u, v) = b(v), \qquad \forall v \in H_0^1(\Omega), \tag{5}$$

where $H_0^1(\Omega)$ is the Hilbert space whose elements are functions with zero values on the boundary and square-integrable with first derivatives also square-integrable. The approximation by finite elements and multigrid techniques requires solving the following set of problems: *find* $u_k \in V_k$ *such that*

$$a(u_k, v) = b(v), \qquad \forall v \in V_k. \tag{6}$$

For linear finite elements, $V_k(\Omega) = \{v \in C_0(\Omega) \mid v|_K \quad \text{is linear} \quad \forall K \in \mathcal{T}_k\} \subset H_0^1(\Omega)$. $C_0(\Omega)$ is the space of continuous functions in Ω which are zero along the boundary. Finally,

$\{\mathcal{T}_k, k = 1, 2, \ldots\}$ is a family of non-nested triangular meshes in the Ω domain, implying that $\mathcal{T}_k \not\subset \mathcal{T}_{k+1}$ and $V_k(\Omega) \not\subset V_{k+1}(\Omega)$.

Let the usual projection operator I_k in V_k be given by

$$I_k : \left\{ \begin{array}{lcl} C_0(\Omega) & \rightarrow & V_k(\Omega), \\ u(x) & \rightarrow & I_k u(x) = \sum_{n_i \in \mathcal{N}_k} u(n_i)\phi_{k,i}(x) = \Phi_k \cdot \mathbf{u}_k. \end{array} \right. \tag{7}$$

$\mathcal{N}_k$ is the set of nodal points corresponding to partition $\mathcal{T}_k$. $\phi_{k,i}(x)$ is the interpolation function associated with node n_i corresponding to the finite element type defining the triangulation $\mathcal{T}_k$. Φ and $\mathbf{u}$ are the vectors of the interpolation function and nodal values.

Since $\tilde{u}_k \in V_k$ is an approximation for the problem (6), the associated residual is

$$\tilde{r}_k(v) = b(v) - a(\tilde{u}_k, v) = \sum_{n_i \in \mathcal{N}_k} [b(\phi_{k,i}) - a(\tilde{u}_k, \phi_{k,i})]v(n_i) = \mathbf{r}_k \cdot \mathbf{v}_k, \quad \forall v \in V_k. \tag{8}$$

The residual is defined for all $v \in V_k$. Thus, for $v = I_k v_{k-1} \in V_k$ we can define the functional $r_k^{k-1} \in \mathcal{L}(V_{k-1}, \Re)$ by

$$\begin{aligned} r_k^{k-1}(v_{k-1}) &\equiv \tilde{r}_k(I_k v_{k-1}) = b(I_k v_{k-1}) - a(\tilde{u}_k, I_k v_{k-1}) \\ &= \sum_{n_i \in \mathcal{N}_k} [b(\phi_{k,i}) - a(\tilde{u}_k, \phi_{k,i})]v_{k-1}(n_i) \\ &= \sum_{n_i \in \mathcal{N}_k} \left([b(\phi_{k,i}) - a(\tilde{u}_k, \phi_{k,i})] \sum_{n_j \in \mathcal{N}_{k-1}} v_{k-1}(n_j)\phi_{k-1,j}(n_i) \right) \\ &= \sum_{n_j \in \mathcal{N}_{k-1}} \left(\sum_{n_i \in \mathcal{N}_k} [b(\phi_{k,i}) - a(\tilde{u}_k, \phi_{k,i})]\phi_{k-1,j}(n_i) \right) v_{k-1}(n_j) \\ &= \mathbf{r}_{k-1} \cdot \mathbf{v}_{k-1}. \end{aligned} \tag{9}$$

Transferring the residual $\mathbf{r}_k$ (which is related to $\tilde{u}_k$) from mesh $\mathcal{T}_k$ to mesh $\mathcal{T}_{k-1}$ requires determining $\mathbf{r}_{k-1}$. Applying (9) gives us

$$\mathbf{r}_{k-1} = I_k^{k-1}\mathbf{r}_k = (\mathbf{N}_{k-1}^k)^T \mathbf{r}_k, \tag{10}$$

where

$$\mathbf{N}_{k-1}^k = \left[\begin{array}{cccc} \phi_{k-1,1}(n_1) & \phi_{k-1,2}(n_1) & \ldots & \phi_{k-1,N_{k-1}}(n_1) \\ \phi_{k-1,1}(n_2) & \phi_{k-1,2}(n_2) & \ldots & \phi_{k-1,N_{k-1}}(n_2) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_{k-1,1}(n_{N_k}) & \phi_{k-1,2}(n_{N_k}) & \ldots & \phi_{k-1,N_{k-1}}(n_{N_k}) \end{array} \right]_{N_k \times N_{k-1}}. \tag{11}$$

The operator I_k^{k-1} is an $N_{k-1} \times N_k$ sparse matrix. The generic element $(I_k^{k-1})_{pq} = \phi_{k-1,p}(n_q)$, $p \in \{1, \ldots, N_{k-1}\}$, $q \in \{1, \ldots, N_k\}$, corresponds to the value of the interpolation function associated with node n_p of triangulation $\mathcal{T}_{k-1}$ calculated on the coordinates of node n_q from triangulation $\mathcal{T}_k$. The operator I_k^{k-1} is consistent in the sense that the residual is maintained while transferring along two consecutive meshes.

From a computational viewpoint, the restriction operator is calculated by taking the local contribution of the residual of node $n_i \in \mathcal{T}_k$ on the nodes of triangle $T_g \in \mathcal{T}_{k-1}$ where n_i is located. Figure 2 shows the operator for linear triangles. Residue $\mathbf{r}^I$, corresponding to the fine node I, is transferred as $\mathbf{r}^1$, $\mathbf{r}^2$, and $\mathbf{r}^3$ to the nodes of the coarser element using area coordinates A_1, A_2, and A_3. Taking the notation indicated in Figure 2, the residual for each node of the triangle T_g due to the residual at node I contained by T_g is given by

$$\mathbf{r}^i = A_i \mathbf{r}^I, \qquad i = 1, 2, 3. \tag{12}$$

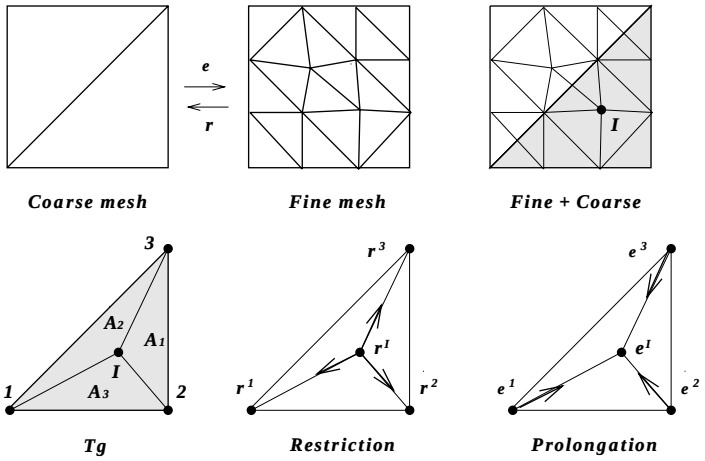

Figure 2: Restriction and prolongation operators.

After ν_1 iterations on level k the residual equation is solved on level $k-1$ using

$$a(e_{k-1}, v) = r_k^{k-1}(v), \quad \forall v \in V_{k-1}, \tag{13}$$

where $r_k^{k-1}(v) = b(I_k v) - a(\tilde{u}_k, I_k v), \ \ \forall v \in V_{k-1}$.

Another process in a multigrid method consists of prolonging the solution between levels $k-1$ and k in a consistent way with the restriction operator used. It can be observed that $\mathbf{e}_k = \mathbf{N}_{k-1}^k \mathbf{e}_{k-1}$ and $I_{k-1}^k = \mathbf{N}_{k-1}^k$, showing that $I_{k-1}^k = (I_k^{k-1})^T$. As illustrated in Figure 2, once corrections $\mathbf{e}^1$, $\mathbf{e}^2$, and $\mathbf{e}^3$ are known, the respective value for the finer node I is calculated as $\mathbf{e}^I = A_1\mathbf{e}^1 + A_2\mathbf{e}^2 + A_3\mathbf{e}^3$.

The same procedures can be extended to three-dimensional examples discretized by linear tetrahedral using volume coordinates. Efficient search procedures based on quadtree (octree) data structures [5, 11] have been applied to find the coarse triangle (tetrahedron) in which a finer node I is contained.

2.5 Multigrid strategies

Figure 3 shows some commonly used multigrid strategies. The V and W cycles are based on the recursive application of the coarse grid correction scheme. When going from a finer mesh Ω^k to a coarser mesh Ω^{k-1}, ν_1 pre-relaxations are executed on the original equation $\mathbf{Au} = \mathbf{b}$ or on the error equation $\mathbf{Ae} = \mathbf{r}$. The residual $\mathbf{r}$ is then calculated and projected onto mesh Ω^{k-1} using the restriction operator I_k^{k-1}. On the coarsest mesh, the error equation is solved by a direct or an iterative numerical method. Finally, corrections $\mathbf{e}$ are mapped to the finer meshes using the prolongation operator I_{k-1}^k followed by ν_2 post-relaxations.

The FMV algorithm uses the concept of nested iterations. Each V cycle is preceded by one or more V cycles on coarser grids based on the value of the parameter ν_0. $\mathbf{Au} = \mathbf{b}$ is solved on the coarsest mesh and its solution is prolonged as an initial approximation to the the solution on the next level. Then ν_0 V cycles are executed, ultimately obtaining a better approximation to the next mesh. This procedure is repeated until the finest mesh is reached. The FMW technique uses W cycles on coarser grids to obtain a better initial approximation to a next level. It is analogous to the FMV cycle. These procedures are illustrated in Figure 3 for $\nu_0 = 1$. Some variations of these algorithms can be used. For example, the FMVV scheme uses an FMV cycle followed by many V cycles.

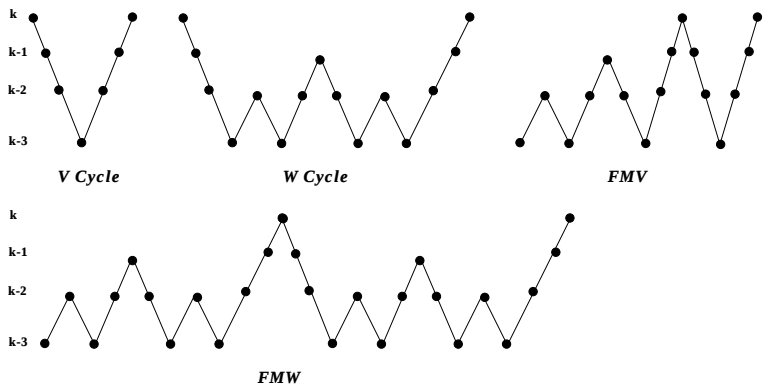

Figure 3: Multigrid strategies.

3 Computational Implementation

All procedures were implemented using C++ in a set of libraries with classes for a database, a matrix, and the finite elements [14]. Some classes for sparse matrices using a row-compressed format with direct and iterative solution methods were incorporated in these libraries [2]. For sparse Gaussian elimination, a symbolic procedure was used to identify the fill-in (i.e., to determine the elements that become non-zero along the factorization process) [22]. A minimum degree algorithm for renumbering of equations is included.

The multigrid strategies were implemented in only one class with parameters for the database information, number of meshes, numbers of pre- and post-relaxations, and pointer vectors to the unknown and the independent terms of each mesh. The implementation of the multigrid techniques used the following methods:

```
int FineToCoarsePart(int Level);
int CoarseToFinePart(int Level);
```

The method FineToCoarsePart is responsible for the descendent part of the multigrid strategies: first ν_1 relaxations are executed, then the residual is calculated and transferred to the next coarse mesh. This is repeated until the coarsest mesh is reached.

The method CoarseToFinePart first solves the coarsest mesh by a direct method. Then it repeatedly prolongs corrections and executes ν_2 post-relaxations until the finest level is reached.

The multigrid strategies presented previously are easily implemented. For example, the main kernel of a V cycle consists of the following C++ commands:

```
//fine to coarse part
for(k = NumMeshes - 1; k >= 0; k--) FineToCoarsePart(k);

//coarse to fine part
for(k = 1; k < NumMeshes; k++) CoarseToFinePart(k);
```

The two methods FineToCoarsePart and CoarseToFinePart extensively use other methods for restriction and interpolation operators. Finally, it should be noted that the mapping between the meshes uses quadtree and octree data structures for two- and three-dimensional problems respectively. It is implemented using the classes developed in [11]. A more detailed description of the multigrid classes can be found in [7].

4 Numerical experiments

With the purpose of evaluating the performance of multigrid schemes, some plane stress linear elastic examples are presented below. Initially, a plate in traction modeled with nested and non-nested meshes is considered. After that, the results for a plate with hole and a fracture problem are presented. All problems were discretized by linear triangle meshes generated by a C++ implementation of the advancing front technique (see [13] and the papers cited therein).

Multigrid results were compared with sparse Gaussian elimination (SGE) and an iterative method based on conjugated gradient with symmetric Gauss-Seidel (CGGS) pre-conditioner [1]. For all examples, the system matrix was stored in a row-compressed sparse format [2]. In the case of iterative and multigrid methods, the convergence criteria $\frac{||\mathbf{Au}-\mathbf{b}||_2}{||\mathbf{b}||_2} < \xi$ in the Euclidian norm with $\xi = 10^{-4}$ was used. The Gauss-Seidel iterative method was used as a relaxation scheme in the multigrid cycles. The following relative errors were used for comparisons of direct ($\mathbf{u}_{dir}$), iterative ($\mathbf{u}_{ite}$), and multigrid ($\mathbf{u}_{mg}$) methods:

$$\begin{cases} ||e_r^{dir/ite}||_2 = \frac{||\mathbf{u}_{dir}-\mathbf{u}_{ite}||_2}{||\mathbf{u}_{dir}||_2}, \\ ||e_r^{dir/mg}||_2 = \frac{||\mathbf{u}_{dir}-\mathbf{u}_{mg}||_2}{||\mathbf{u}_{dir}||_2}, \\ ||e_r^{ite/mg}||_2 = \frac{||\mathbf{u}_{ite}-\mathbf{u}_{mg}||_2}{||\mathbf{u}_{ite}||_2}. \end{cases} \tag{14}$$

For each method (SGE, CGGS, and MG) the equivalent number of finest mesh iterations was determined by taking the corresponding overall computational cost divided by the cost of one Gauss-Seidel iteration on the finest mesh. A flop is a single floating point operation.

Finally, the results for the multigrid methods, in terms of the number of operations, are superior by about 15% over those in [2, 6]. Instead of calculating the residual explicitly by $\mathbf{r} = \mathbf{b} - \mathbf{Au}$, it can be calculated partially at the same time as the Gauss-Seidel relaxations.

All of the examples are two-dimensional in nature. They are solved on

$$\Omega = \{(x, y, z) \in \Re^3 \mid 0 \leq z \leq t\}$$

where t is the domain thickness. Additionally, the body force vector is zero, i.e., $\mathbf{f} = \mathbf{0}$ in (3). The definitions of Γ^1 and Γ^2 are different for the three examples.

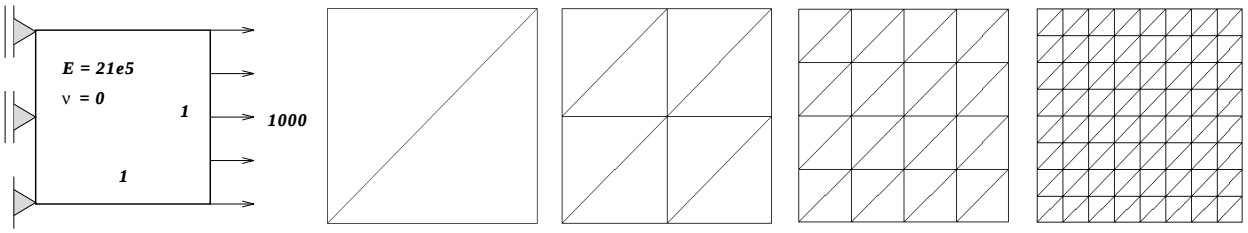

Figure 4: Plate under traction (nested meshes).

4.1 Plate under traction

Figure 4 shows a plate tractioned by a distributed load, rigidly supported on a point and with a displacement constraint in the x direction in the other nodes. Γ_1 and Γ_2 are defined in (3) by

$$\Gamma^1 = \{(x, y, z) \in \Omega \mid u_x = 0 \text{ for } x = y = 0 \text{ and } u_x = 0 \text{ for } x = 0\}$$

and

$$\Gamma^2 = \{(x, y, z) \in \Omega \mid \Phi_x = 1000 \text{ for } x = 1\}.$$

For multigrid analysis, four nested and non-nested meshes were generated and are illustrated in Figures 4 and 5, respectively. The main features of these nested and non-nested meshes are indicated in Tables 1 and 2 and include the following information:

- The numbers of nodes, elements, and equations used.
- The total number of coefficients for direct (NCoef^{dir}) and iterative/multigrid (NCoef^{ite}) methods used.
- The average number of coefficients per row of the sparse matrix, for direct (m^{dir}) and iterative/multigrid (m^{ite}) methods used.

Since the Poisson coefficient is zero, this example is equivalent to a unidimensional problem with the exact solution determined by the approximation space obtained by linear finite elements.

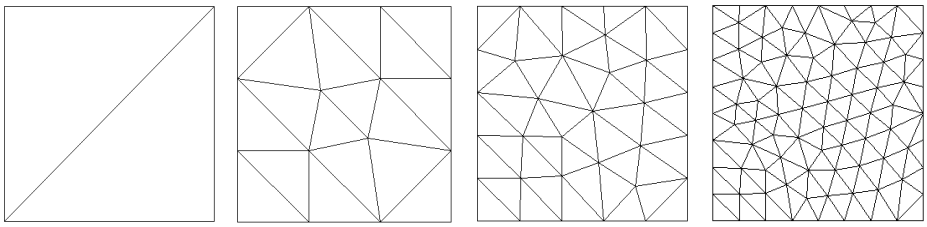

Figure 5: Plate under traction (non-nested meshes).

Table 1: Nested mesh features for the plate.

Mesh	Nodes	Elements	Equations	NCoef^{dir}	m^{dir}	NCoef^{ite}	m^{ite}
1	4	2	5	13	2.6	13	2.6
2	9	8	14	73	5.2	63	4.5
3	25	32	44	357	8.1	253	5.8
4	81	128	152	2183	14.4	993	6.5

Table 2: Non-nested mesh features for the plate.

Mesh	Nodes	Elements	Equations	NCoef^{dir}	m^{dir}	NCoef^{ite}	m^{ite}
1	4	2	5	13	2.6	13	2.6
2	16	18	27	173	6.4	145	5.4
3	35	48	63	548	8.7	380	6.0
4	89	144	168	2223	13.2	1113	6.6

Tables 3 and 4 provide summaries for the nested and non-nested meshes the equivalent number of finest mesh iterations (NIT) for SGE and CGGS methods and several multigrid strategies. The total number of cycles (NC) and the number of pre- (ν_1) and post-relaxations (ν_2) are also included for multigrid methods. The relative errors (14) are also presented.

The results in terms of the number of operations and memory space are illustrated in Figures 6 and 7 for the nested and non-nested meshes. The coefficients of the fitted straight lines are given between brackets. For multigrid methods, 2, 3 and 4 meshes were successively taken

and the points obtained are shown in Figures 6 and 7. The solution on the coarsest mesh was obtained by a direct method while in the other cases the same multigrid strategy indicated in Figures 6 and 7 were used.

Table 3: Results for the plate under traction (nested meshes).

Method	NIT	NC, ν_1, ν_2	$\|\|e_r^{dir/ite}\|\|_2$	$\|\|e_r^{dir/mg}\|\|_2$	$\|\|e_r^{ite/mg}\|\|_2$
SGE	12	–	–	–	–
CGGS	58	–	1.21×10^{-5}	–	–
V	59	15,1,1	–	4.24×10^{-4}	4.29×10^{-4}
W	37	7,1,1	–	1.01×10^{-4}	1.06×10^{-4}
FMV	4	1,1,0	–	4.33×10^{-15}	1.21×10^{-5}
FMW	5	1,1,0	–	3.57×10^{-15}	1.21×10^{-5}

Table 4: Results for the plate under traction (non-nested meshes).

Method	NIT	NC, ν_1, ν_2	$\|\|e_r^{dir/ite}\|\|_2$	$\|\|e_r^{dir/mg}\|\|_2$	$\|\|e_r^{ite/mg}\|\|_2$
SGE	11	–	–	–	–
CGGS	63	–	9.85×10^{-6}	–	–
V	48	11,1,1	–	6.46×10^{-4}	6.49×10^{-4}
W	33	5,1,1	–	6.80×10^{-5}	7.18×10^{-5}
FMV	5	1,1,0	–	6.59×10^{-8}	9.87×10^{-6}
FMW	6	1,1,0	–	3.10×10^{-8}	9.86×10^{-6}

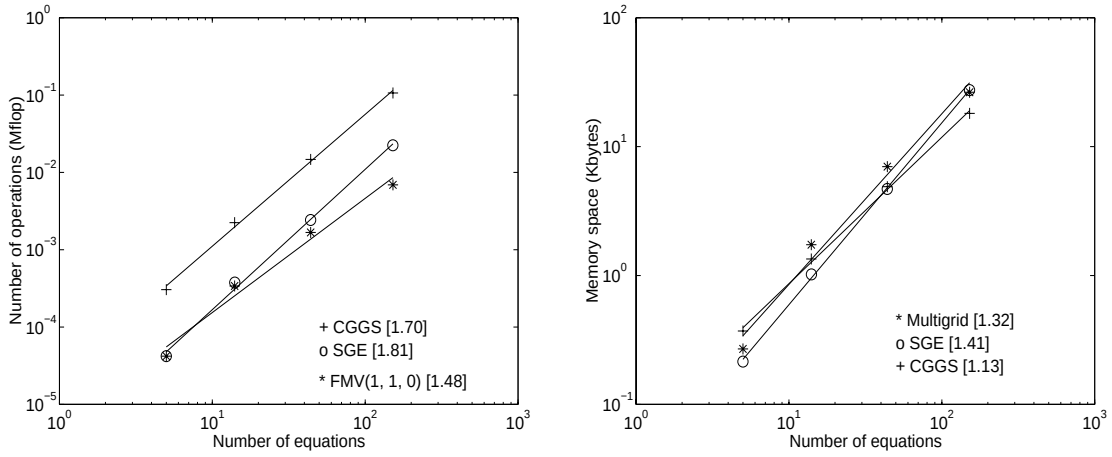

Figure 6: Plate under traction (nested meshes): number of operations and memory requirements.

4.2 Plate with hole and fracture problem

The four linear meshes used to analyze a plate with a hole and a fracture problem are illustrated in Figures 8 and 9. The main mesh features are given in Tables 5 and 6.

For the plate with a hole problem, Γ_1 and Γ_2 are defined in (3) by

$$\Gamma^1 = \{(x, y, z) \in \Omega \mid u_x = 0 \text{ for } x = 0 \text{ and } u_y = 0 \text{ for } y = 0\}$$

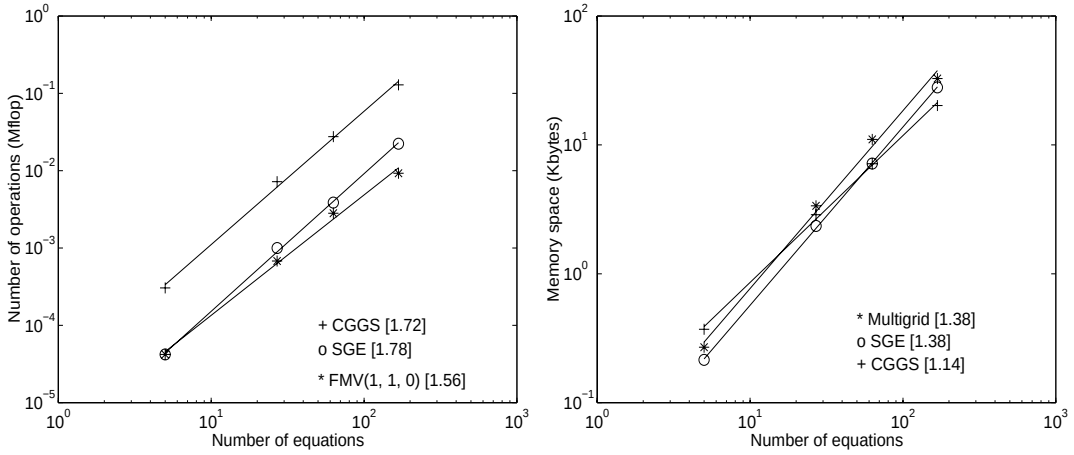

Figure 7: Plate under traction (non-nested meshes): number of operations and memory requirements.

and

$$\Gamma^2 = \{(x, y, z) \in \Omega \mid \Phi_x = 100 \text{ for } x = 20\}.$$

For the plate with a fracture problem, Γ_1 and Γ_2 are defined in (3) by

$$\Gamma^1 = \{(x, y, z) \in \Omega \mid u_x = 0 \text{ for } x = 0 \text{ and } u_y = 0 \text{ for } 2 \leq x \leq 10\}$$

and

$$\Gamma^2 = \{(x, y, z) \in \Omega \mid \Phi_y = 100 \text{ for } y = 20\}.$$

Tables 7 and 8 present the results for the plate with a hole and a fracture problems obtained by SGE, CGGS, and multigrid methods. Figures 10 and 11 show graphical results for the number of operations and the memory requirements.

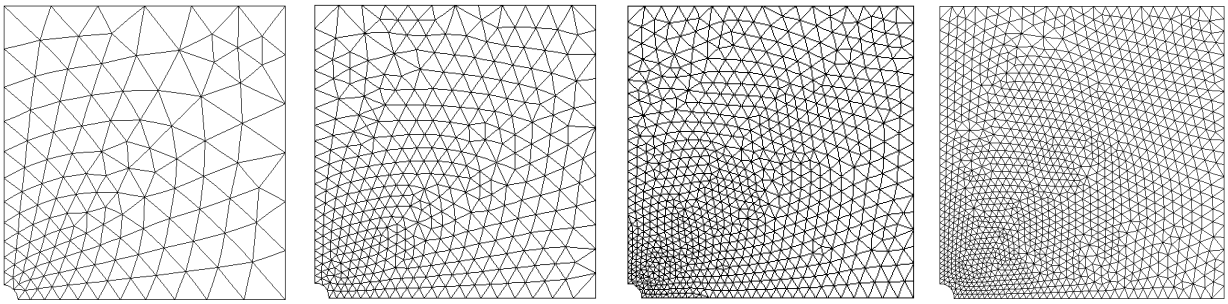

Figure 8: Plate with hole.

Table 5: Mesh features for the plate with hole.

Mesh	Nodes	Elements	Equations	NCoef^{dir}	m^{dir}	NCoef^{ite}	m^{ite}
1	38	57	64	593	9.3	381	6.0
2	121	206	220	3504	15.9	1471	6.7
3	445	817	846	21620	25.6	5985	7.1
4	1679	3215	3272	126794	38.8	23833	7.3

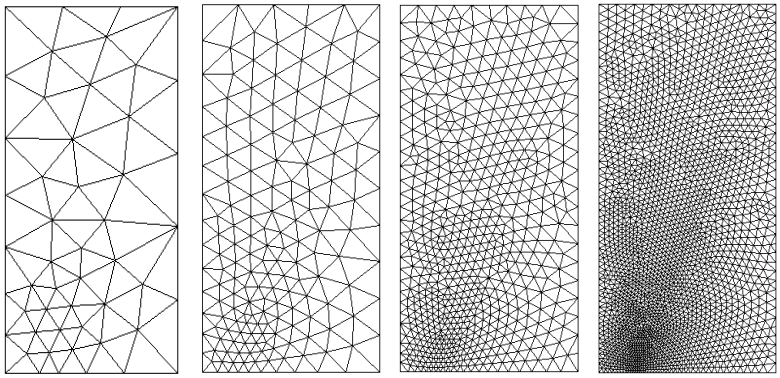

Figure 9: Fracture problem.

Table 6: Mesh features for the fracture problem.

Mesh	Nodes	Elements	Equations	NCoef^{dir}	m^{dir}	NCoef^{ite}	m^{ite}
1	51	80	89	911	10.2	555	6.2
2	170	299	317	5607	17.7	2171	6.8
3	626	1171	1207	32422	26.9	8647	7.2
4	2383	4608	4679	204249	43.7	34312	7.3

Table 7: Results for the plate with hole.

Method	NIT	NC, ν_0, ν_1, ν_2	$\|\|e_r^{dir/ite}\|\|_2$	$\|\|e_r^{dir/mg}\|\|_2$	$\|\|e_r^{ite/mg}\|\|_2$
SGE	100	–	–	–	–
CGGS	208	–	8.72×10^{-6}	–	–
FMV	28	4,1,1,1	–	4.71×10^{-6}	1.01×10^{-5}
FMVV	30	7,2,1,1	–	2.55×10^{-5}	2.74×10^{-5}
FMW	29	2,2,1,1	–	1.12×10^{-6}	8.80×10^{-6}

Table 8: Results for the fracture problem.

Method	NIT	NC, ν_0, ν_1, ν_2	$\|\|e_r^{dir/ite}\|\|_2$	$\|\|e_r^{dir/mg}\|\|_2$	$\|\|e_r^{ite/mg}\|\|_2$
SGE	132	–	–	–	–
CGGS	273	–	1.37×10^{-5}	–	–
FMV	33	4,1,2,1	–	8.76×10^{-6}	1.86×10^{-5}
FMVV	35	8,2,2,1	–	2.42×10^{-5}	3.17×10^{-5}
FMW	33	3,1,2,1	–	3.43×10^{-6}	1.52×10^{-5}

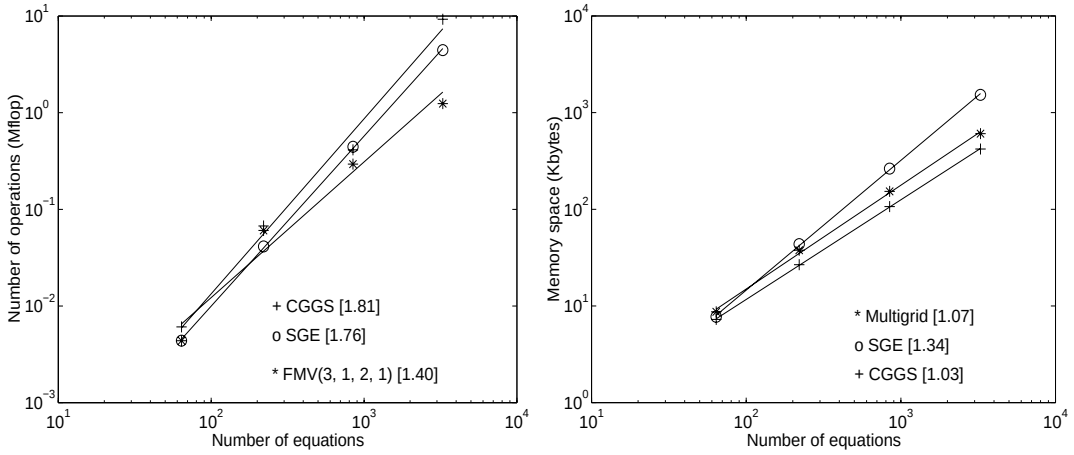

Figure 10: Plate with hole: number of operations and memory requirements.

5 Conclusions

For the elastic two-dimensional examples studied in Section 4, the number of operations for the solution of the system of equations by sparse Gaussian elimination (SGE) is smaller than the CGGS algorithm, which requires much less memory space.

For the simple example of a plate under traction in Section 4.1, it can be said that the FMV and FMW procedures, combining one-way multigrid with correction of the coarser mesh, are better than the V and W cycles. The numerical conditioning of the solutions obtained from the FMV and FMW procedures, as measured by the relative errors given by (14), is better than those obtained from the V and W cycles. Only one cycle of either the FMV or FMW procedures with $\nu_1 = 1$ and $\nu_2 = 0$ is sufficient to solve this problem.

The number of operations required by the FMV and FMW algorithms is smaller when compared to SGE and CGGS methods, as shown in Tables 3 and 4 by comparing the number of iterations on the finest mesh (or see Figures 6 and 7). The CGGS method requires less memory space, as expected, while the multigrid strategies and SGE have similar behavior. Therefore, even for a small-order example, the use of multigrid techniques becomes advantageous when compared to the SGE and CGGS procedures.

It was observed that the numerical conditioning of the solution obtained from the FMV and FMW procedures in non-nested meshes is inferior to the one obtained with nested meshes. However, the order of the relative error is very acceptable, indicating a good quality of this solution.

Taking the results in Table 7 for the plate with hole and fracture problems, it can be observed that FMV, FMW, and FMVV procedures have a similar behaviour. Their performance is superior to the SGE and CGGS methods, as can be seen by taking the equivalent number of finest mesh iterations or observing Figures 10 and 11. However, for the FMVV method, a larger number of cycles is necessary to obtain the same numerical conditioning. When considering the number of pre and post-relaxations, both $\nu_1 = 1$ and $\nu_1 = 2$ have shown, in the majority of cases, to reduce the total number of cycles [2]. This conclusion is also valid for $\nu_0 = 2$. In terms of memory requirements, the multigrid strategies are superior to SGE factorization, as can be seen in Figures 10 and 11.

For all of the examples, the angular coefficients of the lines representing the number of operations for the multigrid strategies are smaller than those for SGE and CGGS strategies. For memory space, the angular coefficients for multigrid and CGGS are similar and close to 1

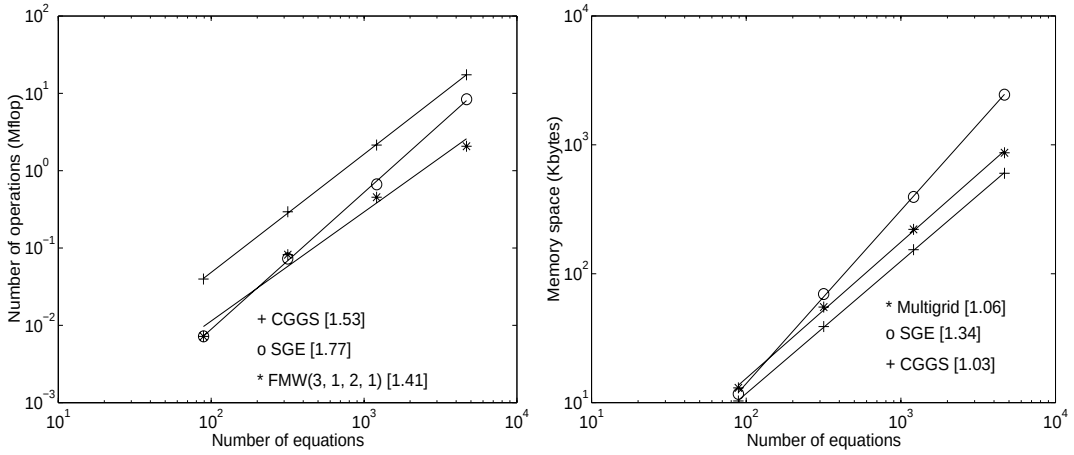

Figure 11: Fracture problem: number of operations and memory requirements.

and are superior to the SGE method. The average behavior of the number of operations for the problems studied here is shown in Figure 12.

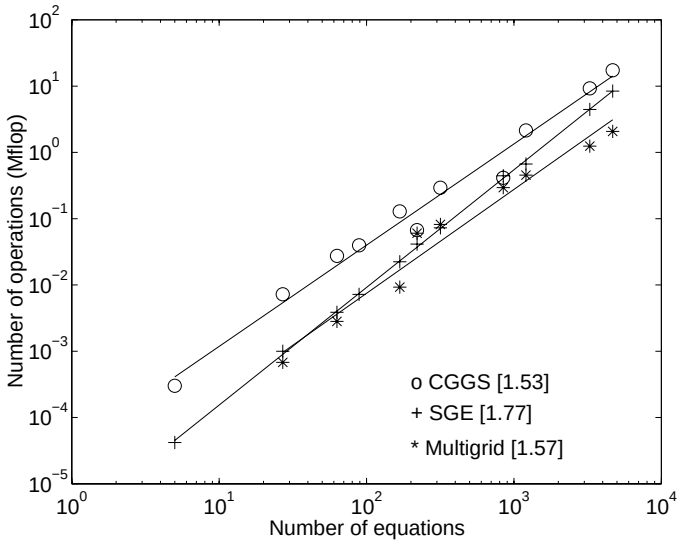

Figure 12: Average behaviour for the number of operations.

Acknowledgments

This work was supported in part by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Fundação de Amparo a Pesquisa do Estado de São Paulo (FAPESP), NSF grants 9707040 and 9721388, NATO grant CRG 971574, and the University of Kentucky Center for Computational Sciences. The authors are grateful for the software facilities provided by the TACSOM Group (http://www.lncc.br/~tacsom).

References

[1] O. Axelsson and V. A. Barker. *Finite Element Solution of Boundary Value Problems –*

Theory and Computation. Academic Press, Orlando, 1984.

[2] M. L. Bittencourt. *Adaptive Iterative and Multigrid Methods Applied to Non-Structured Meshes (in Portuguese)*. PhD thesis, State University of Campinas, Brazil, 1996.

[3] M. L. Bittencourt, C. C. Douglas, and R. A. Feijóo. Adaptive non-nested multigrid methods. Submitted for publication, 1998.

[4] M. L. Bittencourt, C. C. Douglas, and R. A. Feijóo. Linear non-nested multigrid methods. Submitted for publication, 1998.

[5] M. L. Bittencourt, C. C. Douglas, and R. A. Feijóo. Non-nested and non-structured multigrid methods applied to elastic problems. Part II: The three-dimensional case. Submitted for publication, 1998.

[6] M. L. Bittencourt and R. A. Feijóo. Non-nested multigrid methods in finite element linear structural analysis. In *Virtual Proceedings of the 8th Copper Mountain Conference on Multigrid Methods*, http://www.mgnet.org, April 1997. MGNet.

[7] M. L. Bittencourt and R. A. Feijóo. Object-oriented non-nested multigrid methods. In *IV World Conference on Computational Mechanics*, Buenos Aires, June 1998.

[8] J. H. Bramble, J. E. Pasciak, and J. Xu. The analysis of multigrid algorithms with nonnested quadratic forms. *Mathematics of Computation*, 56(193):1–34, 1991.

[9] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Mathematics of Computation*, 31(138):333–390, 1977.

[10] W. L. Briggs. *A Multigrid Tutorial*. SIAM Books, Philadelphia, 1987.

[11] E. Dari. *Contribuciones a la Triangulación Automática de Dominios Tridimensionales*. PhD thesis, Instituto Balseiro, Bariloche, Argentina, 1994.

[12] C. C. Douglas. Multi-grid algorithms with applications to elliptic boundary-value problems. *SIAM Journal on Numerical Analysis*, 21:236–254, 1984.

[13] E. A. Fancello, A. C. S. Guimarães, R. A. Feijóo, and M. Venere. Automatic two-dimensional mesh generation using object-oriented programming. In *Proceedings of 11th Brazilian Congress of Mechanical Engineering*, pages 635–638, São Paulo, December 1991. Brasilian Association of Mechanical Sciences.

[14] A. C. S. Guimarães and R. A. Feijóo. The ACDP System. Research report 27/89, National Laboratory for Scientific Computation, Rio de Janeiro, Brazil, 1989.

[15] M. E. Gurtin. *An Introduction to Continuum Mechanics*, volume 158 of *Mathematics in Science and Engineering*. Academic Press, 1981.

[16] W. Hackbush. *Multigrid Methods*. Springer-Verlag, Berlin, 1985.

[17] W. Hackbush and U. Trottenberg. *Multigrid Methods*. Springer-Verlag, Berlin, 1982.

[18] P. Leinen. Data structures and concepts for adaptive finite element methods. *Computing*, 55:325–354, 1995.

[19] D. J. Mavriplis. Multigrid solution of the two-dimensional Euler equations on unstructured triangular meshes. *AIAA Journal*, 26(7):325–354, 1987.

[20] S. F. McCormick. *Multigrid Methods*, volume 3 of *Frontiers Series*. SIAM Books, Philadelphia, 1987.

[21] J. Peraire, J. Peiro, and K. Morgan. Multigrid solution of the 3D compressible Euler equations on unstructured tetrahedral grids. *International Journal for Numerical Methods in Engineering*, 36:1029–1044, 1993.

[22] S. Pissanetzky. *Sparse Matrix Technology*. Academic Press, London, 1984.

[23] U. Rüde. Fully adaptive multigrid methods. *SIAM Journal on Numerical Analysis*, 30(1):230–248, 1993.

[24] S. Zhang. *Multi-Level Iterative Techniques*. PhD thesis, Department of Mathematics, Pennsylvania State University, 1988.

[25] O. C. Zienkiewicz and J. Z. Zhu. A simple error estimator and adaptative procedure for practical engineering analysis. *International Journal for Numerical Methods in Engineering*, 24:337–357, 1987.