Compatible coarse nodal and edge elements through energy functionals

F. Musy[⋆], L. Nicolas[†], R. Perrussel[⋆†] and M. Schatzman[⋆]

[†]*Centre de Génie Electrique de Lyon*
CEGELY, UMR CNRS 5005

[⋆]*Laboratoire de Mathématiques Appliquées de Lyon*
MAPLY, UMR CNRS 5585

Ecole Centrale de Lyon - 36, av. Guy de Collongue, 69134 Ecully cedex

Email : `ronan.perrussel@ec-lyon.fr`

September 2004

Abstract

The principles of multilevel methods for Maxwell's equations discretized by edge elements are reviewed. A particular emphasis is given to the algebraic method proposed by Reitzinger and Schöberl and the improvements introduced by Bochev et al.. A technique for the construction of coarse nodal elements by using a minimisation problem based on an energy norm is the starting point of our developments. The first development, in the nodal element case, is the replacement of computed coarse matrices by very similar structured matrices for the multilevel preconditioning. The second and main extension concerns a new method for constructing coupled coarse nodal and edge elements; a set of numerical experiments is presented on structured and unstructured meshes, in two and three dimensions. The geometry (square or cube) and the problem remain very simple.

Résumé

Le principe des méthodes multiniveau pour les équations de Maxwell discrétisées par les éléments finis d'arêtes est présenté. On s'attache plus particulièrement à décrire la méthode algébrique introduite par Reitzinger et Schöberl et les améliorations proposées sur la base de cette méthode par Bochev et collab.. Le point de départ de nos développements est une méthode permettant la construction de fonctions d'approximation grossière en éléments finis nodaux, grâce à l'utilisation d'un problème de minimisation en norme d'énergie. Le premier développement, qui s'applique aux éléments finis nodaux, consiste à remplacer des matrices grossières, calculées avec la méthode de minimisation en norme d'énergie, par des matrices structurées très "proches" des matrices initiales pour le préconditionnement multiniveau. Le second développement est une extension de la construction de fonctions d'approximation grossière par minimisation d'énergie pour construire des bases nodales et d'arêtes compatibles ; des résultats numériques sont réunis pour des maillages structurés et non structurés, en deux et trois dimensions. La géométrie (carré ou cube) et le problème considérés restent simples.

Contents

Chapter 1

Introduction

The present research is a part of a project called "*Efficient algorithms for solving linear systems coming from the edge element discretization of the time-harmonic Maxwell's equations*", and it is a cooperation between the *Laboratoire de Mathématiques Appliquées de Lyon* (MAPLY, Laboratory of Applied Mathematics of Lyon) and the *Centre de Génie Electrique de Lyon* (CEGELY, Center of Electrical Engineering of Lyon).

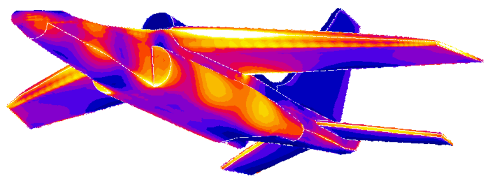


Figure 1.1: Illumination of a plane by a 100Mhz plane wave. Magnitude of the current density.

The design of *efficient communication systems* and the consideration of *electromagnetic compatibility problem*, for electrical or biological systems, as illustrated in Fig. 1.1 and 1.2, require the computation of electromagnetic wave propagation since the beginning of the conception process, in order to reduce the time and cost of the design. This computation must use efficient 3D numerical codes, based on the discretization of Maxwell's equations by finite element or boundary element methods, using fixed space meshes. Currently, the implementation of these methods on realistic systems hits upon the crucial issue of *accuracy against computational cost.*

Regarding this issue, we want to solve efficiently (and if possible optimally with respect to computational time and to memory occupancy) linear systems coming from the discretization of the electric or magnetic field formulations of the time-harmonic Maxwell's equations, by edge elements. Let

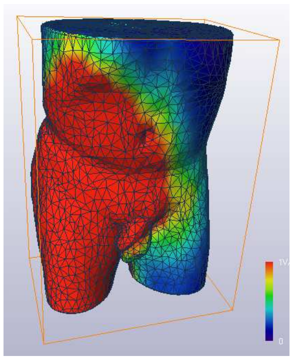


Figure 1.2: Hyperthermia RF (27MHz) for treating deep tumours. Magnitude of the electric field.

us recall that the electric field formulation of these equations can be written:

$$\left\{\begin{array}{l}\text{To find } \mathbf{E} \text{ in } \mathbf{Q} \text{ such that:}\\ a(\mathbf{E},\mathbf{E}') = \mathbf{F}(\mathbf{E}') \quad \forall \mathbf{E}' \in \mathbf{Q},\\ \text{with } a(\mathbf{E},\mathbf{E}') = \displaystyle\int_\Omega \frac{1}{\mu}\operatorname{curl}\mathbf{E}\cdot\operatorname{curl}\overline{\mathbf{E}'} - \omega^2\int_\Omega \tilde{\varepsilon}\mathbf{E}\cdot\overline{\mathbf{E}'}\\ \qquad\qquad\qquad + \mathrm{i}\displaystyle\int_{\Gamma_a}\frac{1}{\mu}\|\mathbf{k}\|(\mathbf{E}\times\mathbf{n})\cdot(\overline{\mathbf{E}'}\times\mathbf{n}),\end{array}\right. \tag{1.1}$$

where μ denotes the magnetic permeability, ε the complex-valued dielectric permittivity, ω the pulsation, $\mathbf{k}$ the wave vector, $\mathbf{n}$ the exterior unit normal, Γ_a the part of the boundary where absorbing conditions are applied and Γ_d the perfect electric conductor boundary ($\mathbf{E}\times\mathbf{n} = 0$ on Γ_d). The space of test functions is:

$$\mathbf{Q} = \{\mathbf{E}' \in \mathbf{H}(\operatorname{curl},\Omega) \ / \ \mathbf{E}'\times\mathbf{n} = 0 \text{ on } \Gamma_d\}.$$

The discretization of this formulation by lowest order edge elements leads to a system whose matrix is complex-valued, symmetric and indefinite, of the form: $K' = S_\mu - \omega^2 M_{\tilde{\varepsilon}} + \mathrm{i}M_{\mu,\Gamma_a}$. The matrices S_μ, $M_{\tilde{\varepsilon}}$ and M_{μ,Γ_a} denote respectively the discretization of the first, second and third integrals in Definition (1.1) of the bilinear form a.

We seek methods which must be efficient for *a priori* unstructured meshes *i.e.* where there is no hierarchy of nested grids. Algebraic multilevel methods enable us to generate representations of the problem at different scales, starting from diverse data: matrix coefficients only, or additional geometric information.

Here, we are more specifically interested in the difficulties due to the operator $\operatorname{curl}\mu^{-1}\operatorname{curl}$. First, we will give an overview of the known multi-level methods for Maxwell's equations, emphasising the so-called algebraic multilevel methods. Then, we will show an energy-minimising formulation which computes a coarse basis from a known fine nodal basis. Finally, we will explain an extension of these ideas to edge elements and we will give the results of numerical experiments performed on simple examples. This extension to coarse edge elements satisfies a fundamental geometric compatibility condition: in the same time, we construct a coarse nodal basis and a coarse edge basis in such a way that the gradient of coarse nodal elements are linear combinations of the coarse edge elements. This compatibility condition is known to be essential for the good performance of our methods.

Chapter 2

Known multilevel methods for Maxwell's equations

Known multilevel methods for Maxwell's equations are introduced. In order to help the reader, we first give some information relative to the classical multigrid algorithm. Then, we will study the specific difficulties of Maxwell's equations and conceivable approaches to deal with them.

2.1 Introduction to the multigrid method

We solve a second order partial differential equation on the domain Ω. We explain the main features of the multigrid method. First, the two-grid method is described. Then, it is extended into the complete multigrid algorithm.

2.1.1 Two-grid method

On the domain Ω, two distinct meshes are defined:

- a "coarse" mesh T_H of parameter H,
- a "fine" mesh T_h, obtained by refining the coarse mesh T_H, of parameter h.

Once the problem has been discretized by finite differences or finite elements, one must solve on the fine mesh the following linear system:

$$A_h u_h = b_h. \tag{2.1}$$

One would like to solve this system by using information given by the coarse mesh, where the linear system can be written:

$$A_H u_H = b_H.$$

Let us define:

- a *prolongation operator* $P_{H\to h}$ which enables us to transfer a solution, which is known on the coarse mesh, to a solution on the fine mesh,
- a *restriction operator* $R_{h\to H}$ which performs the inverse operation, from the fine to the coarse mesh.

The method can then be decomposed into three steps:

1. *Presmoothing*: it consists in using a linear iteration on the vector u_h (*e.g.*: one or two iterations of Gauss-Seidel). It can be written in matrix form:
$$u_h = u_h + M_h^{-1}\big(A_h u_h - b_h\big). \tag{2.2}$$
Here, M_h is an operator which will be chosen so that the highly oscillating components of the error are substantially reduced. An example of the effect of an appropriate smoother for the equation $-\triangle u + u = 0$ on the unit square with Neumann boundary conditions is displayed at Fig. 2.1. Once presmoothing is accomplished, the smooth part of the error is dominant. Therefore restricting it, through $R_{h\to H}$, to the coarse grid loses little information.

2. *Error correction*: it consists in solving on the "coarse" grid (usually by a direct method):
$$A_H \theta_H = R_{h\to H}(b_h - A_h u_h). \tag{2.3}$$
The result obtained enables us to "correct" the smooth iterate:
$$u_h \leftarrow u_h + P_{H\to h}\theta_H. \tag{2.4}$$
The correction step reduces the smooth component of the error, which is its only significant part on the coarse grid. The dimension of the system on the coarse grid is much lower than on the fine grid, and this is the interesting feature of the correction step.

3. *Postsmoothing*: analogous the presmoothing step on the "corrected" solution.

This method is iterative, the previous steps are repeated until the prescribed accuracy is reached (see Fig. 2.2).

However, the use of only two grids limits the performance of the method, because the resolution on the coarse grid can still lead to a large system.

2.1.2 Multigrid extension

Replacing the correction step by a two-level method leads to the recursively defined multilevel method.

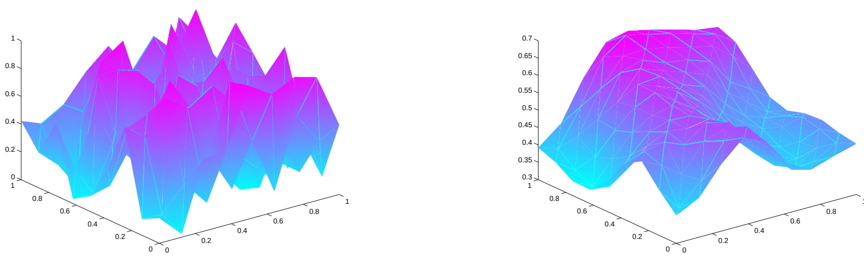


Figure 2.1: Non-smooth error and smooth error after 2 Gauss-Seidel iterations.

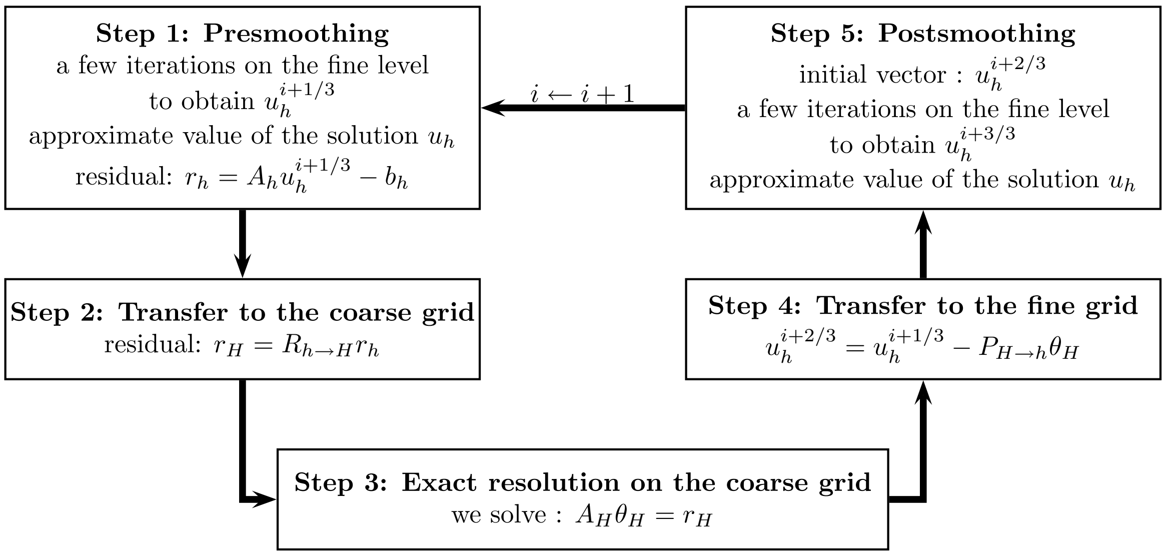

Step 1: Presmoothing
a few iterations on the fine level
to obtain $u_h^{i+1/3}$
approximate value of the solution u_h
residual: $r_h = A_h u_h^{i+1/3} - b_h$
$i \leftarrow i+1$
Step 5: Postsmoothing
initial vector : $u_h^{i+2/3}$
a few iterations on the fine level
to obtain $u_h^{i+3/3}$
approximate value of the solution u_h
Step 2: Transfer to the coarse grid
residual: $r_H = R_{h\to H} r_h$
Step 4: Transfer to the fine grid
$u_h^{i+2/3} = u_h^{i+1/3} - P_{H\to h}\theta_H$
Step 3: Exact resolution on the coarse grid
we solve : $A_H \theta_H = r_H$

Figure 2.2: Two-grid method

Let $T_1, ..., T_j$ be a hierarchy of meshes of respective parameters $h_1, ..., h_j$; T_k is a refinement of T_{k-1}, for all $k = 2 \ldots j$.

$P_{k-1\to k}$ and $R_{k\to k-1}$ refer to the transfer operators (prolongation and restriction) which transfer information forward and backward from level $k-1$ to level k.

A_k denotes the matrix of the problem when discretized on the mesh T_k.

Fig. 2.3 sketches the V-cycle[1], which is the simplest multigrid algorithm.

```
Initial solution vector : $u_j$.
Initial right-hand side : $b_j$.
MGVC(level : $l$, initial solution : $u_l$, right-hand side : $b_l$)
{

    if $j = 1$ then $u_1 \leftarrow A_1^{-1} b_1$
    else
    {
        $u_l \leftarrow \text{liss}(u_l, b_l)$, presmoothing
        $\theta_{l-1} \leftarrow 0$
        MGVC($l-1$, $\theta_{l-1}$, $R_{l\to l-1}(b_l - A_l u_l)$)
        $u_l = u_l + P_{l-1\to l}\theta_{l-1}$
        $u_l \leftarrow \text{liss}(u_l, b_l)$, postsmoothing
    }

}
```

Figure 2.3: V-cycle multigrid algorithm

The multigrid algorithm has been thought to be an optimal solver: the algorithmic complexity[2] and the quantity of data to store for its implementation (*i.e.* CPU time + occupied memory) must evolve linearly with the number of unknowns on the fine grid.

However, there is no valid multilevel algorithm which can solve all problems but only a generic principle, and we have to define:

- an *efficient smoother* which damps the oscillating part of the error and remains as little sensitive as possible to the parameters of the problem studied,

- a *hierarchy of levels* and especially the restriction and the prolongation operators; these must fit the properties of the problem under consideration: for instance, a specific treatment of the anisotropy or the heterogeneity can be required.

[1]One iteration of the multigrid method with only one recursive call in the algorithm.

[2]Number of elementary operations required to accomplish the task.

For the construction of grids, two distinct approaches can be distinguished:

- a geometric approach: a hierarchy of meshes is constructed,
- an algebraic approach: representations at different scales of the initial linear system are constructed algebraically.

Both approaches will be detailed later.

We will go back also to the smoother; it plays a key-rôle, which has been recently clarified for Maxwell's equations, see [1] and [7].

2.2 Properties of multilevel methods for Maxwell's equations

The use of a multilevel method for solving Maxwell's equations requires unusual smoothers dealing with the difficulties coming from the $\mathrm{curl}\,\mathrm{curl}$ operator. Moreover, generating algebraically a hierarchy of levels requires a special care.

2.2.1 Difficulties coming from the $\mathrm{curl}\,\mathrm{curl}$ operator

The classical smoothers (damped Jacobi, Gauss-Seidel...) strongly damp the "*high energy*" part of the error; for a positive bilinear form, the high energy part of the error corresponds to the modal components associated to the eigenvalues of largest modulus.

In the case of the Laplace operator, this part coincides with the oscillating part, which is why the classical smoothers are very efficient.

In contrast, for operators with a $\mathrm{curl}\,\mathrm{curl}$ term, there remains oscillating vectors of low energy. This is due to the existence of a large subspace (infinite dimensional in the continuous case and of dimension proportional to the number of unknowns in the discrete case) of curl-free vector fields, which have a quasi-null energy; indeed, the contribution of the $\mathrm{curl}\,\mathrm{curl}$ part is dominant in the energy.

Thus, the oscillating part contains a curl-free component on which the classical smoothers will have no effect. Moreover, this component cannot be correctly represented on the coarse grid because it is oscillating. It is then impossible to obtain the efficiency of the classical smoothers.

2.2.2 The discrete Helmholtz decomposition

Let T be a given mesh on the domain. Define the following objects:

- $\mathbf{Q}_k$ is the space of edge elements of order k on the mesh T. The construction of these finite element spaces was first proposed by Nédélec [10]. Hiptmair [8] proposed more recently an approach using differential forms.
- $\mathbf{Q}_{k,\Gamma_d} = \{\mathbf{E} \in \mathbf{Q}_k,\ n \times \mathbf{E} = 0 \text{ on } \Gamma_d \subset \partial\Omega\}$,
- S_k is the space of nodal elements of order k on the mesh T,
- $S_{k,\Gamma_d} = \{\phi \in S_k,\ \phi = 0 \text{ on } \Gamma_d \subset \partial\Omega\}$.

A discrete Helmholtz decomposition exists in the space $\mathbf{Q}_{k,\Gamma_d}$; if the domain is contractible, it can be written:

$$\mathbf{Q}_{k,\Gamma_d} = \operatorname{grad} S_{k,\Gamma_d} \oplus \mathbf{D}_{k,\Gamma_d}. \tag{2.5}$$

The notation grad S_{k,Γ_d} denotes the space of curl-free vector fields. Moreover, the direct sum is orthogonal with respect to the $\mathbb{L}^2$-norm. Observe that if the domain has holes, a space of small dimension, containing curl-free vector fields which are not gradients, must be added to the direct sum (2.5). $\mathbf{D}_{k,\Gamma_d}$ is spanned by weakly divergence-free fields *i.e.*:

$$A \in \mathbf{D}_{k,\Gamma_d} \ \Rightarrow\ (A, \operatorname{grad} \phi)_{\mathbb{L}^2(\Omega)} = 0,\ \forall \phi \in S_{k,\Gamma_d}$$

Some numericians (Hiptmair [7], Beck [2]) use this decomposition by solving simultaneously a scalar potential problem in the auxiliary nodal finite element space S_{k,Γ_d}.

2.2.3 Choice of the smoother

One-level overlapping Schwarz methods are used to find efficient smoothers for Maxwell's equations. The mathematical analysis relies on the Helmholtz decomposition to prove the feasibility of the multigrid approach for Maxwell's equations. Practical implementations use this decomposition explicitly or implicitly.

The two main references concerning the choice of smoothers for Maxwell's equations are Hiptmair [7] and Arnold et al. [1].

Hiptmair's smoother uses explicitly the Helmholtz decomposition, by smoothing at each iteration simultaneously, the whole problem discretized by edge elements, and an auxiliary scalar potential problem discretized by nodal elements.

The algorithm is given in Fig. 2.4.

In this figure, the operator T_l^* lets us transfer information from the edge element to the nodal element basis. T_l is its adjoint.

The principle of this algorithm has been reported in [7], [4], and similar references, which also contains numerical experiments, mainly in the

```
liss($x_l$, $b_l$)
{
    Gauss-Seidel iteration on $A_l x_l = b_l$
    $\rho_l \leftarrow b_l - A_l x_l$
    $\rho_l \leftarrow T_l^* \rho_l$
    $\psi_l \leftarrow 0$
    Gauss-Seidel iteration on $\triangle_l \psi_l = \rho_l$
    return to $x_l + T_l \psi_l$
}.
```

Figure 2.4: The smoother defined by Hiptmair.

transient case; see in particular the bibliography of [9] for a more complete list.

The smoother of Arnold et al. does not use explicitly Helmholtz' decomposition. An overlapping Schwarz preconditioner is used as a smoother. The space is then decomposed into a set of overlapping subspaces:

$$\mathbf{Q}_{k,\Gamma_d} = \sum \mathbf{Q}^i_{k,\Gamma_d}. \tag{2.6}$$

The subspace $\mathbf{Q}^i_{k,\Gamma_d}$ is generated by the shape functions associated to the edges connected by the common node i.

Additive or multiplicative Schwarz algorithms are used.

Proving that the multilevel method is optimal for time-harmonic Maxwell's equations, with the choice of either smoother, is done in [6]; however, observe that, in this article, absorbing boundary conditions are not used, the permittivity $\tilde{\varepsilon}$ is real and the domain Ω is convex.

Remark: We implemented these smoothers as a one-level preconditioner. The efficiency gains compared to the classical SSOR preconditioner are reported in [11], [12] and [13] for non-trivial geometry and data (plane, human body model).

2.2.4 Construction of the levels

The classical geometric multigrid approach is recalled but we concentrate mainly on algebraic multilevel methods.

Description of a geometric multilevel method

A convenient way of creating nested finite element spaces is to successively refine the elements, allowing then for a straightforward definition of the transfer (prolongation or restriction) operators.

By construction, this class of method is perfect for adaptive refinement[3]: the grids are constructed successively and the already computed results are used to continue to work on the finest grid.

However, this “geometric” multilevel method requires the use of specific mesh generators that generate a sequence of rather regular grids.

Let us eventually observe that theoretical results are better known for the geometric multilevel algorithms.

Description of an algebraic multilevel method

The principle of the method is to begin with the fine mesh and to generate algebraically a coarse level.

For instance, the simplest approach needs to know only the incidence graph of the matrix on the finest level. Some improvements are possible by taking into account the coefficients, geometric information or elementary matrices computed during the global assembly of the matrix.

Moreover, the operator on the coarse level is usually assembled by using Galerkin method[4].

The quality of the method comes from the good complementarity between the transfer operators and the smoother.

The first algebraic multilevel approach for Maxwell’s equations are described in the articles of Beck [2], [3] and of Reitzinger and Schöberl [14].

Both use a scalar potential matrix A_ϕ for the construction of the different levels. This approach coincides with the search for a good representation of the kernel of the curl operator.

The *node-edge incidence graph* of the mesh can be read from A_ϕ. Indeed, the node i and j are linked by an edge if and only if the coefficient $A_\phi(i, j)$ does not vanish.

In the following, we denote by *master nodes*, the nodes belonging to the coarse mesh and by *slave nodes*, all others.

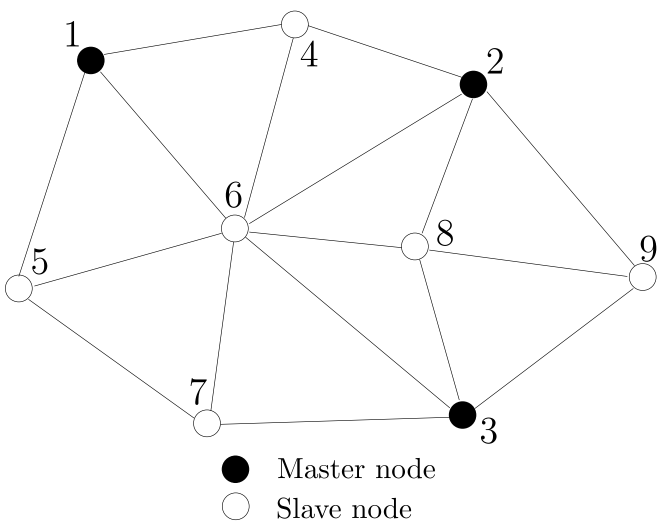
The principles of the approaches of Beck and of Reitzinger and Schöberl, are recalled and some recent improvements by Bochev et al. are also exposed (see [5]).

Method of R. Beck This method is used to compute transient or time-harmonic formulations at low frequencies. Beck uses the smoother defined by Hiptmair.

In order to select the coarse nodes, Beck introduces a simple algorithm (Algorithm 2 [2]), which does not use the values of the coefficients of A_ϕ. Then, in order to define the restriction operators, he enforces the following rule: the value of each master node is transfered with a weight of 1, the

[3]The mesh is only refined in the area where the error will be large.

[4]If α is the prolongation operator, one obtains $A_H = \alpha^t A_h \alpha$.



$$\alpha^t = \begin{pmatrix} 1 & 0 & 0 & 1/2 & 1 & 1/3 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1/2 & 0 & 1/3 & 0 & 1/2 & 1/2 \\ 0 & 0 & 1 & 0 & 0 & 1/3 & 1 & 1/2 & 1/2 \end{pmatrix}$$

Figure 2.5: Representation of master and slave nodes and of the restriction operator α^t.

value of a slave node s connected with n_s master nodes is transfered with a weight of n_s^{-1} to each master node connected with s (Fig. 2.5). Thus, he only uses the incidence graph of the matrix A_ϕ, which represents also usually the mesh.

However, Beck does not define the transfer operators directly in the edge element space.

His multilevel algorithm is applied to an auxiliary basis of vectors whose three components are nodal P_1 elements. He defines a transfer operator from his edge element space to this auxiliary nodal element space, on the same mesh.

Afterwards, he applies the strategy used for scalar problems, in order to determine the coarse level, and to define the transfer operators; indeed, he only works with nodal elements.

This transfer form edge to vector nodal elements creates a problem for the boundary conditions: they are natural in the edge element framework, but they are not adapted to the auxiliary vector nodal space[5]. Beck introduces techniques that solve these questions; see [2] Section 5.

This multilevel algorithm can *only be used as a preconditioner*. The

[5]The trace operator on the boundary are different: tangential component for $\mathbb{H}(\text{rot}, \Omega)$, all the components for $\mathbb{H}^1(\Omega)$.

method gives satisfactory results, which are quasi-optimal for the problems that Beck considers. It does not take into account the specific properties of edge elements.

Method of S. Reitzinger and J. Schöberl As Beck, Reitzinger and Schöberl use the knowledge of the matrix A_ϕ to define the coarse level and the transfer operators. The method is applied to magnetostatic and time-harmonic problems at low frequencies.

However, the transfer operators are differently defined. Reitzinger and Schöberl make a partition of the mesh nodes. This partition uses the knowledge of A_ϕ and is built in two steps:

- the master nodes are selected by using previously developed algebraic techniques (algorithm of Rüge and Stüben for example [15]),
- the aggregates are created by associating uniquely the remaining nodes to one master node.

Thus, every node of the fine mesh is *uniquely associated* to one master node (see Fig. 2.6).

They define the index mapping:

$$\text{ind} : \omega_h^n \to \omega_H^n,$$

where ω_h^n denotes the set of node indices on the fine mesh and ω_H^n those on the coarse mesh.

The transfer operator only contains zeroes and ones and is defined as:

$$\alpha_{ij} = \begin{cases} 1 & \text{if } i \in \omega_h^n \text{ such that } j = \text{ind}(i), \\ 0 & \text{else.} \end{cases} \tag{2.7}$$

Remark: In order to be able to continue the multilevel construction, on the coarse level, the scalar potential matrix can be assembled in the form: $\alpha^t A_\phi \alpha$.

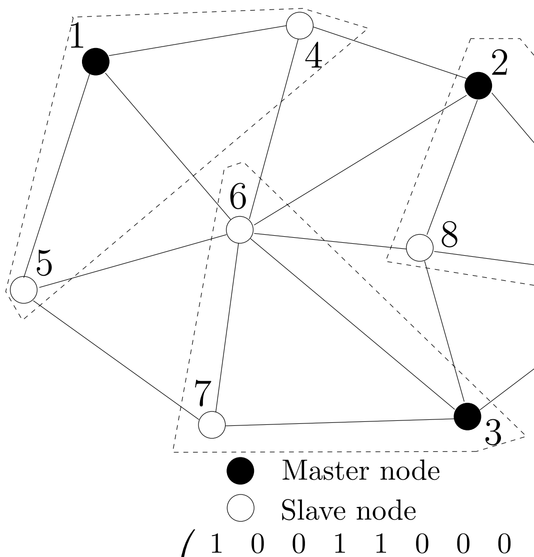
In contrast with Beck, Reitzinger and Schöberl *directly define prolongation operators between edge element spaces from the fine to the coarse level.*

They define their transfer operator in order to verify the following commutativity relation:

$$a\alpha = \beta A. \tag{2.8}$$

where:

- α denotes the prolongation operator for nodal elements,
- β denotes the prolongation operator for edge elements,
- a and A denote respectively the discrete analogue of the grad operator at the fine and coarse levels.



$$\alpha^t = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Figure 2.6: Representation of master and slave nodes and of the restriction operator α^t.

This relation means that the gradient of coarse nodal elements are linear combinations of coarse edge elements. A detailed explanation will be given below in Subsection 4.1.1. Its interest is that *the geometric representation of the kernel of the* curl *operator is preserved during the transfer from one grid to the other* and the vectors of the "coarse" kernel are well interpolated by these on the "fine" kernel.

Let ω_h^e be the set of indices of the fine level edges and ω_H^e be those of the coarse level "edges". Let $i = (i_1, i_2) \in \omega_h^e$ be the representation of the edge linking the node i_1 to the node i_2 and set $j = (j_1, j_2) \in \omega_H^e$.

The transfer operator is defined in the following way:

$$\beta_{ij} = \begin{cases} 1 & \text{if } j = (\text{ind}(i_1), \text{ind}(i_2)), \\ -1 & \text{if } j = (\text{ind}(i_2), \text{ind}(i_1)), \\ 0 & \text{else.} \end{cases} \tag{2.9}$$

This operator verifies the commutativity relation (2.8).

Unfortunately, the efficiency of this multilevel method is not quite optimal, which may be due to the simplicity of the edge interpolation. However, this approach has been extended by the Sandia team, R. Bochev et al. [5], who improve the construction of the transfer operators.

Improvements of transfer operators The improvements introduced by the Sandia team follow two directions.

The first one concerns the prolongation operator for edge elements. The trick is the construction of a coarse basis, verifying the commutativity relation (2.8) and whose energy is as low as possible, in order to ensure good properties of the prolongation. The construction is iterative. Let us recall that it is equivalent to seek a basis, or the matrix β. The method used is the following:

- a first operator $\hat{\beta}$, satisfying the commutativity relation is defined: for instance, the operator of Reitzinger and Schöberl,
- it is smoothed by using one or several damped Jacobi iteration(s):

  $$\beta = (I - \gamma D^{-1} S_\mu)\hat{\beta}, \tag{2.10}$$

  where I is the identity matrix, γ a relaxation coefficient, S_μ denotes the "curl μ^{-1} curl" part of the matrix of the system to be solved, and D is the diagonal matrix made out of the diagonal coefficients of S_μ.

The new basis is given by the columns of β. The energy of the basis given by β is less than the energy of the basis given by $\hat{\beta}$. Moreover, the commutativity relation remains verified even after smoothing. Observe, however, that Bochev et al. consider the case where the matrix is symmetric definite positive, and that the support of the coarse basis functions spreads out and usually only one Jacobi iteration is used.

The second improvement consists in taking a better nodal prolongation operator and constructing an edge prolongation operator, which still verifies Condition (2.8). Then they may use the first improvement again (see [5]).

2.3 Conclusion

The principle of the multilevel methods has been recalled and some features of these methods for Maxwell's equations discretized by edge elements have been described.

An efficient method will have to verify the commutativity relation (2.8). We will return to this question, when using coarse bases, which are slightly different from those proposed by Reitzinger and Schöberl.

Chapter 3

Energy-minimising construction of coarse nodal elements for multilevel methods

Two-level algebraic methods are introduced for solving linear systems coming from a finite element discretization.

Two methods are considered: one takes into account the boundary conditions, the other does not. Indeed, the choice of the coarse level seems to be sensitive to the presence of Dirichlet boundary conditions.

A simple problem is considered on an elementary geometry. The strong formulation of the test case is:

$$\begin{cases} -\triangle u = f \text{ in } \Omega =]0;1[\times]0;1[, \\ \quad u = 0 \text{ on } \partial\Omega, \\ \quad f = \pi \sin(\pi x)\sin(\pi y). \end{cases} \tag{3.1}$$

The weak formulation is the following:

$$\begin{cases} \text{To find } u \in \mathrm{H}_0^1(\Omega) \text{ such that:} \\ \displaystyle\int_\Omega \operatorname{grad} u \cdot \operatorname{grad} v = \int_\Omega f v \quad \forall v \in \mathrm{H}_0^1(\Omega). \end{cases} \tag{3.2}$$

3.1 Principle of the method

An unstructured mesh is given on the domain Ω where a discretization by P_1 finite elements is used.

In the two-level method proposed for solving the linear system, the main point is the construction of the coarse basis. This basis must satisfy the following constraints:

- the basis must span a subspace of the initial P_1 finite element space;
- the support of every coarse basis function is a subdomain Ω_i of Ω. Ω_i is defined so that the matrix at the coarse level has a very regular structure. This regular structure will be exhibited later;
- Given the supports of these coarse functions, an optimisation problem is solved to compute them; it consists in minimising the energy of the coarse basis under appropriate constraints.

3.1.1 Formulation of the optimisation problem

Suppose that Ω is decomposed into overlapping subdomains $(\Omega_i)_{i=1\ldots d_H}$ and that each subdomain is not completely overlapped by its neighbours. $(I_i)_{i=1\ldots d_H}$ denotes the set of nodes belonging to each subdomain Ω_i.

Let $(\Phi_i)_{i=1..d_H}$ be the coarse basis functions. This basis is the solution of the following problem:

$$\left\{\begin{array}{l} \text{To find } (\Phi_i)_{i=1..d_H} \text{ minimising } \displaystyle\sum_{i=1}^{d_H} a(\Phi_i,\Phi_i) \text{ under the constraints:} \\ \displaystyle\sum_{j=1}^{d_H} \Phi_j(x) = 1,\ \forall x \in \overline{\Omega} \text{ and } \mathrm{supp}(\Phi_i) \subset \Omega_i,\ \forall i = 1\ldots d_H. \end{array}\right. \tag{3.3}$$

Observe that $\mathrm{H}^1_0(\Omega)$ is equipped with an energy norm by the bilinear form $a(u,v) = \int_\Omega \mathrm{grad}\, u \cdot \mathrm{grad}\, v$; it would be a semi-norm if Neumann conditions were assigned, and then we would work in $\mathrm{H}^1(\Omega)$, as is the case in [16].

Let $(\phi_i)_{i=1..d_h}$ be the fine basis on the initial mesh. The coarse space is included in the fine one, and thus it is possible to write:

$$\begin{array}{c} \forall i = 1\ldots d_H,\ \Phi_i = \displaystyle\sum_{j=1}^{d_h} \phi_j \alpha_{ji} \\ \text{with } \alpha_{ji} = 0 \text{ if } j \notin I_i \text{ (support constraints).} \end{array} \tag{3.4}$$

The prolongation operator for the two-level method is denoted by α, which is a $d_h \times d_H$ matrix.

In order to describe the problem in an algebraic fashion, a matrix P_j is introduced, with j varying from 1 to d_H. Denoting by e_i the i-th vector of the canonical basis of $\mathbb{R}^{d_h}$, the rows of P_j are the vector e_i^t whose index i belongs to I_j. The dimension of P_j is $n_j \times d_h$ where n_j is the number of nodes in I_j.

Then, d_H vectors $(\alpha_i)_{i=1\ldots d_H}$ of respective dimension n_i can be defined so that: $\alpha_i = P_i \alpha_{\bullet i}$. Here and in what follows, $\alpha_{\bullet i}$ denotes the column vector $(\alpha_{ji})_j$. Let K be the matrix whose elements are $a(\phi_j, \phi_i)$ and let $K_i = P_i K P_i^t$; its dimension is $n_i \times n_i$.

Then, the problem takes the matrix form:

$$\begin{cases} \text{To find } (\alpha_i)_{i=1\dots d_H} \text{ minimising } \displaystyle\sum_{i=1}^{d_H} \alpha_i^t K_i \alpha_i \text{ under the constraints:} \\ \displaystyle\sum_{i=1}^{d_H} P_i^t \alpha_i = 1_{d_h \times 1}. \end{cases} \tag{3.5}$$

The notation $1_{d_h \times 1}$ denotes a column vector with d_h components equal to 1.

3.1.2 Method of resolution

This optimisation problem with constraints is solved by introducing Lagrange multipliers.

An iterative method is used to compute the multiplier vector. The principle of the resolution is sketched.

Let us define:

$$\alpha = \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_{d_H} \end{pmatrix}, \quad Q = \operatorname{diag}(K_i,\ i = 1 \dots d_H), \quad B = \begin{pmatrix} P_1 \\ \vdots \\ P_{d_H} \end{pmatrix}, \quad \gamma = 1_{d_h \times 1}. \tag{3.6}$$

The notation $\operatorname{diag}(A_i,\ i = 1 \dots n)$ refers to a block-diagonal matrix, whose diagonal blocks are the A_i's, $i = 1 \dots n$. Denoting the Lagrange multiplier vector by $\mu \in \mathbb{R}^{d_h}$, the optimisation problem takes the form:

$$\begin{cases} \text{To find the saddle-point } (\alpha_c, \mu_c) \text{ of the Lagrangian } \mathcal{L} \text{ defined by:} \\ \mathcal{L}(\alpha, \mu) = \dfrac{1}{2} \alpha^t Q \alpha + \mu^t (B^t \alpha - \gamma). \end{cases} \tag{3.7}$$

The critical point of $\mathcal{L}$ must then satisfy the following equations:

$$\begin{cases} Q\alpha_c = -B\mu_c \\ B^t \alpha_c = \gamma. \end{cases} \tag{3.8}$$

This system can be solved in the following way:

- first, compute the multiplier vector μ_c by an iterative method applied to the system:

$$B^t Q^{-1} B \mu = -\gamma, \text{ i.e. } \sum_{i=1}^{d_H} P_i^t K_i^{-1} P_i \mu = -\gamma.$$

- then obtain α by solving:

$$Q\alpha = -B\mu_c, \text{ i.e. } \alpha_i = -K_i^{-1} P_i \mu_c,\ \forall i = 1 \dots d_H.$$

All the K_i's are symmetric positive definite, and so is $B^tQ^{-1}B$. Then the system for μ can be solved by the conjugate gradient method. The factorisation of every K_i, which is a local representation of the matrix of the problem, is needed for the matrix-vector product. This product is implemented as follows:

1. for $i = 1 \dots d_H$ compute $b_i \leftarrow R_i \mu$,
2. for $i = 1 \dots d_H$ solve $K_i x_i = b_i$,
3. finally compute $\sum_{i=1}^{d_H} P_i^t x_i$.

Of course, the multiplications by P_i and P_i^t can be computed very fast because these matrices only contain n_i non-zero coefficients. Observe that the matrix-vector product in the conjugate gradient algorithm does not need to assemble the matrix, which is too expensive in computational time and memory.

The computation of the α_i's can be simply obtained from the factorisations of the local matrices K_i.

3.1.3 Decomposition into subdomains

The subdomain decomposition is constructed as follows (see Fig. 3.1):

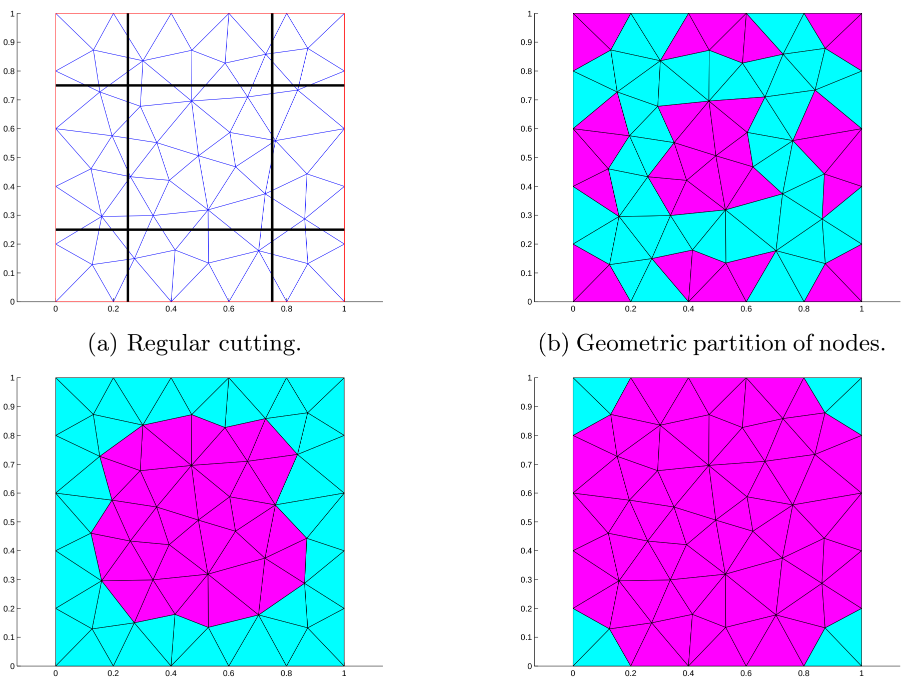
1. the domain Ω is cut according to regular geometric shapes;
2. the nodes are partitioned according to the geometric partition;
3. each node set is extended by taking all its nearest neighbours, in order to increase the overlap between subdomains. Thus, d_H overlapping node sets $(I_i)_{i=1..d_H}$ have been created;
4. the subdomain Ω_i is the union of all the elements which have a node of the set I_i, as one of their vertices.

3.1.4 Quantitative information on the different meshes

The results obtained for the algorithms are compared on the three meshes shown in Fig. 3.2. The number of nodes and elements of these meshes, are given in Table 3.1.

	Number of elements	Number of nodes
Mesh 3.2(a)	312	177
Mesh 3.2(b)	1248	665
Mesh 3.2(c)	4992	2577

Table 3.1: Quantitative information on meshes.



(a) Regular cutting.

(b) Geometric partition of nodes.

(c) Central set: overlapping by extension to the nearest neighbours.

(d) Central set: corresponding subdomain.

Figure 3.1: Decomposition into subdomains. Observe that this mesh is very coarse, and consequently, the central subdomain Ω_i is close to the whole domain Ω.

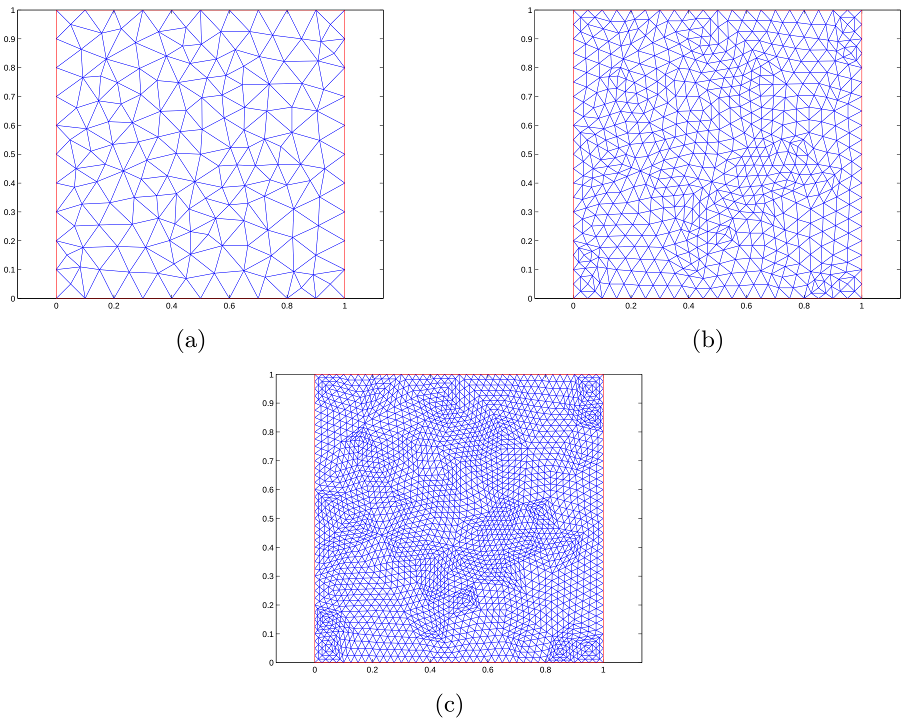


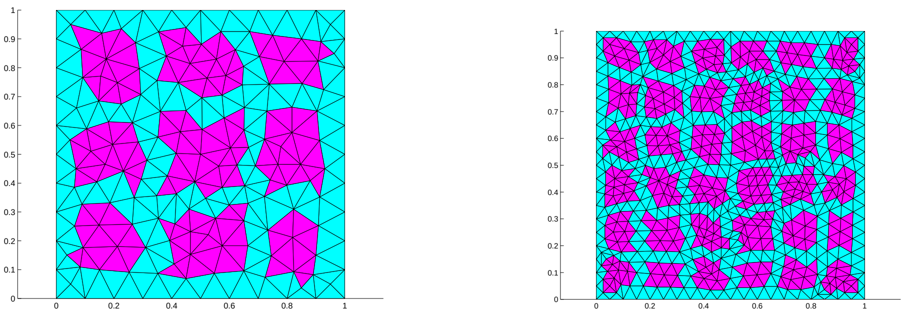
Figure 3.2: Mesh 3.2(a): $h_{\max} < 0.1$, meshes 3.2(b) and 3.2(c) are obtained by regular refinement.

3.2 Dirichlet fine and Dirichlet coarse bases (DFDC)

In this case, the constraint is imposed only on the interior nodes. Moreover, every coarse basis functions is a linear combination of the fine basis functions from the finite element space included in $H_0^1(\Omega)$.

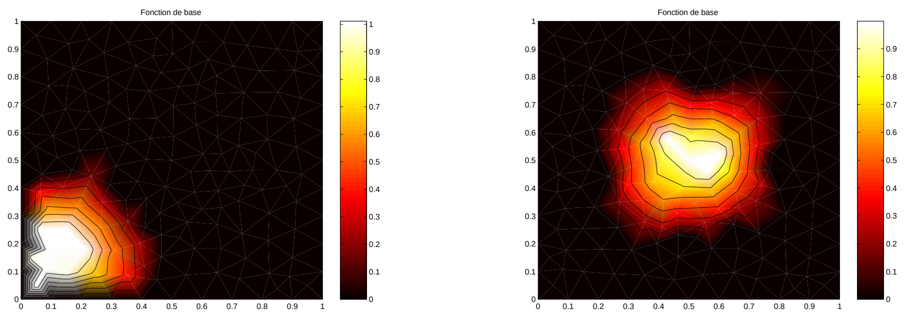
The domain Ω is subdivided into squares of length: $H = 1/\sqrt{d_H}$. The lexicographical numbering is used for the subdomains.

In Fig. 3.3, the partitions obtained with meshes 3.2(a) and 3.2(b) are presented. After solving the optimisation problem, one obtains the coarse basis functions. Two examples are shown in Fig. 3.4, they are obtained from mesh 3.2(a).



(a) (b)

Figure 3.3: Node partitions for meshes 3.2(a) and 3.2(b).



(a) (b)

Figure 3.4: Basis functions on the boundary and in the interior of the domain, generated with Dirichlet boundary conditions.

In Fig. 3.4(a), observe the large gradient of the basis function on the boundary where Dirichlet conditions are imposed: the isovalue curves are very close. This generates large diagonal coefficients, though the problem is homogeneous. This is demonstrated on the stiffness matrix (3.9) of the coarse level for mesh 3.2(a) and for the coefficients $(1,1)$, $(3,3)$, $(7,7)$ and $(9,9)$.

$$\begin{pmatrix} 11.3 & -0.31 & 0 & -0.54 & -0.46 & 0 & 0 & 0 & 0 \\ -0.31 & 7.17 & -0.22 & -0.46 & -0.85 & -0.45 & 0 & 0 & 0 \\ 0 & -0.22 & 9.88 & 0 & -0.37 & -0.24 & 0 & 0 & 0 \\ -0.54 & -0.46 & 0 & 7.56 & -0.67 & 0 & -0.29 & -0.20 & 0 \\ -0.46 & -0.85 & -0.37 & -0.67 & 4.99 & -0.97 & -0.46 & -0.98 & -0.24 \\ 0 & -0.45 & -0.24 & 0 & -0.97 & 7.03 & 0 & -0.4 & -0.22 \\ 0 & 0 & 0 & -0.29 & -0.46 & 0 & 10.22 & -0.54 & 0 \\ 0 & 0 & 0 & -0.20 & -0.98 & -0.4 & -0.54 & 6.71 & -0.38 \\ 0 & 0 & 0 & 0 & -0.24 & -0.22 & 0 & -0.38 & 10.81 \end{pmatrix} \tag{3.9}$$

If we only look at its non-zero entries, we can see that the matrix (3.9) has a regular structure, which is comparable to that which would be obtained with a nine-point stencil and a lexicographical numbering. Moreover, all the off-diagonal entries are negative.

3.3 Neumann fine and Dirichlet coarse bases (NFDC)

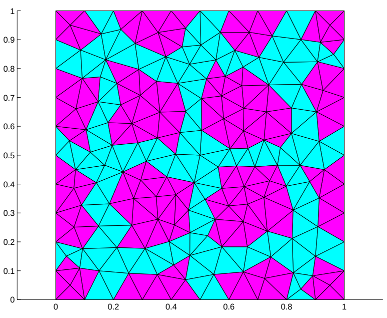
In order to avoid the boundary effects that appeared in the previous method and to enforce on the whole domain $\overline{\Omega}$ the constraint $\sum_i \Phi_i(x) = 1$, we will define coarse elements satisfying a Neumann condition on the boundary of the large domain. Moreover we adopt a different cutting strategy (compare Fig. 3.3 and Fig. 3.5) and we discard the coarse elements on the boundary.

The decomposition is indeed adapted to this NFDC method; the coarse size is the same as in the previous method but boundary elements have normal size $H/2$; we choose $H = (\sqrt{d_H} - 1)^{-1}$. With this choice, all the fine basis functions contribute to coarse elements. The lexicographical numbering is kept for the subdomains.

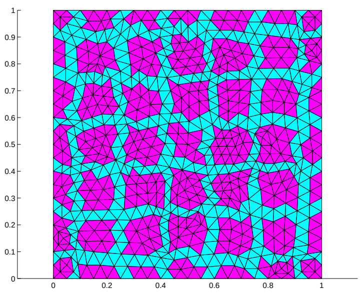
In Fig. 3.5, the partitions obtained with meshes 3.2(a) and 3.2(b) are presented. After solving the optimisation problem, one obtains the coarse basis functions. Two examples are shown in Fig. 3.6, corresponding to a computation on mesh 3.2(a).

Fig. 3.6(a) demonstrates the isovalue lines of a basis function whose support intersects the boundary. In comparison with Fig. 3.4(a), we observe that the gradients here are smaller. However, this basis function is not used in the coarse basis; indeed, on one hand, Dirichlet conditions are imposed on the original problem and on the other hand, we have enough basis functions.

In the case of partition 3.5(a), only four coarse basis functions remain after removing the basis functions whose support intersects the boundary. The coarse stiffness matrix corresponding to this choice of coarse basis functions

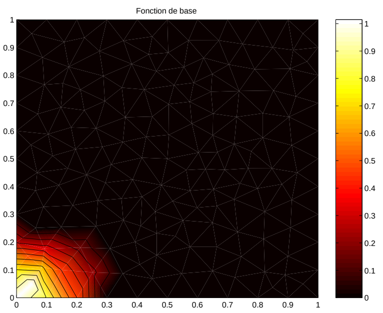


(a)

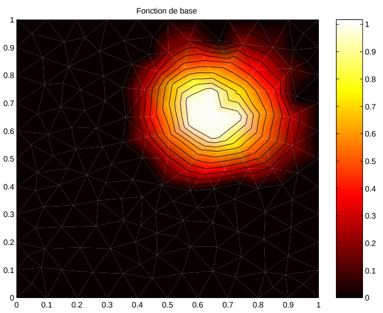


(b)

Figure 3.5: Node partitions for meshes 3.2(a) and 3.2(b).



(a)



(b)

Figure 3.6: Basis functions on the boundary and in the interior of the domain, generated with Neumann boundary conditions.

is:

$$\begin{pmatrix} 5 & -1.06 & -0.8 & -0.47 \\ -1.06 & 4.82 & -0.21 & -1.01 \\ -0.8 & -0.21 & 4.38 & -0.82 \\ -0.47 & -1.01 & -0.82 & 4.99 \end{pmatrix} \tag{3.10}$$

Observe that the variation of coefficients is much smoother than in matrix (3.9). In Fig. 3.7, we display the regular structure of the non-zero entries in the coarse matrix used for solving the problem on mesh 3.2(b) with partition 3.5(b).

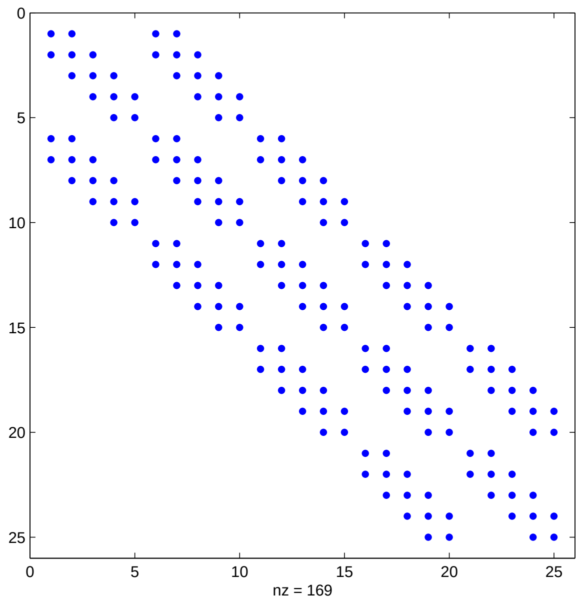


Figure 3.7: Pattern of the non-zero entries in the coarse matrix associated with mesh 3.2(b).

3.4 Adjusting the matrix at the coarse level

The construction of the coarse basis is only the first step towards our aim. It allows us to obtain a matrix with a regular pattern of non-zero elements. Moreover, we observe that every point receives contributions only from its eight direct neighbours and that the similar connectivity relations (diagonal, lateral neighbours) yield coefficient of the same order, which was one of the aims of this construction.

The idea is then to build a mean nine-point stencil and to assemble a new coarse matrix using only this stencil; we can even rewrite it on the old matrix because the pattern is preserved. We conjecture that the new coarse matrix is spectrally equivalent to the old one.

Let us begin by case (3.10), which has only one block. If the mean of the coefficients is calculated separately for each kind of connectivity relation between coarse basis functions *i.e.* the means of, respectively, central, vertical and horizontal, and diagonal interactions, we obtain the matrix:

$$\begin{pmatrix} 4.8 & -0.92 & -0.92 & -0.34 \\ -0.92 & 4.8 & -0.34 & -0.92 \\ -0.92 & -0.34 & 4.8 & -0.92 \\ -0.34 & -0.92 & -0.92 & 4.8 \end{pmatrix} \tag{3.11}$$

that corresponds to the nine-point stencil given by:

$$\begin{matrix} -0.34 & -0.92 & -0.34 \\ -0.92 & 4.8 & -0.92 \\ -0.34 & -0.92 & -0.34 \end{matrix} \tag{3.12}$$

More generally, studying Fig. 3.7 indicates a block-Toeplitz-Toeplitz-block (BTTB) pattern but the matrix itself is not BTTB. Then, we compute the means of the coefficients corresponding to the similar connectivity relations and we obtain the nine-point stencil given by (3.13):

$$\begin{matrix} -0.36 & -0.83 & -0.36 \\ -0.83 & 4.72 & -0.83 \\ -0.36 & -0.83 & -0.36 \end{matrix} \tag{3.13}$$

By using this mean stencil, the matrix which is obtained is BTTB. This property allows us to consider using fast direct solvers.

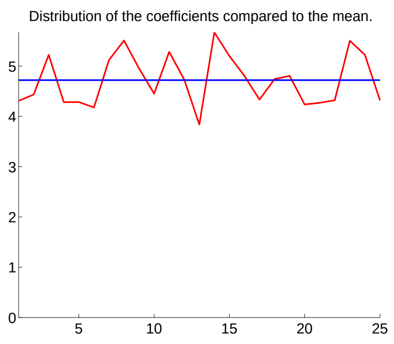
In order to evaluate the accuracy of this process and more precisely the distance to the original matrix, Fig. 3.8 and Table 3.2 give a statistical view of the results. The stencil obtained by using the means is called the mean stencil.

Coefficient	Central	Horizontal	Vertical	Diagonal
Mean	4.72	-0.82	-0.84	-0.36
Standard deviation	0.49	0.24	0.20	0.09
Maximal deviation	0.95	0.35	0.27	0.16
L^1 deviation	0.43	0.20	0.18	0.08

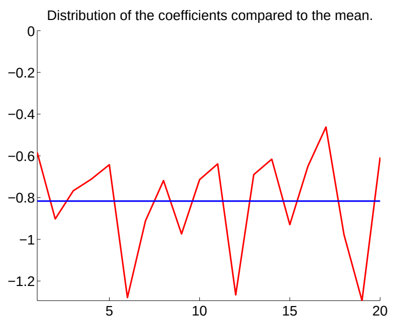
Table 3.2: Statistics comparing the mean stencil with the original coarse basis matrix.

3.5 Comparison of the number of iterations and of the conditioning on different meshes

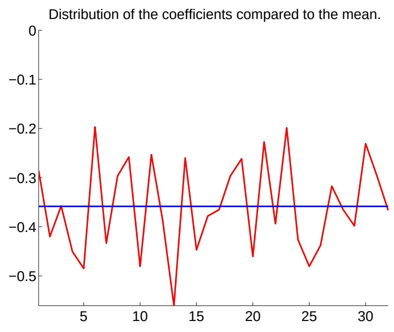
For solving the linear systems coming from the discretization by finite elements, the stopping criterion is $\|r\|_2 \leq 10^{-10}\|r_0\|_2$, with r being the current residual and r_0 being the initial residual.



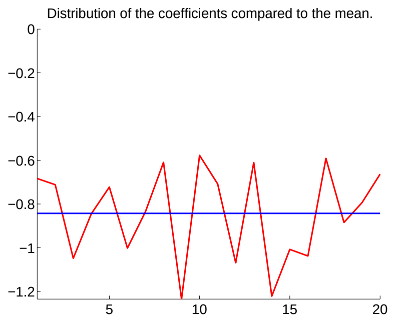
(a) Central.



(b) Horizontal.



(c) Diagonal.



(d) Vertical.

Figure 3.8: Distribution of the coefficients compared to the mean coefficient: central, horizontal, diagonal and vertical.

Table 3.3 compares for different cases the number of unknowns on the coarse grid.

	DFDC	NFDC
Mesh 3.2(a)	9	4
Mesh 3.2(b)	36	25
Mesh 3.2(c)	144	121

Table 3.3: Dimensions of the coarse space.

Table 3.4 compares the condition number based on the 2-norm of the matrix BA where A is the matrix of the system and B the preconditioning matrix. Five different preconditioning methods are used:

- symmetric Gauss-Seidel (GSsym);
- two-level method with one presmoothing and one postsmoothing, using symmetric Gauss-Seidel and 4 different kinds of coarse matrix, DFDC or NFDC, with or without using the mean stencil.

	Mesh 3.2(a)	Mesh 3.2(b)	Mesh 3.2(c)
Without preconditioning	56.7	273.5	1343.5
Symmetric Gauss-Seidel	20	81	304.8
V-cycle$(1,1)$ GSsym smoothing — DFDC	2.5	4.1	8.9
Idem + mean stencil	3.4	7.6	19.7
V-cycle$(1,1)$ GSsym smoothing — NFDC	2.3	3.85	7
Idem + mean stencil	2.3	4	7.8

Table 3.4: Condition number based on the 2-norm of the matrix BA, B being the preconditioning matrix.

We observe that the two-level methods are comparable in the case when we use the original matrix on the coarse space. Moreover, if the mean stencil is used, the best case is with the NFDC method (see Section 3.3). In this case, we obtain a similar behaviour for the original matrix and for the BTTB matrix deduced from the mean stencil.

Let us add that the behaviour of two-level methods does not seem optimal, although it is much more effective than the one-level method.

In Table 3.5, the number of iterations for solving the linear system with the same preconditioning techniques is given.

We can also benefit from the factorisations accomplished during the computation of the coarse basis functions, in order to implement a preconditioner or a smoother using the multiplicative Scharwz method (GSsymblc). The

	Mesh 3.2(a)	Mesh 3.2(b)	Mesh 3.2(c)
Without preconditioning	45	98	X
Symmetric Gauss-Seidel	23	45	88
V-cycle$(1,1)$ GSsym smoothing — DFDC	12	15	18
Idem + mean stencil	12	19	25
V-cycle$(1,1)$ GSsym smoothing — NFDC	13	15	17
Idem + mean stencil	13	15	17

Table 3.5: Number of iterations for solving the linear system for different preconditioners. X means that the algorithm did not converge after 100 iterations.

	Mesh 3.2(a)	Mesh 3.2(b)	Mesh 3.2(c)
Symmetric block Gauss-Seidel DFDC	8	12	22
Symmetric block Gauss-Seidel NFDC	7	12	22
V-cycle$(1,1)$ GSsymblc smoothing — DFDC	5	7	10
Idem + mean stencil DFDC	5	8	13
V-cycle$(1,1)$ GSsymblc smoothing — NFDC	5	7	9
Idem + mean stencil NFDC	5	7	9

Table 3.6: Number of iterations for solving the linear system for different preconditioners.

number of iterations needed then to solve the system are reported in Table 3.6.

3.6 Comparison for other boundary conditions

We infer from the numerical experiments reported in Table 3.4, 3.5 and 3.6 that the NFDC method is the best. In this section, we explore the applicability of this method to problem involving boundary conditions which are not everywhere Dirichlet conditions. Thus, we replaced the Dirichlet conditions on some parts of the boundary by periodicity conditions. The problem to solve can then be written:

$$\begin{cases} -\triangle u = f \text{ on } \Omega =]0;1[\times]0;1[, \\ u(0,y) = u(1,y) = 0 \ \forall y \in]0;1[, \\ u(x,0) = u(x,1) \ \forall x \in]0;1[, \\ f = 5\pi \sin(\pi x)\sin(2\pi y). \end{cases} \tag{3.14}$$

We solved this linear system and we obtained the results gathered in Table 3.7.

	Mesh 3.2(a)	Mesh 3.2(b)	Mesh 3.2(c)
Without preconditioning	53	X	X
Symmetric Gauss-Seidel	26	51	99
V-cycle$(1,1)$ GSsym smoothing — NFDC	13	16	17
Idem + mean stencil	14	16	17

Table 3.7: Number of iterations for solving the linear system for different preconditioners. X means that the algorithm did not converge after 100 iterations.

Then, we also added Neumann conditions on the part of the boundary where $x = 1$ and changed the right-hand side ($f = 4.25\sin(\frac{\pi}{2}x)\sin(2\pi y)$). The results obtained are gathered in Table 3.8. There is now a difference between the results with the mean stencil or the initial matrix. This should be due to the rough way used to compute the mean stencil or to the disagreement between the boundary conditions implicitly chosen by the mean stencil and the true boundary conditions.

	Mesh 3.2(a)	Mesh 3.2(b)	Mesh 3.2(c)
Without preconditioning	62	X	X
Symmetric Gauss-Seidel	29	58	X
V-cycle$(1,1)$ GSsym smoothing — NFDC	14	15	17
Idem + mean stencil	16	20	25

Table 3.8: Number of iterations for solving the linear system for different preconditioners. X means that the algorithm did not converge after 100 iterations.

3.7 Some results about the Helmholtz equation

In order to test the robustness of the NFDC algorithm and in view of the applications, Laplace's equation is replaced by Helmholtz' equation. The formulation of the problem then becomes:

$$\begin{cases} -\triangle u - k^2 u = f \text{ on } \Omega =]0;1[\times]0;1[, \\ u = 0 \text{ on } \partial\Omega, \\ f = (2\pi - k^2)\sin(\pi x)\sin(\pi y). \end{cases} \tag{3.15}$$

The wave number is denoted by k. It can also be written $k = 2\pi/\lambda$ where λ is the wavelength. Thus, a study can be performed with varying wavelength. An interesting comparison can be made by following the evolution of the number of iterations as a function of the ratio H/λ where H is the width of the initial subdomains. The results given in Table 3.9 are obtained with mesh 3.2(b).

H/λ	1/12	1/6	1/3	5/12	1/2
Without preconditioning	X	X	X	X	X
Symmetric Gauss-Seidel	48	57	92	X	X
V-cycle(1, 1) GSsym smoothing NFDC	16	20	42	48	87
Idem + mean stencil	16	19	42	47	87

Table 3.9: Number of iterations for solving the linear system for different preconditioners. X means that the algorithm did not converge after 100 iterations.

In the case $H/\lambda = 1/2$, all the diagonal coefficients became negative, and we observed a serious deterioration of the results.

The mean of the diagonal coefficients for the coarse matrix as a function of the ratio H/λ is given in Table 3.10.

H/λ	1/12	1/6	1/3	5/12	1/2
Mean of the diag. coeff.	4.56	4.07	2.08	0.9	-1.3

Table 3.10: Mean of the diagonal coefficients for the coarse matrix with the ratio H/λ.

3.8 Conclusion

These different tests allowed us to understand the principle of the coarse basis construction by energy minimisation. They also demonstrated the possibility, in very simple cases, to use well-structured matrices close to the original matrix and this gave satisfactory results. However, we also observe the limits of this approach for the Helmholtz problem.

We will extend this method by energy minimisation to the case of edge elements, also for a simple problem and a simple geometry.

Chapter 4

Extension to edge elements

Our aim, now, is to extend the algebraic multilevel method introduced in Chapter 3 for the nodal element case, to the edge elements. The following simple problem will be considered for numerical validation; we work on the unit square, and a very similar formulation is used on the unit cube:

$$\begin{cases} \operatorname{curl}\operatorname{curl}\mathbf{E} + \mathbf{E} = \mathbf{0} \text{ on } \Omega =]0;1[\times]0;1[, \\ \mathbf{E}\times\mathbf{n} = 1 \text{ on } \partial\Omega. \end{cases} \tag{4.1}$$

Let $\mathbf{E}_{\text{ess}}$ be a field satisfying the essential boundary conditions on $\partial\Omega$. The $\mathbf{E}$ field can be decomposed into: $\mathbf{E} = \mathbf{E}_0 + \mathbf{E}_{\text{ess}}$. The variational formulation can be written:

$$\begin{cases} \text{To find } \mathbf{E}_0 \in \mathbf{H}_0(\text{curl},\Omega) \text{ such that :} \\ a'(\mathbf{E}_0,\mathbf{E}') = -a'(\mathbf{E}_{\text{ess}},\mathbf{E}'),\ \forall \mathbf{E}' \in \mathbf{H}_0(\text{curl},\Omega), \\ \text{with } a'(\mathbf{E},\mathbf{E}') = \displaystyle\int_\Omega \operatorname{curl}\mathbf{E}\cdot\operatorname{curl}\mathbf{E}' + \int_\Omega \mathbf{E}\cdot\mathbf{E}'. \end{cases} \tag{4.2}$$

4.1 Principle of the construction

We want to compute two coarse bases viz. a nodal and a edge coarse basis, which minimise an appropriate energy to be described below, and satisfy a number of constraints among which the commutativity relation (2.8) plays a prominent rôle.

4.1.1 Notations and description of the basis constraints

Let us denote as follows the elements:

- $(\phi_i)_{i=\ldots d_h}$ is the fine nodal basis,
- $(\lambda_i)_{i=\ldots d'_h}$ is the fine edge basis,
- $(\Phi_i)_{i=\ldots d_H}$ is the coarse nodal basis,

- $(\Lambda_i)_{i=\ldots d'_H}$ is the coarse edge basis.

These coarse bases are constructed so as to satisfy the inclusion of finite element spaces, the "coarse" being included in the "fine", which is expressed by the following algebraic relations:

$$\begin{aligned}\Phi_i &= \sum_{j=1}^{d_h} \phi_j \alpha_{ji},\ \forall i = 1 \ldots d_H,\\ \Lambda_i &= \sum_{j=1}^{d'_h} \lambda_j \beta_{ji},\ \forall i = 1 \ldots d'_H.\end{aligned} \tag{4.3}$$

We already know a fundamental property linking the edge elements to the nodal elements: the gradient of a fine nodal element belongs to the space of fine edge elements; it can be recast as:

$$\operatorname{grad} \phi_i = \sum_{j=1}^{d'_h} \lambda_j a_{ji},\ \forall i = 1 \ldots d_h. \tag{4.4}$$

Our aim is to preserve this relation for the coarse bases, which is natural in the geometric multigrid context. Thus, a coherent representation of the kernel of the curl operator is preserved, which permits the efficient use of the smoothers proposed by Hiptmair and by Arnold et al.. Therefore the gradient of a coarse nodal element must be a combination of coarse edge elements; this condition is written algebraically as:

$$\operatorname{grad} \Phi_i = \sum_{j=1}^{d'_H} \Lambda_j A_{ji},\ \forall i = 1 \ldots d_H. \tag{4.5}$$

By using in (4.5) the prolongation operators α and β defined in (4.3) which contain the components of the coarse basis functions on the fine bases, we obtain:

$$\sum_{j=1}^{d_h} \operatorname{grad} \phi_j \alpha_{ji} = \sum_{j=1}^{d'_H} \sum_{k=1}^{d'_h} \lambda_k \beta_{kj} A_{ji},\ \forall i = 1 \ldots d_H. \tag{4.6}$$

Substituting (4.4) in (4.6), we eventually obtain:

$$\sum_{j=1}^{d_h} \sum_{k=1}^{d'_h} \lambda_k a_{kj} \alpha_{ji} = \sum_{j=1}^{d'_H} \sum_{k=1}^{d'_h} \lambda_k \beta_{kj} A_{ji},\ \forall i = 1 \ldots d_H. \tag{4.7}$$

Relation (4.7) is equivalent to:

$$a\alpha = \beta A. \tag{4.8}$$

and we will determine our coarse nodal and edge elements so as to satisfy (4.8).

We recall the dimensions of our matrices:

$$a \in \mathbb{R}^{d'_h \times d_h},\ A \in \mathbb{R}^{d'_H \times d_H},\ \alpha \in \mathbb{R}^{d_h \times d_H},\ \beta \in \mathbb{R}^{d'_h \times d'_H}. \tag{4.9}$$

The matrix a is known; more precisely, a is the node-edge incidence matrix on the fine mesh. It is a discrete analogue of the grad operator on this mesh. If a coarse mesh existed, A would be the discrete analogue of grad on this mesh. In the algebraic context, A is still the discrete analogue of the grad operator and should resemble a node-edge incidence matrix. However, there are many possible choices of A which will be studied below (Subsection 4.1.7). In a first approach, we tried to consider A as an unknown in the minimisation problem. This was a terrible idea, leading to difficult non-linear problems, whose solution was numerically inefficient.

4.1.2 Minimisation: global coupling

Ω is decomposed into overlapping subdomains $\Omega_i,\ \forall i = 1 \dots d_H$; this enables us to define the supports of the nodal coarse basis functions (as in Chapter 3). Denote by I_i the set of indices of the nodes which belong to interior of the subdomain Ω_i.

In the same way, Ω is decomposed into overlapping subdomains $\mathcal{U}_i,\ i = 1 \dots d'_H$ in order to localise the supports of the coarse edge basis functions. The subdomain $\mathcal{U}_i$ will usually be the intersection of two subdomains Ω_j and Ω_k.

Given the subdomains Ω_j, the choice of the matrix A determines the definition of the subdomains $\mathcal{U}_i$.

A natural extension of the Laplace equation case is:

$$\left\{\begin{array}{l} \text{To find } (\Phi_i)_{i=1..d_H},\ (\Lambda_i)_{i=1..d'_H} \text{ minimising} \\ \displaystyle\sum_{i=1}^{d_H} a(\Phi_i,\Phi_i) + \sum_{i=1}^{d'_H} b(\Lambda_i,\Lambda_i) \text{ under the constraints:} \\ \displaystyle\sum_{j=1}^{d_H} \Phi_j(x) = 1,\ \forall x \in \overline{\Omega} \text{ and } \operatorname{supp}(\Phi_i) \subset \Omega_i,\ \forall i = 1\dots d_H, \\ \displaystyle\operatorname{grad}\Phi_i = \sum_{j=1}^{d'_H} \Lambda_j A_{ji},\ \forall i = 1\dots d_H \text{ and } \operatorname{supp}(\Lambda_i) \subset \mathcal{U}_i,\ \forall i = 1\dots d'_H. \end{array}\right. \tag{4.10}$$

A priori, a is the bilinear form associated with the Laplace's equation, b is a bilinear form which can be a' but variants will be also implemented.

We introduce algebraic notations which encode the support constraints. The operators P_i, Q_i and R_i are canonical projection operators; P_i maps

$\mathbb{R}^{d_h}$ to $\mathbb{R}^{n_i}$ by keeping all the components indexed by I_i. Similarly, if I'_i denotes the indices of edges in $\mathcal{U}_i$, Q_i maps $\mathbb{R}^{d'_h}$ to $\mathbb{R}^{n'_i}$ by keeping all the components indexed by I'_i; here n'_i is the number of elements of I'_i written $|I'_i|$. Finally, denoting by I''_i an appropriate subset of indices of edges in Ω_i, to be described in Subsection 4.1.4, R_i maps $\mathbb{R}^{d'_h}$ to $\mathbb{R}^{n''_i}$, $n''_i = |I''_i|$ by keeping all the components indexed by I''_i.

$$\begin{aligned} P_i &: \mathbb{R}^{d_h} \to \mathbb{R}^{n_i},\ \forall i = 1 \dots d_H, \\ Q_i &: \mathbb{R}^{d'_h} \to \mathbb{R}^{n'_i},\ \forall i = 1 \dots d'_H, \\ R_i &: \mathbb{R}^{d'_h} \to \mathbb{R}^{n''_i},\ \forall i = 1 \dots d_H. \end{aligned} \tag{4.11}$$

In order to obtain a matrix form of the minimisation problem, we define more operators which are restrictions to the subdomains under consideration of the bilinear forms:

$$K_i = P_i^t K P_i \text{ and } K'_i = Q_i^t K' Q_i. \tag{4.12}$$

We also define:

$$\alpha_i = P_i \alpha_{\bullet i}, \quad \beta_i = Q_i \beta_{\bullet i} \tag{4.13}$$

We want now to minimise:

$$\sum_{i=1}^{d_H} \alpha_i^t K_i \alpha_i + \sum_{i=1}^{d'_H} \beta_i^t K'_i \beta_i \tag{4.14}$$

under the constraints:

$$\sum_{i=1}^{d_H} P_i^t \alpha_i = \gamma, \tag{4.15}$$

$$a P_i^t \alpha_i = \sum_{j=1}^{d'_H} Q_j^t \beta_j A_{ji},\ \forall i = 1 \dots d_H. \tag{4.16}$$

4.1.3 Method of resolution

In order to solve this problem, we introduce Lagrange multipliers: one vector $\mu \in \mathbb{R}^{d_h}$ and d_H vectors $\rho_i \in \mathbb{R}^{n''_i}$. Indeed, the constraint (4.15) must be applied to the d_h nodes of the initial mesh. The dimensions of the vectors ρ_i are determined from the choice of the subdomains $\mathcal{U}_j$ and Ω_i, and from the properties of the systems obtained; this process will be made clearer in the following.

The reader should be aware now that the matrices R_i have not yet been fully defined, and that their construction can be explicitly given and justified only after the following algebraic construction. We trust that the reader will be patient, and wait until Subsection 4.1.4 to get the end of the story.

Let us define column vectors α, β, ρ, γ and matrices B, Q, Q', C, D:

$$\alpha = \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_{d_H} \end{pmatrix}, \quad \beta = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_{d'_H} \end{pmatrix}, \quad \rho = \begin{pmatrix} \rho_1 \\ \vdots \\ \rho_{d_H} \end{pmatrix}, \quad \gamma = 1_{d_h \times 1}, \tag{4.17a}$$

$$B = \begin{pmatrix} P_1 \\ \vdots \\ P_{d_H} \end{pmatrix}, \tag{4.17b}$$

$$Q = \operatorname{diag}(K_i,\ i = 1 \dots d_H), \quad Q' = \operatorname{diag}(K'_i,\ i = 1 \dots d'_H), \tag{4.17c}$$

$$C = \operatorname{diag}(P_i,\ i = 1 \dots d_H)(I_{d_H} \otimes a^t) \operatorname{diag}(R_i^t, i = 1 \dots d_H), \tag{4.17d}$$

$$D = \operatorname{diag}(Q_i,\ i = 1 \dots d'_H)(A \otimes I_{d'_h}) \operatorname{diag}(R_i^t, i = 1 \dots d_H). \tag{4.17e}$$

Here, $\otimes$ denotes the tensor (or equivalently) Kronecker product under the convention $M \otimes N = (M_{ij}N)_{i,j}$. The notation $\operatorname{diag}(A_i,\ i = 1 \dots n)$ refers to a block-diagonal matrix, whose diagonal blocks are the A_i's, $i = 1 \dots n$.

The minimisation problem can now be written:

$$\begin{cases} \text{To find a critical point } (\alpha_c, \beta_c, \mu_c, \rho_c) \text{ of the Lagrangian } \mathcal{L} \text{ defined by:} \\ \mathcal{L}(\alpha, \beta, \mu, \rho) = \dfrac{1}{2}\alpha^t Q \alpha + \dfrac{1}{2}\beta^t Q' \beta + \mu^t (B^t \alpha - \gamma) + \rho^t (C^t \alpha - D^t \beta). \end{cases} \tag{4.18}$$

This critical point must verify the following system of equations:

$$Q\alpha_c = -B\mu_c - C\rho_c, \tag{4.19a}$$

$$Q'\beta_c = D\rho_c, \tag{4.19b}$$

$$B^t \alpha_c = \gamma, \tag{4.19c}$$

$$C^t \alpha_c = D^t \beta_c. \tag{4.19d}$$

As for the nodal element case, this linear system can be solved in the following way:

- first, the vectors μ_c and ρ_c of Lagrange multipliers are determined applying an iterative method to the system:

$$B^t Q^{-1} B\mu + B^t Q^{-1} C\rho = -\gamma, \tag{4.20a}$$

$$C^t Q^{-1} B\mu + [C^t Q^{-1} C + D^t Q'^{-1} D]\rho = 0. \tag{4.20b}$$

  Relation (4.20a) is obtained by solving (4.19a) for α_c and substituting into (4.19c); relation (4.20b) is obtained by solving (4.19b) for β_c and substituting this value and the value of α_c into (4.19d). At this point, we see that (4.20) is symmetric, we do not know yet that is positive definite; the construction of the R_i's will precisely ensure this property.

- then, we return to the computation of α_c and β_c by solving:

$$\begin{cases} Q\alpha = -B\mu_c - C\rho_c, \\ Q'\beta = D\rho_c. \end{cases} \tag{4.21}$$

The number of Lagrange multipliers is $d_h + \sum_{i=1}^{d_H} n''_i$.

The dominant part is the computation of the Lagrange multipliers. The second stage *i.e.* the computation of α_c and β_c is simpler to implement. The matrix of the system:

$$M = \begin{pmatrix} B^tQ^{-1}B & B^tQ^{-1}C \\ C^tQ^{-1}B & C^tQ^{-1}C + D^tQ'^{-1}D \end{pmatrix}, \tag{4.22}$$

is symmetric, as we have observed above.

The first numerical tests demonstrated that our systems contained a set of linearly dependent equations. We tried therefore to understand the origin of this rank defect and for this purpose, we studied the properties of the bilinear form:

$$\begin{pmatrix} \mu \\ \rho \end{pmatrix}^t M \begin{pmatrix} \mu \\ \rho \end{pmatrix} = (B\mu + C\rho)^tQ^{-1}(B\mu + C\rho) + (D\rho)^tQ'^{-1}(D\rho). \tag{4.23}$$

The blocks K_i^{-1}, $\forall i = 1 \dots d_H$ and $K_i'^{-1}$, $\forall i = 1 \dots d'_H$ are symmetric definite positive, and so are the matrices Q^{-1} and Q'^{-1} defined by relation (4.17c). The above bilinear form is consequently positive. Moreover, consider:

$$(B\mu + C\rho)^tQ^{-1}(B\mu + C\rho) + (D\rho)^tQ'^{-1}(D\rho) = 0. \tag{4.24}$$

Since Q and Q' are symmetric positive definite, any solution of (4.24) satisfies:

$$\begin{cases} D\rho = 0, \\ B\mu + C\rho = 0. \end{cases} \tag{4.25}$$

A sufficient condition for our bilinear form (4.23) to be definite, is that D be injective; since we already know that B is injective.

4.1.4 Construction of the index sets I''_j and of the projections R_j

Let us look for a sufficient condition implying that D is injective. It is now that the choice of the index set I''_j and therefore the definitions of the projections R_j can be explicitly defined. The operator R_i maps n''_i vectors from the canonical basis of $\mathbb{R}^{d'_h}$ to the vectors of the basis of $\mathbb{R}^{n''_i}$. For all $i = 1 \dots d_H$, these vectors are called $(e_l)_{l \in I''_i}$. The $(R_ie_l)_{l \in I''_i}$ make up a basis of $\mathrm{Im}(R_i)$ and we observe that $R_i^tR_ie_l = e_l$. We define also:

$$\forall k = 1 \dots d'_H,\ L_k = \{i,\ A_{ik} \neq 0\},\ \forall i = 1 \dots d_H,\ J_i = \{k,\ A_{ik} \neq 0\}. \tag{4.26}$$

We consider:

$$D\rho = 0, \tag{4.27}$$

or equivalently:

$$\sum_{i=1}^{d_H} Q_k A_{ik} R_i^t \rho_i = 0,\ \forall k = 1 \dots d'_H, \tag{4.28}$$

and substituting ρ_i by its decomposition on the basis of $(R_i e_l)_{l \in I''_i}$:

$$\sum_{i=1}^{d_H} Q_k A_{ik} R_i^t \left(\sum_{l \in I''_i} \gamma_l^i R_i e_l \right) = 0,\ \forall k = 1 \dots d'_H, \tag{4.29}$$

this finally gives:

$$\sum_{i \in L_k} \sum_{l \in I''_i} \gamma_l^i A_{ik} Q_k e_l = 0,\ \forall k = 1 \dots d'_H. \tag{4.30}$$

We choose a particular $i \in L_k$ and a particular $l \in I''_i$ and we are interested in the coefficient γ_l^i. Let us suppose that:

$$\forall k \in J_i,\ Q_k e_l = 0 \tag{4.31}$$

then D cannot be injective because in that case γ_l^i could take any value and there would exist a non-zero vector ρ such that $D\rho = 0$. If this situation is avoided, one necessarily has:

$$\{e_l,\ l \in I''_i\} = \mathrm{Im}(R_i^t) \subset \cup\{\mathrm{Im}(Q_k^t),\ k \in J_i\}. \tag{4.32}$$

This should help us to define the support of the multiplier vectors ρ_i. Let us suppose that condition (4.32) is verified. Then, in particular, there exists a k in J_i satisfying $Q_k e_l \neq 0$. Let E be the set $\{k \in J_i,\ Q_k e_l \neq 0\}$, which of course depends on l and i.

Let us suppose that there is a k in E such that e_l is not in $\mathrm{Im}(R_j^t)$, for all j in $L_k \setminus \{i\}$. If we decompose (4.30) by isolating the term $Q_k e_l$, we obtain:

$$\sum_{j \in L_k - i} \sum_{n \in I''_j} \gamma_n^j A_{jk} Q_k e_n + \gamma_l^i A_{il} Q_k e_l + \sum_{n \neq l, n \in I''_i} \gamma_n^i A_{ik} Q_k e_n = 0. \tag{4.33}$$

The term $Q_k e_l$ will be independent of anything else and we will have necessarily $\gamma_l^i = 0$. In that case, the matrix D is injective.

Consider the case where $|J_i|$ is strictly larger 1. In contrast with the previous situation, suppose that:

$$\forall k \in E,\ e_l \in \mathrm{Im}(R_m^t),\ \forall m \in L_k \setminus \{i\};$$

then, we can define a non-zero vector in the kernel of D, once k_1 and k_2 have been found in J_i. It is given by:

$$\begin{pmatrix} 0 & \frac{1}{A_{ik_1}} R_{k_1} e_l & \dots & \frac{-1}{A_{ik_2}} R_{k_2} e_l & 0 \end{pmatrix}^t . \tag{4.34}$$

Consequently, the condition:

$$\exists k \in E \text{ such that } e_l \notin \mathrm{Im}(R_m^t),\ \forall m \in L_k \setminus \{i\}, \tag{4.35}$$

is required in order for D to be injective with $|J_i|> 1$ for all i — but here this is satisfied; it is also sufficient until since we postulate (4.32). Condition (4.35) is eventually sufficient for M to be definite positive.

In order to determine n_i'' constraints, the following process is used:

1. We start from the trial operator $\tilde{R}_i$ such that: $\mathrm{Im}(\tilde{R}_i^t) = \cup\{\mathrm{Im}(Q_k^t),\ k \in J_i\}$,
2. We decrease the dimension of the spaces $\mathrm{Im}(\tilde{R}_i^t)$ in order to verify the injectivity condition: this leads us to remove one unknown per edge where we had initially enforced a constraint.

From our trial operators where all the edges are constrained, we remove d_h' constraints to obtain a full-rank matrix.

This yields a symmetric positive definite matrix and allows us to use the conjugate gradient method. In the following, we give an appropriate algorithm for computing the matrix-vector product.

4.1.5 Multiplication algorithm

In order to compute $Mv = w$, the multiplication can be seen as composed of two distinct parts: $Mv = M_{\text{nod}}v + M_{\text{edg}}v$.

The same block decomposition is kept for w as for v: $w = \begin{pmatrix} \tilde{\mu} \\ \tilde{\rho} \end{pmatrix}$.

1. For $i = 1 \dots d_H$, compute: $u_i = R_i^t \rho_i$,
2. Computation of $M_{\text{nod}}v = w$:
   - For $i = 1 \dots d_H$, compute: $b_i = P_i(\mu + a^t u_i)$,
   - For $i = 1 \dots d_H$, solve: $K_i x_i = b_i$,
   - $\tilde{\mu} = \sum_{i=1}^{d_H} P_i^t x_i$,
   - For $i = 1 \dots d_H$, $\tilde{\rho}_i = R_i a P_i^t x_i$.
3. Computation of $M_{\text{edg}}v = w$:
   - For $i = 1 \dots d_H'$, compute: $b_i = \sum_{i=1}^{d_H} Q_i A_{ij} u_j$,

- For $i = 1 \dots d'_H$, solve: $K'_i x_i = b_i$,
- $\tilde{\mu} = 0_{d_h,1}$,
- For $i = 1 \dots d_H$, $\tilde{\rho}_i = R_i(\sum_{j=1}^{d'_H} P_j^t A_{ji} x_j)$.

4.1.6 Decoupling edge and nodal minimisations

The same decomposition into subdomains is made as in the global coupling case. However, this time, the two minimisation problems under constraints are decoupled and solved successively. First:

$$\begin{cases} \text{To find } (\Phi_i)_{i=1..d_H} \text{ minimising } \displaystyle\sum_{i=1}^{d_H} a(\Phi_i, \Phi_i) \text{ under the constraints:} \\ \displaystyle\sum_{j=1}^{d_H} \Phi_j(x) = 1,\ \forall x \in \overline{\Omega} \text{ and } \operatorname{supp}(\Phi_i) \subset \Omega_i,\ \forall i = 1 \dots d_H. \end{cases} \tag{4.36}$$

Next, the basis $(\Phi_i)_i$ is computed and can be used to solve:

$$\begin{cases} \text{To find } (\Lambda_i)_{i=1..d'_H} \text{ minimising } \displaystyle\sum_{i=1}^{d'_H} b(\Lambda_i, \Lambda_i) \text{ under the constraints:} \\ \operatorname{grad} \Phi_i = \displaystyle\sum_{j=1}^{d'_H} \Lambda_j A_{ji},\ \forall i = 1 \dots d_H \text{ and } \operatorname{supp}(\Lambda_i) \subset \mathcal{U}_i,\ \forall i = 1 \dots d'_H. \end{cases} \tag{4.37}$$

The nodal minimisation problem can be solved using the techniques introduced in Chapter 3. We must take care of the edge problem, which is formulated algebraically as:

$$\begin{cases} \text{To minimise } \displaystyle\sum_{i=1}^{d'_H} \beta_i^t K'_i \beta_i \text{ under the constraints:} \\ a P_i^t \alpha_i = \displaystyle\sum_{j=1}^{d'_H} Q_j^t \beta_j A_{ji},\ \forall i = 1 \dots d_H. \end{cases} \tag{4.38}$$

Finding the critical point of the Lagrangian that can be associated with (4.38) leads to solving the linear system:

$$\begin{cases} Q' \beta_c = -D \rho_c, \\ D^t \rho_c = C^t \alpha_{\text{known}}. \end{cases} \tag{4.39}$$

We observe that the injectivity of D is necessary and sufficient for the existence and uniqueness of a solution of (4.39).

The dominant part is again the implementation of the matrix-vector product and here, we are inspired by the algorithm introduced at the end of Subsection 4.1.2, for the global coupling.

This algorithm is simpler than the global coupling and gives us some freedom in the computation of the coarse basis. This allows us to use a coarse nodal basis computed in another way than the one described in Chapter 3. This method is relatively close to that used by the Sandia team (Bochev et al., Chapter 2).

4.1.7 Choice of the coarse node-edge incidence matrix

We want to identify the features of the operator A. We look for a simple definition for A. We must also check that the support of the $(\Lambda_i)_{i=1\dots d'_H}$ is not too large; the drawback would be a large overlap, and then it would be difficult to control the fill-in of the operators.

To make the explanations simple, only a case with 4 sub-domains is studied here.

Let Ω_i be the support of the coarse nodal function Φ_i; we define:

$$\begin{aligned}
\mathcal{U}_{ij} &= \Omega_i \cap \Omega_j \backslash \bigcup_{k \neq i,j} \Omega_k, \\
\mathcal{U}_{ijk} &= \Omega_i \cap \Omega_j \cap \Omega_k \backslash \Omega_l \ (l \neq i,j,k), \\
\mathcal{U}_{1234} &= \bigcap_{i=1}^{4} \Omega_i.
\end{aligned}$$

We would like to have $d'_H = 4$ with

$$\operatorname{supp}(\Lambda_i) = \Omega_{\text{startnod}(i)} \cap \Omega_{\text{endnod}(i)},$$

where startnod(i) and endnod(i) refers respectively to the initial and the final node of the edge, *i.e.* we would like to imitate the incidence graph given in Fig. 4.1:

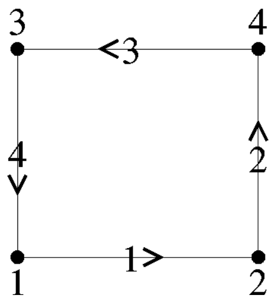


Figure 4.1: Coarse node-edge incidence graph

The supports must be chosen so as to verify the constraint:

$$\sum_{j=1}^{4} \operatorname{grad} \Phi_j = 0. \tag{4.40}$$

First case: $\mathcal{U}_{14} = \emptyset,\ \mathcal{U}_{23} = \emptyset.$

Concentrating on $\mathcal{U}_{12}$, we observe: $\operatorname{grad} \Phi_1 + \operatorname{grad} \Phi_2 = 0$. Under the conditions on the support of $(\Lambda_i)_i$, a condition on the coefficients is: $A_{11} = A_{12}$.

In the same way we obtain:

- on $\mathcal{U}_{24}$, $A_{22} = A_{24}$,
- on $\mathcal{U}_{34}$, $A_{33} = A_{34}$,
- on $\mathcal{U}_{24}$, $A_{22} = A_{24}$,
- on $\mathcal{U}_{123}$, $A_{11} + A_{12} + A_{13} = 0$, and $A_{21} + A_{22} + A_{23} = 0$,
- on $\mathcal{U}_{234}$, $A_{22} + A_{23} + A_{24} = 0$, and $A_{32} + A_{33} + A_{34} = 0$,
- on $\mathcal{U}_{431}$, $A_{31} + A_{33} + A_{34} = 0$, and $A_{41} + A_{43} + A_{44} = 0$,
- on $\mathcal{U}_{1234}$, $\sum_{j=1}^{4} A_{ij} = 0,\ \forall i = 1 \ldots 4.$

The simplest solution for the operator A to satisfy these equalities is to choose:

$$A = \begin{pmatrix} A_{11} & -A_{11} & 0 & 0 \\ 0 & A_{22} & 0 & -A_{22} \\ 0 & 0 & A_{33} & -A_{33} \\ A_{41} & 0 & A_{41} & 0 \end{pmatrix}.$$

We must choose a non-zero value to the coefficients of A, otherwise the minimisation problem will give a zero solution. The simplest choice is the value which gives an A coinciding with the incidence matrix of the graph in Fig. 4.1 *i.e.*:

$$A_{11} = A_{22} = -A_{33} = -A_{41} = -1.$$

This gives:

$$A = \begin{pmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & -1 \\ 1 & 0 & -1 & 0 \end{pmatrix}.$$

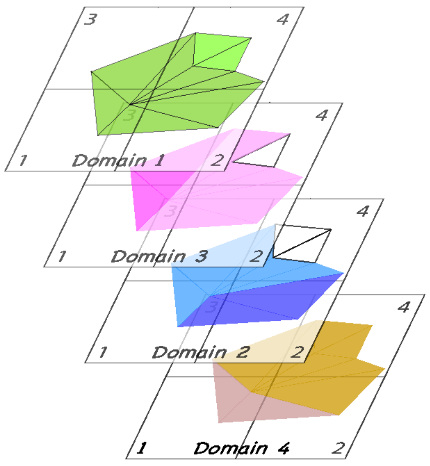


Figure 4.2: The triangles close to the intersection of four subdomains are represented. The elements belonging to each subdomain are coloured. The subdomains before overlapping appeared with darker colours. The white triangle in Domain 3 belongs to $\mathcal{U}_{14}$.

Second case: $\mathcal{U}_{14} \neq \emptyset$, or $\mathcal{U}_{23} \neq \emptyset$. Some additional difficulties can be encountered if $\mathcal{U}_{14}$ or $\mathcal{U}_{23}$ are not empty. These different cases are possible as demonstrated in Fig. 4.2.

If $\mathcal{U}_{14}$ or $\mathcal{U}_{23}$ is not included in the support of any Λ_j, then it is possible to obtain $\operatorname{grad}\Phi_i \neq 0$ ($i = 1, 4$ or $2, 3$ following the context) and necessarily: $\operatorname{span}(\operatorname{grad}\Phi_i,\ i = 1\dots4) \not\subset \operatorname{span}(\Lambda_i,\ i = 1\dots4)$.

Several ideas should enable us to remedy this difficulty:

- enlarge the supports of the functions Λ_i,
- create additional edges,
- modify the domains in order to avoid this situation.

Using 6 edges instead of 4 and keeping the same definition of supports, enables us to remedy the problem; the incidence graph in Fig. 4.3 illustrates this proposition:

$$A = \begin{pmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & -1 \\ 1 & 0 & -1 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & -1 & 1 & 0 \end{pmatrix}.$$

The drawback is that we created new edges, in particular, there will be 8 edges out of each node, we could imagine that we would obtain 26 edges in

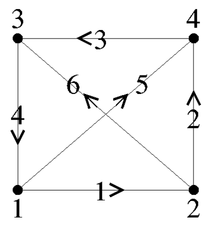


Figure 4.3: Coarse node-edge incidence graph with 6 edges.

3D. Moreover with the wide overlap of the subdomains, a huge optimisation problem could be obtained.

We can also try to modify the domains in order to remove these parasitic phenomena, for instance to enlarge Ω_3 and Ω_2 to absorb the possible difficulties related to $\Omega_1 \cap \Omega_4 \neq \emptyset$.

With the overlap we use, we are currently always in the case 4.1, without any difficulties. With narrower overlaps, however, the problem becomes systematic

Choosing 6 edges as in Fig. 4.3 coincide with the choice of Reitzinger and Schöberl, preserved by the Sandia team, Bochev et al.; this choice is currently used for this class of methods.

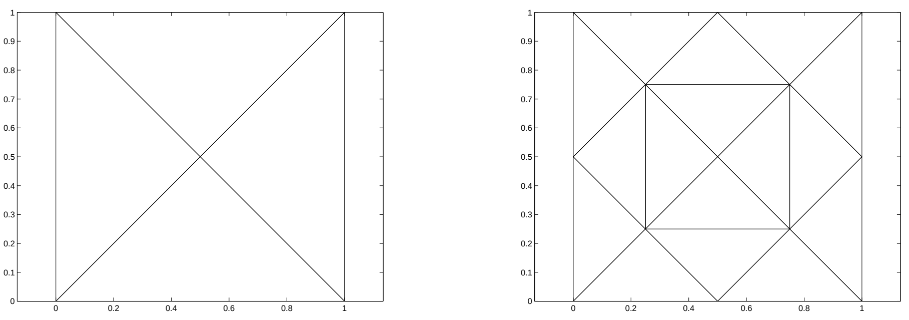
4.2 Numerical results in 2D

In the following tables, different bilinear forms are used for the edge part of the minimisation. For convenience, we define a few abbreviations used in the tables:

- M refers to the norm defined by the mass matrix on edge elements,
- K' refers to the energy norm associated with the matrix K' of the problem in edge elements,
- $K' + aM_\phi^{-1}a^t$ refers to the norm defined by this matrix, M_ϕ being the mass matrix on nodal elements, with mass lumping; hence M_ϕ is diagonal.
- γ denotes the empirical coefficient 0.1/number of coarse edges.
- A_i is the operator on level i, N_i is the number of unknowns at this level and nnz(A_i) the number of non-zero elements in the matrix A_i; our convention is that level 1 is the finest.

4.2.1 Structured meshes

We begin with a very simple mesh on the unit square which is regularly refined (see Fig. 4.2.1). The decomposition of the domain and the choice of the coarse variables is made as if we wanted to use a geometric multigrid method; the only change is the computation of the prolongation operator, which does not take into account the geometrical information, but uses the previously introduced algebraic method.



(a) (b)

Figure 4.4: Initial mesh (4.4(a)) and first refinement (4.4(b)).

Dimensions

The numbers of Lagrange multipliers coming (1) from the constraints (4.15) associated with the vector μ, and (2) from the constraints (4.16) associated with the vectors $(\rho_i)_{i=1\ldots d_H}$ are given separately; they are presented under the form:

$$\text{dimension of } \mu + \text{sum of the dimensions of } (\rho_i).$$

in Table 4.1.

	1st refinement	2nd	3rd	4th
Nb of multipliers	13+40	41+152	145+592	545+2336
Nb of unknowns	20	88	368	1504

Table 4.1: Number (Nb) of Lagrange multipliers and unknowns — structured meshes.

Computation of Lagrange multipliers

We start with vanishing multipliers. Whatever the approach (coupled or decoupled minimisation), the conjugate gradient without preconditioning is used to compute the Lagrange multipliers. The number of iterations required to divide the residual by 10^3 during the iterative process is reported in Table 4.2 for the coupled minimisation approach. These numbers are given for different choices of norms in the edge case. In the first column, in parentheses, the condition number is given; it is only given for the smallest mesh.

Norm	1st refinement	2nd	3rd	4th
M	18 (52.3)	28	30	30
K'	35 (541.4)	108	239	333
$\gamma(K' + aM_\phi^{-1}a^t)$	15 (36.5)	25	27	27

Table 4.2: Convergence of the multipliers computation (division of the residual by 10^3) — coupled minimisation, structured meshes.

In Table 4.3, results are also given for different choices of norms. This time however, the number of iterations needed to compute the Lagrange multipliers associated with the vector μ on one hand and to the vectors (ρ_i) on the other hand are clearly separated. Indeed, in the decoupled case, μ and (ρ_i) are computed separately. The conjugate gradient method with the same stopping criterion, division of the residual by 10^3, is used.

Norm	1st refinement	2nd	3rd	4th
M	3+6	6+9	8+9	8+9
K'	3+5	6+43	8+64	8+79
$K' + aM_\phi^{-1}a^t$	3+4	6+9	8+9	8+8

Table 4.3: Convergence of the multipliers computation (division of the residual by 10^3) — decoupled minimisation, structured meshes.

Graphical representations of computed coarse basis functions are given in Fig. 4.5.

Convergence for the two-level method

The system is solved by using a preconditioned conjugate gradient algorithm. The preconditioner is a two-level method which uses one pre- and one postsmoothing step by Arnold's algorithm (Chapter 2); on the coarse grid, a direct solver is used. The conjugate gradient method stops when the initial residual has been divided by 10^{10}. Table 4.4 gives the results in the case of the coupled minimisation. The results obtained by using bilinear interpolation for the prolongation operator, as in classical multigrid, are also reported for comparison with a method whose behaviour is known. The corresponding mention in the norm column is "geometric".

Table 4.5 gives the results in the decoupled case. Apart from a not very relevant test, the results are identical.

Statistics for the two-level method

Independently of the method chosen to compute the prolongation operator, the parameters in Table 4.6 enable us to qualify, in part, the efficiency of the two-level method.

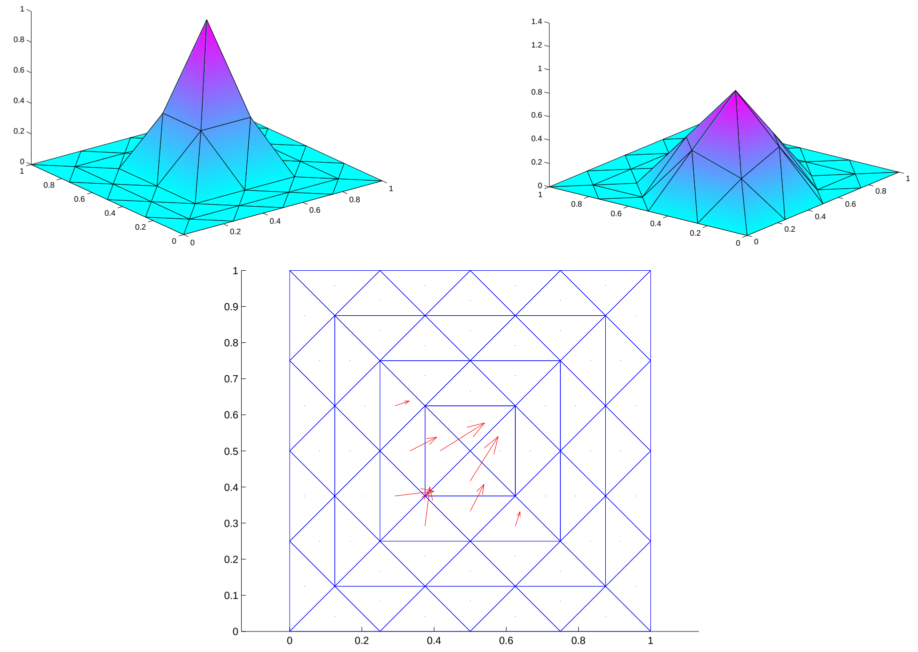


Figure 4.5: Coarse basis functions: on top two coarse nodal functions and on bottom the coarse edge function linking them — structured mesh.

Norm	1st refinement	2nd	3rd	4th
M	5	7	10	11
K'	5	6	6	7
$\gamma(K' + aM_\phi^{-1}a^t)$	4	6	6	7
geometric	5	6	6	7

Table 4.4: Convergence of the conjugate gradient preconditioned by a two-level method (division of the residual by 10^{10}) — coupled minimisation, structured meshes.

Norm	1st refinement	2nd	3rd	4th
M	5	7	10	11
K'	5	6	6	7
$K' + aM_\phi^{-1}a^t$	5	6	6	7

Table 4.5: Convergence of the conjugate gradient preconditioned by a two-level method (division of the residual by 10^{10}) — decoupled minimisation, structured meshes.

First, the "grid complexity" (GC) parameter is the ratio of the sum of the unknowns on every level to the number of unknowns on the finest level *i.e.*:

$$\mathrm{GC} = \frac{\sum_{i=1}^{\mathrm{niv}} N_i}{N_1} \tag{4.41}$$

where niv refers to the number of levels. The GC parameter enables us to check that the number of unknowns per level decrease fast enough; classically we expect N_i/N_1 to decrease geometrically for increasing i.

The operator complexity (OC) parameter is the ratio of the sum of the number of non-zero elements on every level to the number of non-zero elements on the finest level *i.e.*:

$$\mathrm{OC} = \frac{\sum_{i=1}^{\mathrm{niv}} \mathrm{nnz}(A_i)}{\mathrm{nnz}(A_1)} \tag{4.42}$$

This parameter enables us to check that the number of non-zero elements in the matrices is proportional to the number of unknowns which means the sparsity is approximately independent of the level; this aspect influences the memory cost and also the computational cost of the matrix-vector product.

Parameters	1st	2nd	3rd	4th
grid complexity	1.2	1.23	1.24	1.25
operator complexity	1.15	1.21	1.23	1.24

Table 4.6: Some statistics for the two-level method — structured meshes.

4.2.2 Unstructured meshes

An mesh generator is used to produce an unstructured mesh without particular specification. The only parameter we set, is the maximal diameter $h_{\max}$ of the elements (see Fig. 4.6 for $h_{\max} = 0.1$). The decomposition into subdomains is performed according to the method described in Subsection 3.3 with a narrower strip close to the boundary.

Dimensions

The number of Lagrange multipliers is given in Table 4.7. In parentheses, the number of unknowns on the coarse grids is also reported.

Computation of Lagrange multipliers

The rules for computation are the same as for structured meshes. The results for the coupled minimisation are given in Table 4.8.

The results for the decoupled minimisation are given in Table 4.9.

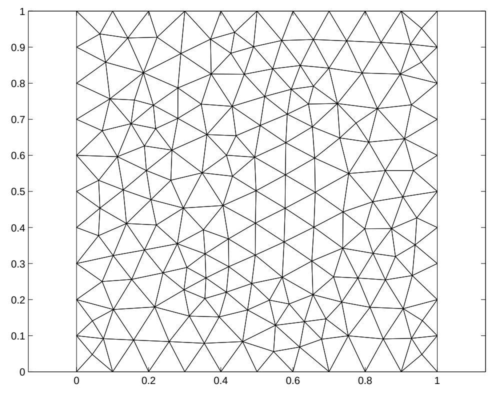


Figure 4.6: Unstructured mesh with $h_{\max} = 0.1$.

	$h_{\max} = 0.2$	0.1	0.05	0.025
Nb of multipliers	52+272	182+1039	730+4926	2805+16946
Nb of unknowns	113	463	2027	8092
(coarse)	(12)	(40)	(144)	(544)

Table 4.7: Number (Nb) of Lagrange multipliers and unknowns — unstructured meshes.

Norm	$h_{\max} = 0.2$	0.1	0.05	0.025
M	25 (84.5)	30	34	33
K'	198 (1110)	488	898	X
$\gamma(K' + aM_\phi^{-1}a^t)$	30 (131.4)	40	48	48

Table 4.8: Convergence of the multipliers computation (division of the residual by 10^3). X : the convergence criterion is not reached after 1000 iterations — coupled minimisation, unstructured meshes.

Norm	$h_{\max} = 0.2$	0.1	0.05	0.025
M	10+18	13+18	14+20	14+21
K'	10+138	13+353	14+646	14+889
$K' + aM_\phi^{-1}a^t$	10+23	13+30	14+32	14+32

Table 4.9: Convergence of the multipliers computation (division of the residual by 10^3) — decoupled minimisation, unstructured meshes.

For coupled or decoupled minimisation, the energy-norm gives unsatisfactory results in this 2D case. The evolution of the number of iterations in both formulations is comparable.

Convergence for a two-level method

Preconditioned conjugate gradient For the preconditioned conjugate gradient, the parameters are the same as for structured meshes. The results for the coupled minimisation are given in Table 4.10.

Norm	$h_{\max} = 0.2$	0.1	0.05	0.025
M	7	11	15	17
K'	7	10	12	13
$\gamma(K' + aM_\phi^{-1}a^t)$	7	10	12	13

Table 4.10: Convergence of the conjugate gradient preconditioned by a two-level method (division of the residual by 10^{10}) — coupled minimisation, unstructured meshes.

Table 4.11 gathers the results for the decoupled minimisation. The results are almost the same in both cases.

Norm	$h_{\max} = 0.2$	0.1	0.05	0.025
M	8	11	15	17
K'	7	10	12	13
$K' + aM_\phi^{-1}a^t$	8	10	13	14

Table 4.11: Convergence of the conjugate gradient preconditioned by a two-level method (division of the residual by 10^{10}) — decoupled minimisation, unstructured meshes.

Simple two-level method The number of iterations for the two-level method without Krylov accelerator are given. The stopping criterion is the same as for the preconditioned conjugate gradient (division of the residual by 10^{10}). Table 4.12 gathers the results for the coupled minimisation.

Norm	$h_{\max} = 0.2$	0.1	0.05	0.025
M	11	22	38	X
K'	11	17	X	X
$\gamma(K' + aM_\phi^{-1}a^t)$	11	17	27	X

Table 4.12: Convergence of the two-level method (division of the residual by 10^{10}). X : unsuccessful computation — coupled minimisation, unstructured meshes.

The table 4.13 gathers the results for the decoupled minimisation.

Norm	$h_{\max} = 0.2$	0.1	0.05	0.025
M	12	22	39	51
K'	11	17	25	27
$\gamma(K' + aM_\phi^{-1}a^t)$	11	18	29	34

Table 4.13: Convergence of the two-level method (division of the residual by 10^{10}) — decoupled minimisation, unstructured meshes.

The two-level solver is not optimal but the progression of the number of iterations is not dramatically unfavourable.

Graphical representations of computed coarse basis functions are given in Fig. 4.7.

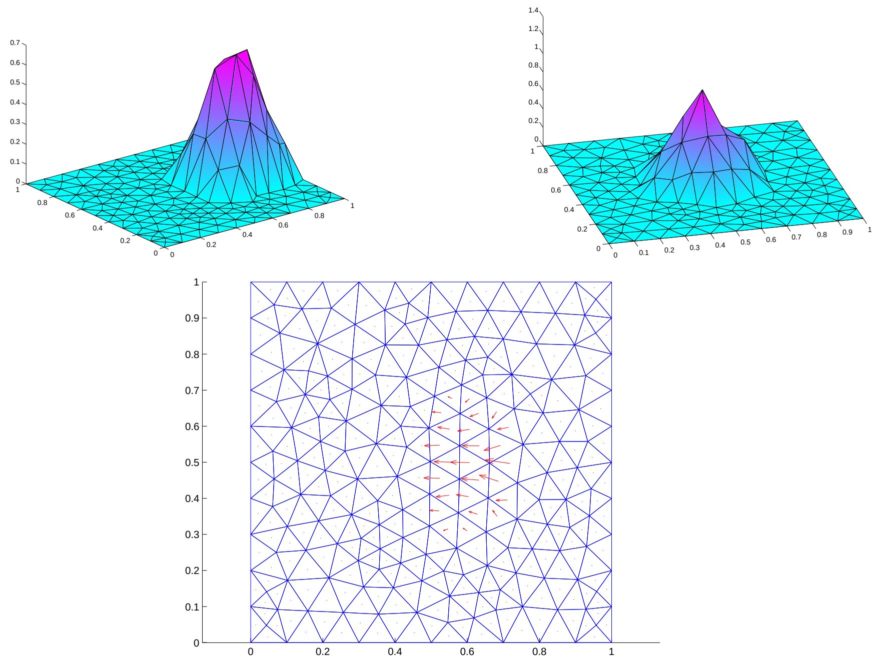


Figure 4.7: Coarse basis functions: on top two coarse nodal functions and on the bottom the coarse edge function linking them — unstructured mesh.

Statistics for the two-level method

Table 4.14 gathers the same statistics as for the structured meshes, in Table 4.6.

Parameters	$h_{\max} = 0.2$	0.1	0.05	0.025
grid complexity	1.1	1.09	1.07	1.07
operator complexity	1.13	1.12	1.1	1.1

Table 4.14: Some statistics for the two-level methods — unstructured meshes.

How to decrease overlap

It is also possible to decrease overlap which lets us decrease significantly the number of unknowns in the computation of Lagrange multipliers. However, some difficulties appear:

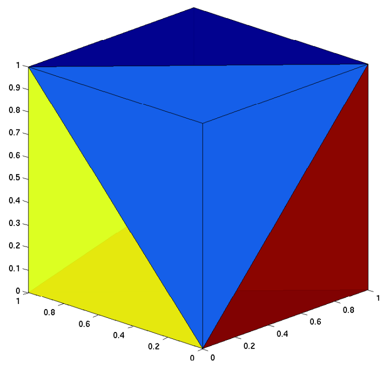
- many edges do not contribute to the interpolation;
- the second case described in Subsection 4.1.7 creates many difficulties.

4.3 3D numerical results

4.3.1 Structured meshes

The unit cube is decomposed into 6 tetrahedra that will be regularly refined (see Fig. 4.8).

The decomposition of the domain and the choice of the coarse variables is performed as if we wanted to use a geometric multigrid method; the only change is, as in 2D, the computation of the prolongation operator.


(a)

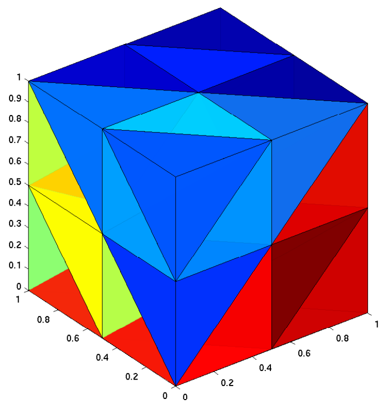

(b)

Figure 4.8: Initial mesh (4.8(a)) and first refinement (4.8(b))

Dimensions

Data are gathered in Table 4.15. It contains the same information as in 2D (see Table 4.1), in particular, the two kinds of multipliers are distinguished.

	1st	2nd	3rd	4th
Nb of multipliers	27+164	125+1060	729+7544	4913+56752
Nb of unknowns	26 (1)	316	3032	26416

Table 4.15: Number (Nb) of Lagrange multipliers and unknowns — structured meshes.

Computation of Lagrange multipliers

The multipliers are computed with the same parameters as in 2D. We only use the decoupled minimisation, which gave the best results in 2D; some tests, which are not reported here, demonstrate that the coupled minimisation is not very efficient in 3D. The results are given in Table 4.16.

Norm	1st	2nd	3rd	4th
M	9+18	14+24	18+33	22+35
K'	9+25	14+32	18+47	22+60
$K' + aM_\phi^{-1}a^t$	9+42	14+94	18+189	X

Table 4.16: Convergence of the multipliers computation (division of the residual by 10^3). X : unsuccessful computation — structured meshes.

Convergence for the two-level method

As in 2D, some tests are performed with the preconditioned conjugate gradient method and the results are gathered in Table 4.17. The comparison with the trilinear interpolation, used in classical geometric multigrid, is also given.

Norm	1st	2nd	3rd	4th
M	4	7	10	X
K'	4	7	10	11
$K' + aM_\phi^{-1}a^t$	4	7	10	X
geometric	4	7	10	X

Table 4.17: Convergence of the conjugate gradient preconditioned by a two-level method (division of the residual by 10^{10}). X : unsuccessful computation — structured meshes.

The results obtained resemble those obtained with the standard geometric multigrid method.

Statistics for the two-level method

The statistics have the same meaning as in 2D (see Table 4.6). The results are gathered in Table 4.18.

Parameters	1st	2nd	3rd	4th
grid complexity	1.04	1.08	1.10	1.12
operator complexity	1.00	1.05	1.09	1.11

Table 4.18: Some statistics for the two-level method — structured meshes.

4.3.2 Unstructured meshes

As in the 2D case, an mesh generator is used to produce unstructured meshes. The only parameter we set, is the maximal diameter $h_{\max}$ of the elements; see Fig. 4.9 for $h_{\max} = 0.4$).

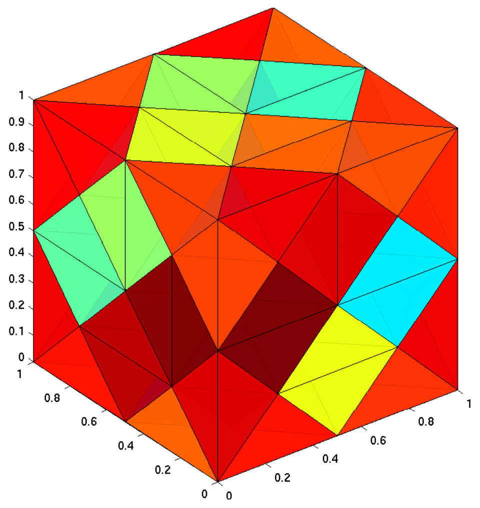


Figure 4.9: Unstructured mesh for $h_{\max} = 0.4$.

Dimensions

Data are given in Table 4.19, which contains the same data as in the 2D unstructured case (see Table 4.7); in particular, the two kinds of multipliers are distinguished and the numbers of coarse unknowns is written in parentheses.

	$h_{\max} = 0.4$	0.2	0.1
Nb of multipliers	63+1100	271+6997	1718+51847
Nb of unknowns	158	1055	8994
(coarse)	(12)	(54)	(300)

Table 4.19: Number (Nb) of Lagrange multipliers and unknowns — unstructured meshes.

Computation of Lagrange multipliers

The computation of the multipliers is performed with the same parameters as in the 3D structured case. We also use the decoupled minimisation. We only kept the norms defined by M and K'; indeed, the case of the 3D structured mesh lets us think that the regularised form was not as interesting as in 2D. The results are given in the Table 4.20.

Norm	$h_{\max} = 0.4$	0.2	0.1
M	8+42	13+66	15+X
K'	8+56	13+106	15+X

Table 4.20: Convergence of the multipliers computation (division of the residual by 10^3). X: the convergence criterion is not reached after 1000 iterations — unstructured meshes.

The increase in the number of iterations for computing the Lagrange multipliers associated with the commutativity relation is dramatic.

Convergence for the two-level method

As in 2D or 3D with structured meshes, the preconditioned conjugate gradient is used and the results are given in Table 4.21. Here, the computation is performed using the unconverged bases previously obtained; see Table 4.20.

Norm	$h_{\max} = 0.4$	0.2	0.1
M	5	8	8
K'	5	8	8

Table 4.21: Convergence of the preconditioned conjugate gradient (division of the residual by 10^{10}) — unstructured meshes.

The results concerning the number of iterations needed to solve the linear system are very interesting, but the total cost must include the enormous price of the computation of Lagrange multipliers.

Statistics for the two-level method

The statistics have the same meaning as in 2D (see Table 4.6). The results are given in Table 4.22.

Parameters	$h_{\max} = 0.4$	0.2	0.1
grid complexity	1.08	1.05	1.03
operator complexity	1.04	1.08	1.24

Table 4.22: Some statistics for the two-level method — unstructured meshes.

Chapter 5

Conclusion

Some known results relative to multilevel methods applied to Maxwell's equations and edge elements allows us to understand what we have accomplished, and we still have to accomplish. The implementation of an algebraic multilevel method for nodal elements and simple problems enabled us to construct an extension of this methods to edge elements, also for simple problems.

Some interesting and promising results have been obtained; however the methods are not currently conclusive. Some research directions can be imagined to evaluate the possibilities of this kind of method:

- full validation in 3D where we might relax the stopping criterion for the computation of Lagrange multipliers,
- more than two levels,
- on a structured grid, choose the geometric coarse nodal basis, and construct with the decoupled minimisation method the coarse edge basis. The aim of this experiment is to provide one more validation,
- test with a coarse incidence matrix A closer to the one used by Reitzinger and Schöberl; this can make comparisons easier,
- construct subdomains without geometric information,
- use of variable coefficients and/or domains with more complex geometry,
- efficient implementation to compare memory occupancy and computational time.

Bibliography

[1] D. N. Arnold, R. S. Falk, and R. Winther. Multigrid in $\mathbb{H}$(div) and $\mathbb{H}$(curl). *Numer. Maths*, 85:197–217, 2000.

[2] R. Beck. Algebraic Multigrid by Components Splitting for Edge Elements on Simplicial Triangulations. *Preprint SC 99-40, ZIB*, December 1999.

[3] R. Beck. Graph-Based Algebraic Multigrid for Lagrange-Type Finite Elements on Simplicial Meshes. *Preprint SC 99-22, ZIB*, July 1999.

[4] R. Beck and R. Hiptmair. Multilevel solution of the time-harmonic Maxwell's equations based on edge elements. *Int. J. Num. Meth. Engr.*, 45(7):901–920, 1999.

[5] Pavel B. Bochev, C. Garasi, J. Hu, A. Robinson, and R. Tuminaro. An improved algebraic multigrid method for solving Maxwell's equations. *SIAM J. Sci. Comput.*, 25(2):623–642 (electronic), 2003.

[6] L. Demkowicz, J. Gopalakrishnan, and J.E. Pasciak. Analysis of a multigrid algorithm for time harmonic Maxwell equations. *SIAM J. Numer. Anal.*, 42:90–108, 2005.

[7] R. Hiptmair. Multigrid method for Maxwell's equations. *SIAM J. Numer. Anal.*, 36:204–225, 1999.

[8] R. Hiptmair. Finite elements in computational electromagnetism. *Acta Numer.*, 11:237–339, 2002.

[9] M. Kaltenbacher and S. Reitzinger. Algebraic multigrid methods for nodal and edge based discretizations of maxwell's equations. *International Compumag Society Newsletter*, 9(2):15–23, 2002.

[10] J. C. Nédélec. Mixed finite elements in $\mathbb{R}^3$. *Num. Math.*, 35(3):315 – 341, 1980.

[11] R. Perrussel, L. Nicolas, and F. Musy. Un préconditionneur adapté l'opérateur rotationnel en éléments finis. 4th European Conference on Numerical Methods in Electromagnetism, NUMELEC 2003, 2003.

[12] R. Perrussel, L. Nicolas, and F. Musy. An efficient preconditioner for linear systems issued from the finite-element method for scattering problems. *IEEE Trans. on Mag.*, 40(2):1080–1083, mars 2004.

[13] R. Perrussel, L. Nicolas, F. Musy, and M. Schatzman. Preconditioners for finite element method in scattering problems. 4th European Congress on Computational Methods in Applied Sciences and Engineering, ECCOMAS 2004, 2004.

[14] S. Reitzinger and J. Schöberl. An algebraic multigrid method for finite element discretizations with edge elements. *Numer. Linear Algebra Appl.*, 9:223–238, 2002.

[15] J.W. Ruge and K. Stuben. Algebraic multigrid (amg). *Multigrid Methods (St.Mc Cormick ed.), Frontiers in Applied Mathematics, SIAM*, 5:73–130, 1987.

[16] W. L. Wan, Tony F. Chan, and Barry Smith. An energy-minimizing interpolation for robust multigrid methods. *SIAM J. Sci. Comput.*, 21(4):1632–1649 (electronic), 1999/00.