UCLA
COMPUTATIONAL AND APPLIED MATHEMATICS

Lecture Notes on Multilevel Methods for Elliptic Problems on Unstructured Grids

Tony F. Chan
Susie Go
Ludmil Zikatanov

March 1997

CAM Report 97-11

Department of Mathematics
University of California, Los Angeles
Los Angeles, CA. 90024-1555

Lecture Notes on Multilevel Methods for Elliptic Problems on Unstructured Grids †

Tony F. Chan* Susie Go* Ludmil Zikatanov*

March 18, 1997

Abstract

An overview of multilevel methods on unstructured grids for elliptic problems will be given. The advantages which make such grids suitable for practical implementations are flexible approximation of the boundaries of complicated physical domains and the ability to adapt the mesh to resolve fine-scaled structures in the solution. Multilevel methods, which include multigrid methods and overlapping and non-overlapping domain decomposition methods, depend on proper splittings of appropriate finite element spaces: either by dividing the original problem into subproblems defined on smaller subdomains, or by generating a hierarchy of coarse spaces. The standard splittings used in structured grid case cannot be directly extended for unstructured grids because they require a hierarchical grid structure, which is not readily available in unstructured grids.

We will discuss some of the issues which arise when applying multilevel methods on unstructured grids, such as how the coarse spaces and transfer operators are defined, and how different types of boundary conditions are treated. An obvious way to generate a coarse mesh is to re-grid the physical domain several times. In these lecture notes, we will propose and discuss different and possibly better alternatives: node nested coarse spaces, agglomerated coarse spaces and algebraically generated coarse spaces.

†Prepared for the lecture course "28-th Computational Fluid Dynamics", 3-7 March, 1997, von Karman Institute for Fluid Dynamics, Belgium.

*Department of Mathematics, University of California at Los Angeles, Los Angeles, CA 90095-1555, USA. E-mail: chan@math.ucla.edu, sgo@math.ucla.edu, lzikatan@math.ucla.edu. ONR under contract ONR-N00014-92-J-1890, and the Army Research Office under contract DAAL-03-91-C-0047 (Univ. of Tenn subcontract ORA4466.04 Amendment 1 and 2). The first two authors acknowledge support from RIACS under contract number NAS 2-13721 for visits to RIACS/NASA Ames. The third author is supported by Grant ONR-N00014-92-J-1890 and NSF Grant Int-95-06184.

Contents

1 Introduction

In these lecture notes, multilevel methods applied to problems on general unstructured grids will be discussed. The advantages which make the such grids interesting for the practical implementations are flexible approximation of the geometry and ability to adapt the mesh. We will describe various approaches for dealing with the solution of discrete equations arising from unstructured grids. Our interest will be in the performance of multilevel methods, including multigrid and domain decomposition methods.

The beauty of multilevel methods is that the convergence speed can often be proven to be independent of the problem size and they can be naturally parallelized. This makes them the most powerful and useful tool for a wide variety of applications. Unfortunately, these methods require a hierarchical grid structure, which is not readily available in unstructured grids. In our context, we use them not used as solvers on their own, but rather as preconditioners for Krylov subspace iterative methods.

Various approaches for dealing with these issues and their effect on the convergence properties of these methods will be covered. These notes are organized as follows: Section 1 begins with an introduction to Krylov subspace methods and multilevel methods, followed by some two-level theory in Section 2. Specific examples of how to deal with node-nested multilevel methods are covered in Section 3. Section 4 concerns agglomerated multigrid methods. The material contained in Section 5 deals with nonoverlapping domain decomposition methods.

Many of the topics described in these lecture notes represent previous and continuing joint work with Barry Smith and Jun Zou [1, 2, 3, 4], (Section 3), with Jinchao Xu [5] (Section 4) and with Timothy Barth and Wei-Pai Tang [6] (Section 5).

1.1 Elliptic problems

Elliptic problems are one of most extensively investigated problems in applied mathematics. Their relation to many physical models is well known and the theoretical and numerical results obtained in this area are very useful in practice. As a first approximation to more complicated physical and mathematical models (such as those in computational fluid dynamics), elliptic problems are sometimes the only ones for which rigorous theoretical results are known. The design of numerical methods for such model problems can often be adapted and applied to more complicated situations. Elliptic problems are also important in their own right, for example in computational fluid dynamics in the solution of the pressure equation, implicit time integration schemes, etc.

In this section, we will briefly review the properties of the model problems we consider. Our goal is to design effective solvers for the resulting systems of linear equations, and we will not pay much attention to the discretization techniques. Detailed discussions of the finite element element discretizations that we use can be found in [7, 8, 9, 10].

Let $\Omega \subset \mathbb{R}^d$ be a polygonal (polyhedral) domain, $d = 2, 3$. We consider the following differential problem:

$$\begin{cases} \mathcal{L}u \equiv \nabla \cdot (\alpha(x)\nabla u) = F(x) & x \in \Omega, \\ u = 0 & x \in \bar{\Gamma}_D \\ \dfrac{\partial u}{\partial \eta} = 0 & x \in \Gamma_N. \end{cases} \tag{1.1}$$

The following formulation is known as the variational (or Galerkin) formulation of (1.1): Find $u \in H_0^1(\Omega; \Gamma_D)$ such that

$$a(u, v) = F(v) \text{ for all } v \in H_0^1(\Omega; \Gamma_D), \tag{1.2}$$

where

$$a(u, v) = \int_\Omega \alpha(x)\nabla u \nabla v dx, \quad F(v) = \int_\Omega F(x) v dx. \tag{1.3}$$

Here $H_0^1(\Omega; \Gamma_D)$ denotes the Sobolev space which contains functions which vanish on Γ_D with square integrable first derivatives. It is well-known that (1.2) is uniquely solvable if $\alpha(x)$ is a strictly positive scalar function and F is square integrable.

One of the basic tools for solving such problems is the finite element method. Discretizations based on this method are popular because they are very robust and can easily deal with different types of boundary conditions and complicated geometries in two or three spatial dimensions. Much of what we say in these lecture notes also applies to finite difference discretizations.

We will use the simplest finite element discretization of the elliptic problem (1.2). First, cover Ω with simplicial finite elements (triangles in $\mathbb{R}^2$ and tetrahedra in $\mathbb{R}^3$). Then the discrete problem could be formulated as follows:

Find $u_h \in V_h$ such that

$$a(u_h, v_h) = F(v_h) \text{ for all } v_h \in V_h, \tag{1.4}$$

where V_h is the finite dimensional subspace of $H_0^1(\Omega; \Gamma_D)$ consisting of continuous functions linear on each of the simplexes forming the partition.

The values of the discrete solution on the grid nodes are then determined by solving the resulting system of linear equations:

$$Au = f, \tag{1.5}$$

where A is a symmetric and positive definite matrix, f is the right hand side and the nodal values of the discrete solution u_h will be obtained in u after solving the system (1.5). The matrix $A = \{a_{ij}\}_{i,j=1}^n$ and the right hand side $f = \{f_j\}_{j=1}^n$ are obtained through equation (1.4) , using the standard nodal basis finite element functions. These functions are determined as linear over each simplex element and for every i the corresponding basis function satisfies the relation $\varphi_i(x_j) = \delta_{ij}$, where x_i are the coordinates of the i-th grid node. Using this basis we get the following expressions for the elements of A and f: $a_{ij} = a(\varphi_i, \varphi_j)$, $f_j = f(\varphi_j)$. To obtain an accurate enough approximate solution of (1.2), one often has to solve huge discrete problems, which are badly conditioned, with condition number growing like $O(h^{-2})$, where h is the characteristic mesh size. Our goal in the next sections will be to construct robust and effective methods for solving the discrete equations (1.5) and investigate their properties.

1.2 Unstructured grids

With the vast improvements in computational resources today, the motivating reasons for using structured grids over unstructured grids become less obvious. Cartesian or mapped Cartesian grids are popular because they are directional, so efficient methods can be used, such as the alternating direction implicit methods (ADI) and fast Fourier transforms (FFT). This structure, however, imposes limitations on the types of domains which can be considered. In addition, local refinement cannot be easily done without affecting large portions of the grid, so the ability to adapt the grids for resolving steep gradients in the solution is a source of difficulty.

One of the alternative approaches for dealing with complicated geometries is the composite grid method as proposed by Brown, Chesshire, Henshaw and Kreiss [11] and Chesshire and Henshaw [12]. Different alternatives will also be discussed and proposed during this course by other speakers.

Unstructured grids provide the flexibility needed to adapt to rapidly changing or dynamic solutions as well as complex geometries. These grids have irregular connectivity and so do not have to adhere to the strict structure of Cartesian-based grids, see figure 1.1. Computations on unstructured grids require more complicated data structures and possibly modifications in some solvers, e.g. multilevel methods. Such modifications in multilevel methods will be further discussed in these lecture notes.

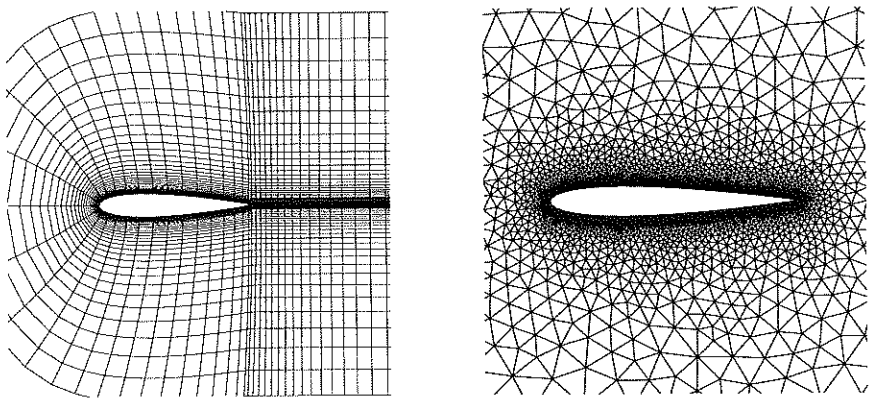


Figure 1.1: A structured grid (left) and unstructured grid (right).

1.3 Preconditioned iterative methods

As we mentioned in the introduction, multilevel methods will be used in our framework as preconditioners in Krylov subspace iterative methods. This section is a brief introduction to preconditioned iterative methods. We will discuss the basic properties of the most popular Krylov subspace iterative methods: the Conjugate Gradient (CG) and the Generalized Minimum Residual (GMRES) method. We will be interested particularly in their preconditioned versions and the presentation here will closely follow Saad [13]. We emphasize that the summary here does not pretend to be complete; details can be found in [13] and references therein.

We will use the lower case letters $u, v, w, \ldots$ to denote vectors, capital letters $A, B, \ldots$ for matrices, and $(\cdot, \cdot)$ will denote the Euclidian scalar product in $\mathbb{R}^n$. Let $A \in \mathbb{R}^{n \times n}$ be a given square matrix and $v \in \mathbb{R}^n$ be a vector. Denote by $\|\cdot\|$, the Euclidean norm in $\mathbb{R}^n$, i.e. $\|v\|^2 = (v, v)$. The spectral radius of A, $\rho(A)$, is defined by $\rho(A) = \max_{1 \leq i \leq n} |\lambda_i|$, where λ_i's are the eigenvalues of A. The matrix norm corresponding to $\|v\|$ is given by $\|A\| = \sqrt{\rho(AA^T)} = \sqrt{\rho(A^T A)}$. Another quantity which will be often used in the analysis of iterative methods is the condition number of A defined by $\kappa(A) = \|A\|\|A^{-1}\|$. Also, if A is symmetric positive definite, the A-norm (energy norm) is defined by $\|v\|_A^2 = (Av, v)$. For symmetric positive definite (SPD) matrices, the above definition of the condition number of A is equivalent to

$$\kappa(A) = \frac{\max_{1 \leq i \leq n} \lambda_i}{\min_{1 \leq i \leq n} \lambda_i}. \tag{1.6}$$

Let us now split A in the following way:

$$A = M - N,$$

where $M \approx A$ is a non-singular matrix. In accordance with this splitting, we obtain the following linear *fixed-point* iteration:

$$u^{k+1} = M^{-1}Nu^k + M^{-1}f = u^k + M^{-1}(f - Au^k). \tag{1.7}$$

The matrix M is called the *preconditioning matrix*. The matrix $G = I - M^{-1}A$ is often called the *iteration matrix* or *iterator*.

The well-known result related to the convergence of the above procedure is stated in the following basic theorem:

Theorem 1.1 *The iterative procedure (1.7) converges for any initial guess u^0 if and only if $\rho(I - M^{-1}A) < 1$.*

A serious drawback of basic iterative methods (e.g. Richardson, Jacobi, Gauß-Seidel) is the fact that the number of iterations needed is often huge for ill-conditioned A. We summarize the results concerning these methods in the next theorem (see [14]). For this, we split A in the usual way: $A = D - L - U$, where D is the diagonal of A, and $-L$ and $-U$ are the strictly lower and upper triangular parts of A, respectively.

For the definitions of M, we have:

$$M^{-1} = \begin{cases} \omega I & \text{Richardson,} \\ D^{-1} & \text{Jacobi,} \\ \omega D^{-1} & \text{Damped Jacobi,} \\ (D - L)^{-1} & \text{Gauß-Seidel,} \\ \omega(D - \omega L)^{-1} & \text{SOR,} \\ \omega(2 - \omega)(D - \omega U)^{-1} D (D - \omega L)^{-1} & \text{SSOR.} \end{cases} \tag{1.8}$$

Theorem 1.2 *If A is symmetric positive definite then*

- *the Richardson method converges iff $0 < \omega < 2/\rho(A)$,*

- *the Jacobi method converges iff* $2D - A$ *is symmetric and positive definite,*
- *the Damped Jacobi method converges iff* $0 < \omega < 2/\rho(D^{-1}A)$,
- *the Gauß-Seidel method always converges,*
- *the SOR and SSOR methods converge iff* $0 < \omega < 2$.

If A is SPD, it is more convenient to use a symmetric M. If the initially chosen M is not symmetric, a natural way to symmetrize it is by performing two successive iterations, one with M and the other with M^T:

$$\begin{aligned} u^{k+1/2} &= u^k + M^{-1}(f - Au^k) \\ u^{k+1} &= u^{k+1/2} + M^{-T}(f - Au^{k+1/2}). \end{aligned} \tag{1.9}$$

The symmetrized iterative scheme also can be written in the form:

$$u^{k+1} = u^k - M_{symm}^{-1}(Au^k - f), \tag{1.10}$$

where $M_{symm}^{-1} = M^{-T} + M^{-1} - M^{-T}AM^{-1}$. SSOR is in fact the symmetrization of SOR.

The rate of convergence for these simple relaxation schemes depends on the condition number of A. For finite element and finite difference equations such as (1.5), the asymptotic convergence rate for Gauß-Seidel method is of order $1 - O(h^2)$, which makes the method impractical for small mesh sizes h.

These basic methods can be accelerated by Krylov subspace methods, e.g. the preconditioned conjugate gradient (PCG) when A is SPD and the preconditioned GMRES (PGMRES) for general non-symmetric A. We will describe briefly several important examples of these Krylov subspace methods.

Very often it is benefitial to precondition (1.5) before applying any iterative method:

$$M^{-1}Au = M^{-1}f. \tag{1.11}$$

When A and M are both SPD, it is more convenient to work with the symmetrized version of (1.11):

$$\begin{aligned} &\hat{A}\hat{u} = \hat{f}, \quad \text{where} \quad \hat{A} = M^{-\frac{1}{2}}AM^{-\frac{1}{2}}, \\ &\hat{u} = M^{\frac{1}{2}}u \quad \text{and} \quad \hat{f} = M^{-\frac{1}{2}}f. \end{aligned} \tag{1.12}$$

The choice of M is very important because it can improve the convergence rate. A good preconditioner M for A should have the following properties:

- The action of $M^{-1}v$ for a given vector v should be less expensive to compute, than $A^{-1}v$.
- The condition number $\kappa(\hat{A})$ should be as close to 1 as possible, preferably uniformly bounded above (with respect to the mesh size h).
- If A is SPD then M should be SPD.

The Krylov subspace of dimension k, $K_k(A, r^0)$, is the subspace that is spanned by vectors $r^0, Ar^0, \ldots, A^k r^0$. Let us now set $r^0 = f - Au^0$. The CG method is a minimization algorithm which minimizes the energy norm (A-norm) of the error in the space $u^0 + K_k$. The PCG method is essentially the same minimization procedure, but the minimization now is done using the preconditioned Krylov space:

$$K_k(\hat{A}, \hat{r}_0) = \text{span}\{\hat{r}^0, \hat{A}\hat{r}^0, \ldots \hat{A}^{k-1}\hat{r}^0\},$$

defined through $\hat{A}$ from the symmetrized version (1.12).

Preconditioned CG

1. Set $r^0 = f - Au^0$, and solve $Mz^0 = r^0$;
2. Set $p^0 = z^0$, $i = 0$;
3. While No-Convergence Do;
4. $\quad i = i + 1$;
5. $\quad \alpha_i = (r^i, z^i)/(Ap^i, p^i)$;
6. $\quad u^{i+1} = u^i + \alpha_i p^i$;
7. $\quad r^{i+1} = r^i - \alpha_i A p^i$;
8. $\quad$ Solve $Mz^{i+1} = r^{i+1}$;
9. $\quad \beta_i = (r^{i+1}, z^{i+1})/(r^i, z^i)$;
10. $\quad p^{i+1} = z^{i+1} + \beta_i p^i$;
11. endWhile

Taking $M = I$, one obtains the usual CG method. Among the basic relaxation methods listed in (1.8), only Jacobi and SSOR method can be used as preconditioners for CG when A is SPD.

The convergence of the PCG method depends on the condition number of the matrix $\hat{A}$. If the condition number $\kappa(\hat{A})$ is uniformly bounded with respect to n, then the iteration (1.7) has a uniform damping factor independent of n. More precisely, the following result holds:

Theorem 1.3 *Let A and M be SPD matrices and $\|v\|_A = (Av, v)$. Let u be the solution of the system (1.5). Then for the k-th iterate u^k the following inequality holds*

$$\|u - u^k\|_A \leq 2\left(\frac{\sqrt{\kappa(\hat{A})} - 1}{\sqrt{\kappa(\hat{A})} + 1}\right)^k \|u - u^0\|_A. \tag{1.13}$$

Next we describe the preconditioned GMRES algorithm, which can be used for solving systems with non-symmetric matrices. There are many variants of GMRES; we include here the so-called right preconditioned GMRES with restart.

Right Preconditioned $GMRES(m)$

1. Solve $r^0 = f - Au^0$, and set $v^1 = r^0/\|r^0\|$, $s = \|r^0\|e_1$;
2. For $i = 1, 2, \ldots, m$ do
3. Compute $w = AM^{-1}v^i$
4. For $k = 1, 2, \ldots i$
5. $h_{ki} = (w, v^k)$;
6. $w = w - h_{ki}v^k$;
7. endFor
8. $h_{i+1i} = \|w\|$;
9. $v^{i+1} = h_{i+1i}/\|w\|$;
10. endFor
11. Set $V_m = [v^1, \ldots, v^m]$ and $H_m = \{h_{ij}\}$, where $j = 1, \ldots, m$; $i = 1, \ldots, j+1$;
12. Solve the least-squares problem: $y^m = \min_y \|H_m y - s\|$
13. Set $u^m = u^0 + M^{-1}V_m y^m$
14. If *convergence* then
15. Stop
16. else
17. $u^0 = u^m$
18. go to 1.
19. endIf

In the above algorithm, care must be taken in making the choice of the restart parameter m. In some cases, however, the convergence is independent of this choice, as it is stated in the following theorem:

Theorem 1.4 *If $(AM^{-1} + M^{-T}A^T)/2$ is positive definite then the preconditioned GMRES converges for any number $m \geq 1$.*

The analysis of the GMRES convergence is more difficult and challenging than for the CG method. The following theorem gives a bound for the residual after k iterations.

Theorem 1.5 *Suppose that AM^{-1} is diagonalizable, i.e. there exists non-singular matrix C such that $AM^{-1} = C\Lambda C^{-1}$, where $\Lambda = \mathrm{diag}\{\lambda_1, \ldots, \lambda_n\}$. Define*

$$\varepsilon^{(k)} = \min_{p \in \mathcal{P}_k, p(0)=1} \max_{1 \leq i \leq n} |p(\lambda_i)|,$$

where $\mathcal{P}_k$ is the space of polynomials of degree $\leq k$. Then the residual norm after k iterations of right preconditioned GMRES method can be bounded by:

$$\|r^k\| \leq \kappa(C)\varepsilon^{(k)}\|r^0\|, \tag{1.14}$$

where $\kappa(C) = \|C\|\|C^{-1}\|$.

There is also a very useful variant of the above right preconditioned GMRES method which allows the use of different preconditioners M_i at each step. This variant is called *flexible GMRES*. For completeness we give the algorithm below.

Right Preconditioned Flexible $GMRES(m)$

1. Solve $r^0 = f - Au^0$, and set $v^1 = r^0/\|r^0\|$, $s = \|r^0\|e_1$;
2. For $i = 1, 2, \ldots, m$ do
3. Compute $z_i = M_i^{-1} v^i$
4. Compute $w = Az^i$
5. For $k = 1, 2, \ldots i$
6. $h_{ki} = (w, v^k)$;
7. $w = w - h_{ki} v^k$;
8. endFor
9. $h_{i+1i} = \|w\|$;
10. $v^{i+1} = h_{i+1i}/\|w\|$;
11. endFor
12. Set $Z_m = [z^1, \ldots, z^m]$ and $H_m = \{h_{ij}\}$, where $j = 1, \ldots, m$; $i = 1, \ldots, j+1$;
13. Solve the $least - squares$ problem $y^m = \min_y \|H_m y - s\|$
14. Set $u^m = u^0 + Z_m y^m$
15. If *convergence* then
16. Stop
17. else
18. $u^0 = u^m$
19. go to 1.
20. end If

Note that the flexible variant in addition requires m vectors $z_i = M_i^{-1} v^i$ to be stored in memory (compared to the previous algorithm). If $M_i = M$ for all i, then the flexible GMRES method is mathematically equivalent to GMRES(m).

In addition to CG and GMRES, there are many other Krylov subspace methods which can be used as alternatives, especially for non-symmetric A's. Some popular methods are BiCGSTAB, CGS, QMR, TFQMR, BCG and their many variants. At this point there is no widespread agreement on the relative merits of these methods.

In the next sections, our particular interest will be focused on multilevel methods (such as domain decomposition methods and multigrid methods) used as preconditioners in PCG. The popularity of these methods as preconditioners is based on the fact that they exactly fit in the applications where finite element or finite difference method is used. In other words, the design of such preconditioners uses the properties of finite element spaces which allows precise optimal constructions and theoretical analysis to be done.

1.4 Multilevel methods

For many practical problems, the system of linear equations which arises from finite element or finite difference discretizations might be huge — on the order of 10^5 to 10^6 unknowns. A challenge is how to effectively solve such large systems of linear equations. Direct methods face the problem of excessive memory requirements and number of the floating point operations needed. In this connection, iterative methods, and especially multilevel methods such as multigrid and domain decomposition methods, are very attractive. These methods are popular because the amount of work required to solve a problem is on the order of the number of unknowns, the convergence rates are independent of the problem size and they can be easily parallelized.

1.4.1 Multigrid methods.

In this section, we briefly describe the multigrid methods for solving linear systems of discrete equations. We will consider the case where these systems are obtained via finite element discretization of an elliptic partial differential equation. Detailed discussion on multigrid methods can be found in standard references, e.g. Briggs [15], Bramble [16], Hackbusch [17], and Xu [18, 14].

The idea behind multigrid methods is based on the fact that simple relaxation schemes such as Gauß-Seidel, Jacobi and Richardson possess good smoothing property: they reduce the highly oscillatory part of the error very well in few iterations. This part of the error lies in the subspace spanned by the eigenvectors corresponding to large eigenvalues, i.e. the high frequencies. The global error, or the low frequencies unfortunately cannot be corrected well by such iterative schemes and this is where multigrid helps. The low frequencies from fine grid (say original one) are transfered to the coarse grid, where they behave like high-frequencies, and are smoothed quickly by a simple relaxation scheme. Recursive application of this idea leads to the multigrid method.

We will denote the space which contains the solution u by V_J. We assume that coarse grids are given and with each grid we associate a finite dimensional space (like V_J for the fine grid). We denote these spaces by $V_0, \ldots, V_{J-1}$. To unify the notation in this section we define $A_J := A$. We assume that the operators A_k, $k = 0, \ldots J-1$, are given (these operators correspond to different approximations of A on the coarse grids). We also assume that the prolongation operator R_k^T and the smoothing operators S_k are also given. One can consider the action of the smoother on $g \in V_k$ as a fixed number of Gauß-Seidel or Jacobi iterations with right-hand side g and zero initial guess.

We view the multigrid method as a way of defining a preconditioner M_J. We will describe in matrix notation the action $M_J^{-1}g$ in the simplest case when one pre- and post-smoothing steps are applied.

The action of M_k^{-1} is then obtained through the following steps:

ALGORITHM 1.1 *(V-cycle multigrid preconditioner $M_k^{-1}g$)*

0. *If $k = 0$ then $M_0^{-1}g = A_0^{-1}g$*
1. Pre-smoothing*: Apply one transposed smoothing iteration with initial guess $x^0 = 0$ and right hand side g, i.e.*

$$x^1 = S_k^T g$$

2. Coarse grid correction*:*
   1. *Restrict the residual: $q^0 = R_k(I - A_k S_k^T)g$.*
   2. *"Solve" on the coarse grid: $q^1 = M_{k-1}^{-1}q^0 = M_{k-1}^{-1}R_k(I - A_k S_k^T)g$.*
   3. *Interpolate back and correct:*
      $x^2 = x^1 + R_k^T q^1 = \left[S_k^T + R_k^T M_{k-1}^{-1}R_k(I - A_k S_k^T)\right] g$.
3. Post-smoothing*: Apply one smoothing iteration with initial guess x^2 and right hand side g, i.e.*

$$\begin{aligned} M_k^{-1}g &= x^2 + S_k(g - A_k x^2) \\ &= \left[S_k + S_k^T - S_k A_k S_k^T + (I - S_k A_k)R_k^T M_{k-1}^{-1}R_k(I - A_k S_k^T)\right] g. \end{aligned}$$

Here R_k^T is the formal adjoint of R_k with respect to the scalar product $(\cdot, \cdot)$. Note also that the above definition is recursive, the action of $M_k^{-1}g$ is defined in terms of $M_{k-1}^{-1}g$. Let us now consider the simplest case: a two-level method (when $J = 1$). For the sake of simplicity we omit the index 1 in the next equation. We have:

$$M^{-1}g = \left[S + S^T - SAS^T + (I - SA)R^T A_0^{-1}R(I - AS^T)\right] g.$$

The iteration matrix $G = I - M^{-1}A$ (see Section 1.3) is given by:

$$\begin{aligned} I - M^{-1}A &= I - \left[S + S^T - SAS^T + (I - SA)R^T A_0^{-1}R(I - AS^T)\right] A \\ &= I - SA - S^T A + SAS^T A - (I - SA)R^T A_0^{-1}R(A - AS^T A) \\ &= (I - SA) - (I - SA)S^T A(I - SA)R^T A_0^{-1}RA(I - S^T A) \\ &= (I - SA)(I - R^T A_0^{-1}RA)(I - S^T A). \end{aligned}$$

Let us now consider the more general case when ν_1^k pre-smoothing and ν_2^k post-smoothing steps are applied. In this case, the recursively defined V-cycle preconditioner cannot be written in simple form. We will use a form in which A_k^{-1} is involved but it is straightforward to see that the terms involving this inverse are cancelled out:

$$M_k^{-1}g = \left[I - (I - S_k A_k)^{\nu_1^k}(I - R_k^T M_{k-1}^{-1}R_k A_k)(I - S_k^T A_k)^{\nu_2^k}\right] A_k^{-1}g.$$

In this case the iteration matrix has more convenient form:

$$I - M_k^{-1}A_k = (I - S_k A_k)^{\nu_1^k}(I - R_k^T M_{k-1}^{-1}R_k A_k)(I - S_k^T A_k)^{\nu_2^k}. \tag{1.15}$$

One flexible variant of this preconditioner is called *variable* V-cycle preconditioner (see [19], [20]), in which the number of smoothing steps is doubled on each level, i.e. $\nu_1^k = \nu_2^k = 2^{J-k}$, for $k = J, \ldots, 0$.

1.4.2 Domain decomposition methods

Domain decomposition (DD) methods are divide-and-conquer methods which take a large problem defined on a physical domain, and appropriately decompose it into many smaller problems defined on subdomains. These smaller subdomain problems can then be solved quickly and independently of each other and their solution suitably combined, usually via an iterative process to obtain the solution to the original problem. The domain decomposition methods we will discuss fall into two broad categories: overlapping DD (Schwarz methods) and nonoverlapping DD (substructuring or Schur complement methods). Our description here follows that in Chan-Mathew [21]; see also the recently published book by Smith, Bjørstad and Gropp [22].

1.4.2.1 "Classical" domain decomposition The idea of domain decomposition was introduced in the late 1800's by H. A. Schwarz [23] for proving existence of harmonic functions on complicated domains. The Schwarz alternating procedure in its simplest form solves a problem on Ω by decomposing the physical domain into two subdomains $\Omega = \Omega_1 \cup \Omega_2$, where $\Omega_1 \cap \Omega_2 \neq \emptyset$, see Fig. 1.2. Given an initial guess, successive iterates are found by solving the subproblems alternately on Ω_1 and Ω_2, with the updated values used as boundary conditions on $\partial\Omega_1 \setminus \partial\Omega$ and $\partial\Omega_2 \setminus \partial\Omega$, respectively.

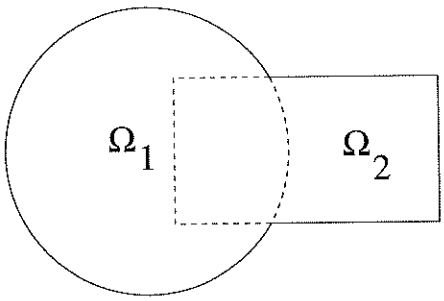


Figure 1.2: Schwarz' original diagram of overlapping domain decomposition.

Many years later, limited computational resources (especially computer memory) lead to the development of the iterative substructuring method by Kron [24] and Przemieniecki [25]. In examples shown in Fig. 1.3, Ω is divided into nonoverlapping subdomains which are separated from each other by an interface, Γ, so that $\Omega_1 \cap \Omega_2 = \emptyset$, $\Omega = \bar{\Omega}_1 \cup \bar{\Omega}_2$, and $\Gamma = \bar{\Omega}_1 \cap \bar{\Omega}_2$. After elimination of the unknowns on $\Omega \setminus \Gamma$, the much smaller interface problem $Su_\Gamma = g_\Gamma$ is then solved. The motivation for this approach was that identical substructures could be exploited and the interface problem, which was much smaller than the original problem, could then be solved by direct methods since it would fit in the limited amount of memory available in those days.

1.4.2.2 "Modern" domain decomposition The recent resurgence of interest in domain decomposition methods is motivated by computational advantages that these methods possess:

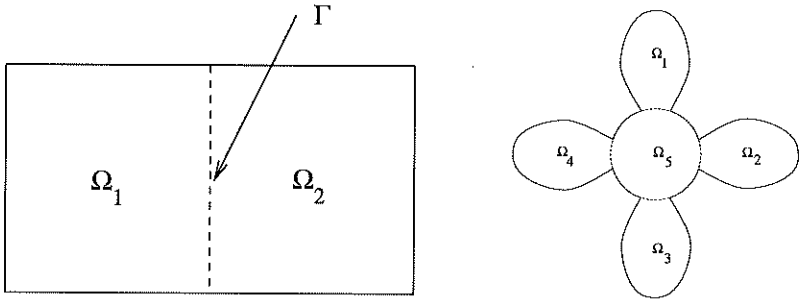


Figure 1.3: Some nonoverlapping domain decompositions. Dotted lines represent interfaces, Γ, between subdomains.

- Parallelizability: DD respects the different speeds of the *memory* hierarchy. Subdomain problems can be solved efficiently within the fast memory of individual processors, with only rare but costlier communication across processors.
- Scalability: DD can lead to *optimal* algorithms, whose efficiency are independent of the mesh width, h, and subdomain width, H, see Fig. 1.4.
- Adaptivity: DD methods are flexible enough to allow different models, meshes, and solvers to be used on each subdomain.
- Geometry: DD methods can exploit fast solvers on subdomains with regular geometries.
- Software: Software for subdomain solvers can be reused on different processors.

Modern DD methods are also different from their classical counterparts in several ways:

- they don't form the interface matrix S explicitly, as it is costly,
- they don't solve the interface problem $S_\Gamma u_\Gamma = g_\Gamma$ using direct methods, but instead use preconditioned iterative methods.

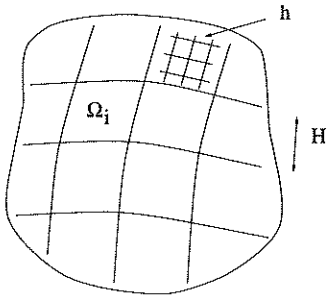


Figure 1.4: Domain decomposition with fine structures of size, h, and coarse structures of size, H.

1.4.2.3 Overlapping DD We will describe domain decomposition methods in matrix notations for solving the linear system (1.5), which arises from the Dirichlet problem discretized by Galerkin finite elements. The main ingredients required in all DD methods are:

- **Restriction matrices**: Let R_i be the $n_i \times n$ restriction matrix of 1's and 0's which takes a full-length vector in $\mathbb{R}^n$ and maps it to a restricted vector in $\mathbb{R}^{n_i}$, where n_i denotes the number of unknowns in subdomain Ω_i. The effect on an n-vector is injection onto the subdomain, Ω_i.

- **Extension matrices**: Let R_i^T be the $n \times n_i$ extension matrix, which is defined as the transpose of the restriction matrix, R_i. The effect on an n_i-vector is identity on the subdomain, Ω_i, and zero extension outside the subdomain, i.e. on $\Omega \setminus \Omega_i$. (1.16)

- **Subdomain matrices**: Define the local stiffness matrix on Ω_i to be $A_i = R_i A R_i^T$, where $A_i \in \mathbb{R}^{n_i \times n_i}$. Because the restriction and interpolation matrices consist only of 0's and 1's, the local stiffness matrices are simply principal submatrices of A.

- **Subdomain solvers**: Let A_i^{-1} symbolically denote the solver for the restricted operator. These can be either exact or inexact solvers (see Sec. 1.4.2).

In overlapping DD methods, a set of p overlapping subdomains are formed by taking a set of nonoverlapping subdomains $\{\Omega_i'\}_{i=1}^p$, and extending them to larger subdomains, $\{\Omega_i\}_{i=1}^p$ by some small distance, δ, see Fig. 1.5. Let A_i be the local stiffness matrix on Ω_i, R_i and R_i^T be the corresponding restriction and extension matrices, respectively. The resulting subproblems are then solved independently of each other.

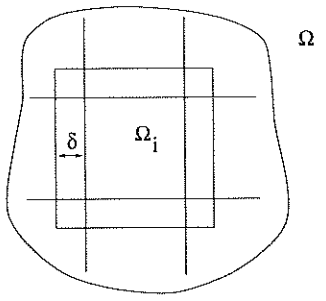


Figure 1.5: Generating a set of overlapping subdomains.

The partitioning induced by such a decomposition amounts to an overlapping block decomposition of the system (1.5). Thus, the overlapping DD methods can be thought of as block iterative solvers, either overlapping block Jacobi or block Gauß-Seidel, depending on whether or not the the most updated iterates are used for boundary conditions.

The additive Schwarz (block Jacobi) method on p subdomains is given by:

$$u^{k+i/p} = u^{k+(i-1)/p} + R_i^T A_i^{-1} R_i (f - Au^k), \quad i = 1, ..., p.$$

In this form, it is seen that corrections are done simultaneously on p subdomains. Rewriting this as one equation reveals the preconditioned iterative method:

$$u^{k+1} = u^k + M_{as}^{-1}(f - Au^k)$$

where the preconditioner M_{as} is given by:

$$M_{as}^{-1} = \sum_{i=1}^{p} R_i^T A_i^{-1} R_i.$$

Additive Schwarz preconditioner. (*block Jacobi on A*)

$$M_{as}^{-1} = \sum_{i=1}^{p} R_i^T A_i^{-1} R_i. \tag{1.17}$$

Instead of simultaneous corrections, the corrections can also be done successively, to yield the multiplicative Schwarz (block Gauß-Seidel) method:

$$u^{k+i/p} = u^{k+(i-1)/p} + R_i^T A_i^{-1} R_i (f - Au^{k+(i-1)/p}), \quad i = 1, ..., p.$$

Because the most currently updated information is used, this method will generally converge faster than additive Schwarz. The drawback is that it is less parallel (but this can be remedied by appropriate coloring of the subdomains).

For the multiplicative Schwarz method on p subdomains, the preconditioner can be written as:

Multiplicative Schwarz preconditioner. (*block Gauß-Seidel on A*)

$$M_{ms}^{-1} = [I - (I - R_p^T A_p^{-1} R_p A) \cdots (I - R_1^T A_1^{-1} R_1 A)] A^{-1}. \tag{1.18}$$

Note that the multiplicative Schwarz preconditioner is non-symmetric with respect to the A norm. A symmetrized multiplicative Schwarz method can constructed by the additional application of subdomain steps in reverse order. The preconditioner for the symmetrized multiplicative Schwarz method is:

$$M_{ms,symm}^{-1} = [I - (I - R_1^T A_1^{-1} R_1 A) \cdots (I - R_p^T A_p^{-1} R_p A)(I - R_p^T A_p^{-1} R_p A) \cdots (I - R_1^T A_1^{-1} R_1 A)] A^{-1}.$$

1.4.2.4 Coarse grid The domain of dependence for elliptic problems is the entire domain, but because Schwarz methods decompose the problem into smaller, independent problems, information from one subdomain must travel large distances to reach another subdomain. To avoid deterioration of the convergence rates of these methods, some sort of mechanism for the global transfer of data is needed. This is achieved, to some degree, by the overlapping of subdomains in the Schwarz methods. More overlap leads to more coupling between subdomains. However, this adds redundant work and communications overhead if too much overlap is introduced. Dryja and Widlund [26, 27] showed that the condition number for additive Schwarz is given by:

$$\kappa(M_{as}^{-1}A) = O\left(H^{-2}\left(1+\left(\frac{H}{\delta}\right)^2\right)\right).$$

The condition number is independent of h. For sufficient amount of overlap (choosing $\delta = O(H)$), the condition number is $O(H^{-2})$ and so will increase as H tends to zero. This means that the method will not be scalable to a large number of processors.

This deterioration can be remedied by introducing a coarse grid to achieve additional global coupling, see fig. 1.6. In addition to the subdomain restriction, interpolation and stiffness matrices used in the one-level Schwarz methods (1.16), we need coarse versions of them: $R_H, R_H^T, A_H = R_H A R_H^T$, and A_H^{-1}. Here, R_H and R_H^T will instead be the full weighting restriction and linear interpolation matrices, respectively, which are commonly used in multigrid methods. The two-level additive Schwarz preconditioner can then be written as:

Additive Schwarz preconditioner with coarse grid.

$$M_{asc}^{-1} = R_H^T A_H^{-1} R_H + \sum_{i=1}^{p} R_i^T A_i^{-1} R_i.$$

It can be shown that the condition number for this two-level method is:

$$\kappa(M_{asc}^{-1}A) = O(1 + (H/\delta)^2),$$

and the method can be made independent of H, h with sufficient overlap by choosing $\delta = O(H)$.

The condition numbers for the one- and two-level additive Schwarz methods above are dependent on the coefficients α in the PDE (1.1). For coefficients with large jumps but which are constant or mildly varying within each coarse grid element, it can be shown that the condition number for the two-level additive Schwarz algorithm is given by:

$$\kappa(M_{asc}^{-1}A) \;=\; O\left(1+\log\left(\frac{H}{h}\right)\right) \quad \text{in 2D},$$

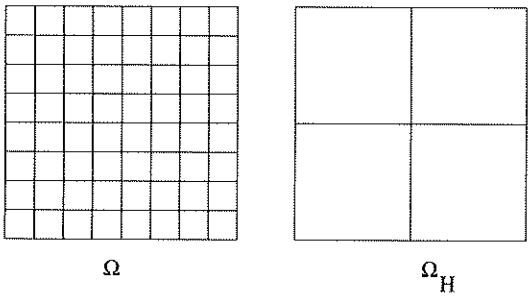


Figure 1.6: Fine grid, Ω, and coarse grid, Ω_H, for two-level domain decomposition methods.

and

$$\kappa(M_{asc}^{-1}A) \;=\; O\,(H/h) \quad \text{in 3D.}$$

1.4.2.5 Multilevel Schwarz Multilevel Schwarz is an extension of two-level Schwarz with l different coarse levels, each level being decomposed into i subdomains as previously described. We will denote the i^{th} subdomain on the l^{th} level as: Ω_i^l. Several different variants of multilevel Schwarz can be created, depending on when the most currently updated information is used:

- Fully additive multilevel methods would be additive among subdomains on the same level as well as additive between levels.
- Multilevel methods which are multiplicative among subdomains on the same level, but additive between levels can be viewed as "additive MG".
- Classical V-cycle MG can be viewed as a multilevel Schwarz method which is multiplicative both among subdomains on the same level as well as between levels.

The fully additive multilevel Schwarz preconditioner can be written as:

Fully additive multilevel Schwarz preconditioner.

$$M_{mlas}^{-1} = \sum_l \sum_i (R_i^l)^T (A_i^l)^{-1} (R_i^l).$$

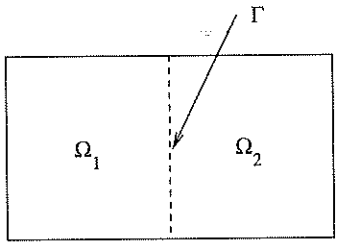


Figure 1.7: Nonoverlapping domain decomposition with two subdomains, Ω_1 and Ω_2 and interface separating the two subdomains, Γ.

1.4.2.6 Nonoverlapping DD

Two subdomain case. Let us consider first, the two subdomain case separated by one interface: $\Omega = \bar{\Omega}_1 \cup \bar{\Omega}_2$, and $\Gamma = \bar{\Omega}_1 \cap \bar{\Omega}_2$, see Fig. 1.7.

By reordering the linear system (1.5) so that the unknowns on the subdomains and the interface boundary are grouped together, the solution can be written as $u = (u_1, u_2, u_\Gamma)^T$, the right-hand side as $u = (f_1, f_2, f_\Gamma)^T$ and the linear system as:

$$\begin{pmatrix} A_{11} & 0 & A_{1\Gamma} \\ 0 & A_{22} & A_{2\Gamma} \\ A_{\Gamma 1} & A_{\Gamma 2} & A_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_\Gamma \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_\Gamma \end{pmatrix}. \tag{1.19}$$

Here, the blocks A_{12} and A_{21} are zero because we are assuming there is no coupling between the two subdomains. For discretizations with small stencils, this will be true.

Eliminating u_1, u_2 from (1.19) gives the Schur complement system:

$$Su_\Gamma = g_\Gamma \tag{1.20}$$

where

$$\begin{aligned} S &= A_{\Gamma\Gamma} - A_{\Gamma 1}A_{11}^{-1}A_{1\Gamma} - A_{\Gamma 2}A_{22}^{-1}A_{2\Gamma} \\ g_\Gamma &= f_\Gamma - A_{\Gamma 1}A_{11}^{-1}f_1 - A_{\Gamma 2}A_{22}^{-1}f_2. \end{aligned}$$

Although the Schur complement system is small, solving it by direct methods is expensive since the Schur complement matrix S is dense. Also, forming S requires as many subdomain solves as there are nodes on each of the boundary edges. It can be shown that the condition number of the Schur complement system is $\kappa(S) = O(h^{-1})$, an improvement over the condition number of the original system, $\kappa(A) = O(h^{-2})$, but still not optimal for iterative methods. Thus the solution is typically done by preconditioned Krylov subspace methods. One only needs to compute Sv, which requires solves on Ω_i, e.g. in $A_{\Gamma 1}A_{11}^{-1}A_{1\Gamma}v$. However, one still needs a good *interface preconditioner* for S and there are many different ones available. One such interface preconditioner is $\sqrt{\Delta_{1D}}$, where Δ_{1D} is the one-dimensional Laplacian matrix.

Many subdomain case.(see Fig. 1.8)

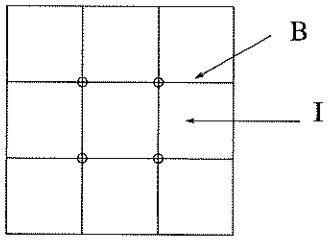


Figure 1.8: Nonoverlapping domain decomposition with many subdomains and interface B.

As before, we group the set of unknowns into interior nodes, u_I, and boundary nodes, u_B, which in turn are made up of edges and vertices, $u_B = (u_E, u_V)^T$. Then as before, the system can be written in block notation as:

$$\begin{pmatrix} A_{II} & A_{IB} \\ A_{BI} & A_{BB} \end{pmatrix} \begin{pmatrix} u_I \\ u_B \end{pmatrix} = \begin{pmatrix} f_I \\ f_B \end{pmatrix}$$

where $A_{II} = \text{block diag}(A_{11}, A_{22}, \ldots, A_{pp})$. After elimination of the interior nodes, the Schur complement system is:

$$Su_B = f_B - A_{BI}A_{11}^{-1}f,$$

where

$$S = A_{BB} - A_{BI}A_{II}^{-1}A_{IB}.$$

It can be shown that $\kappa(S) = O(h^{-1}H^{-1})$, an improvement over the $O(h^{-2})$ condition number of A, but again requiring the use of good preconditioners. There are many available, one such example by Bramble, Pasciak, Schatz [28] is:

$$M_{BPS}^{-1} = R_H^T A_H^{-1} R_H + \sum_{i=1}^{m} R_{E_i}^T M_{E_i}^{-1} R_{E_i},$$

where A_H is the matrix associated with the vertex separators of the coarse grid, $R_H : B \to V$ is restriction from the set of all interface boundary nodes to the vertex separators, $R_{E_i} : B \to E_i$, is restriction from the set of all interface boundary nodes to the i^{th} edge E_i, and M_{E_i} is an approximation to A_{E_i}, the local stiffness matrix restricted to edge E_i. The condition number of this preconditioner can be shown to be:

$$\kappa(M_{BPS}^{-1}S) = O\left(1 + \log^2\left(\frac{H}{h}\right)\right).$$

The dependence on H and h occurs because the edges E_i and V are not coupled in the preconditioner. This dependence can be removed by introducing one more level of coupling, namely, adding small "vertex spaces" which are small neighborhoods around each vertex to provide some overlap between edges and vertices (see [29]).

1.4.2.7 Inexact subdomain solves In all of the domain decomposition methods described above, subdomain solves, A_i^{-1}, are required. These can be done either exactly or inexactly. Though the subdomain and coarse problems are much smaller than the original problem, it can still be quite expensive to attempt exact solves on these problems.

In the two-level additive Schwarz methods, we can simply replace the exact solves with inexact solves: let $M_i \approx A_i$, $M_H \approx A_H$ represent the inexact solves. Then the preconditioner is given by:

$$\tilde{M}_{asc}^{-1} = R_H^T M_H^{-1} R_H + \sum_{i=1}^{p} R_i^T M_i^{-1} R_i.$$

In the Schur complement methods, let $M_{II} = \text{block diag}(M_{ii})$, where $M_{ii} \approx A_{ii}$ and $M_S \approx S$. Then from the block LU factorization of A:

$$A = \begin{pmatrix} A_{II} & A_{IB} \\ A_{BI} & A_{BB} \end{pmatrix} = \begin{pmatrix} I & 0 \\ A_{BI}A_{II}^{-1} & I \end{pmatrix} \begin{pmatrix} A_{II} & A_{IB} \\ 0 & S \end{pmatrix}$$

we can define the following preconditioner:

$$M = \begin{pmatrix} I & 0 \\ A_{BI}M_{II}^{-1} & I \end{pmatrix} \begin{pmatrix} M_{II} & A_{IB} \\ 0 & M_S \end{pmatrix}.$$

Note that:

- the iteration is on the whole domain, not just on the interfaces, and
- $M^{-1}v$ requires 2 solves on each subdomain.

Finally, care must be taken in terms of the scaling of A_{BI}, A_{IB} with M_{II}, M_S

1.5 Approaches for designing multilevel methods on unstructured grids

The multilevel methods require a hierarchical grid structure. For structured grids, these grids can be recovered from the fine grid. For unstructured grids, however, there is no natural grid hierarchy. In addition, their lack of structure prevents these methods from exploiting regularity and using fast solvers as with structured grids. Difficulties exist in identifying coarse grid problems/spaces/boundary conditions which do not occur when using structured grids. The algorithms which are based on unstructured grids must be re-designed to handle these issues without sacrificing too much in terms of complexity and performance.

There are several approaches for constructing the coarse spaces for unstructured grids. The first one (see Mavriplis [30]) is based on independently generated coarse grids and piecewise linear interpolation between the grids. The advantage of this approach is convenience: the coarse grids can be generated by using the same grid generator which produced the original fine grid. The disadvantage is that the interpolations can

be expensive to apply since the set of nodes in the coarse grids are not related in any way to the nodes in the fine grid. Thus no fast search routines can be applied and the implementation will be $O(n^2)$.

An alternative approach is based on generating node-nested coarse grids, which are created by selecting subsets of a vertex set, retriangulating the subset, and using piecewise linear interpolation between the grids (see [31, 32]). This still provides an automatic way of generating coarse grids and now faster implementations of the interpolation (can be implemented in $O(n)$ time). The drawback is that in three dimensions, retetrahedralization can be problematic.

Another effective coarsening strategy proposed by Bank and Xu [33] uses the geometrical coordinates of the fine grid (which is available in most cases).

New coarsening strategies based on the algebraic approach recently were published by Hackbusch [34], Braess [35] and Reusken [36].

In many of these approaches, problems may occur in producing coarse grids which are valid and with boundaries which preserve the important features of the fine domain. One of most popular and promising new coarsening techniques which avoids this problem is based on the agglomeration technique (see Koobus, Lallemand and Dervieux [37]). Instead of constructiong a proper coarse grid , a neighboring fine grid elements are agglomerated together to form macroelements. Since these agglomerated regions are not standard finite elements, appropriate basis functions and interpolation operators must be constructed on them. Such algorithms have also been investigated by Mandel, Vaněk, Brezina [38]) and Vaněk, Křížková [39]. This approach essentially uses a simple initial interpolation matrix, which might not be stable, and then this matrix is smoothed and stabilized by some of the basic relaxation schemes, e.g. Jacobi method.

2 Introduction to convergence theory

As mentioned in Section 1.3, the estimate of the convergence rate of the PCG requires an estimate of the upper bound of $\kappa(M^{-1}A)$. In particular, estimates on the extreme eigenvalues of $M^{-1}A$ must be obtained. In this section, we first give a general framework for bounding $\kappa(M^{-1}A)$ and then we show how such an analysis can be carried out for the overlapping domain decomposition method. Such an analysis can show (or predict) the convergence rate and in most cases gives a good guess as to how the parameters and approximate operators should be chosen in order to get an optimal iterative method. For a similar approach in analyzing the convergence properties of iterative methods using general subspace splittings for structured meshes, we refer to [18].

We shall adopt a matrix approach for analyzing the domain decomposition methods, in the hope that it is more intuitive and easier to understand. Such a presentation of the analysis can also be found in (see e.g. [22]). More references concerning the theoretical analysis of the domain decomposition methods also can be found there.

Although we are going to present the domain decomposition methods in matrix formulation, the use of Sobolev norms cannot be avoided in a few places, so we will define the notation for them here.

Let Ω be a fixed domain in $\mathbb{R}^d$. The norm in Sobolev space $H^k(\Omega)$ is defined to be:

$$\|u\|_k = \left(\sum_{|\alpha| \leq k} \int_\Omega [D^\alpha u(x)]^2 dx \right)^{\frac{1}{2}},$$

and the seminorm in $H^k(\Omega)$ is defined by:

$$|u|_k = \left(\sum_{|\alpha| = k} \int_\Omega [D^\alpha u(x)]^2 dx \right)^{\frac{1}{2}},$$

where $\alpha = (\alpha_1, \ldots, \alpha_d)$ is a multi-index, $D^\alpha = \dfrac{\partial^{|\alpha|}}{\partial x_1^{\alpha_1} \ldots x_d^{\alpha_d}}$ and $|\alpha| = \alpha_1 + \ldots + \alpha_d$. When the domain (e.g. Ω) needs to be emphasized or clarified, the notation for the seminorm and norm will be $|u|_{k,\Omega}$ and $\|u\|_{k,\Omega}$, respectively.

2.1 Subspace correction framework: matrix formulation

Our initial setting in matrix form is as follows: Let Ω be covered by p overlapping subdomains Ω_i, $i = 0, 1, \ldots, p$. Each subdomain Ω_i corresponds to a subspace $V_i \subset \mathbb{R}^n$. The subspaces are defined through the restriction operators $R_i \in \mathbb{R}^{n_i \times n}$, $i = 0, 1, ..., p$, and we set $V_i = \text{Range}(R_i)$

REMARK. Note here that we will interchangably use the notation for the coarse grid versions denoted with subscript H in the previous section, with the subscript 0, when convenient. Here, V_0 denotes the coarse space.

We wish to construct a preconditioner for solving the following linear algebra problem

$$Au = f, \quad A \in \mathbb{R}^{n \times n} \text{ is SPD.} \tag{2.1}$$

Let us first explain the intuition behind the construction of a preconditioner based on this splitting of $\mathbb{R}^n$. It is natural to take the best approximation to the solution from each subspace, and then to extend these different approximations to the whole $\mathbb{R}^n$ somehow in order to get a global solution. Thus the question is: What is the best correction to the k-th iterate u^k from V_i?

If we measured the error in the A-norm, $\|\cdot\|_A$, then this question can be reformulated as the following minimization problem:

$$\min_{y_i} \|(u^k + R_i^T y_i) - A^{-1} f\|_A. \tag{2.2}$$

The solution is given by:

$$y_i = (R_i A R_i^T)^{-1} R_i (f - Au^k) = A_i^{-1} R_i (f - Au^k). \tag{2.3}$$

The next iterate is then obtained via the equation (note that we correct here only in one subspace V_i)

$$u^{k+1} = u^k + R_i^T A_i^{-1} R_i (f - Au^k) \tag{2.4}$$

Example. The Jacobi iteration (see Section 1.3) corresponds to the splitting $V_i = span\{e_i\}$ where e_i is the i-th unit coordinate vector. The restrictions, R_i, in this case are defined as $R_i v = (v, e_i)e_i$.

Performing these subspace corrections simultaneously gives the additive subspace correction preconditioner:

$$M_{asc}^{-1} = \sum_{i=0}^{p} R_i^T A_i^{-1} R_i.$$

Defining now the projections $P_i \equiv R_i^T A_i^{-1} R_i A$, for $i = 0, \ldots p$, we get

$$M_{asc}^{-1} A = \sum_{i=0}^{p} P_i.$$

As we pointed out earlier, the convergence of the PCG method depends on the condition number of $M^{-1}A$. Thus our goal is to find an upper bound for $\kappa(M^{-1}A)$ which amounts to finding an upper bound for $\lambda_{\max}(M_{asc}^{-1}A)$ and a lower bound for $\lambda_{\min}(M_{asc}^{-1}A)$.

The estimate on the upper bound for $\lambda_{\max}(M_{asc}^{-1}A)$ is easier and it follows directly from the following simple lemmas.

Lemma 2.1 *P_i is a projection in $(\cdot,\cdot)_A$, i.e.*

$$AP_i = P_i^T A, \quad P_i^2 = P_i \quad \|P_i\|_A \leq 1.$$

Proof. This follows by direct verification. □

Lemma 2.2 *The maximal eigenvalue of the preconditioned matrix satisfies the following inequiality:*

$$\lambda_{\max}(M_{asc}^{-1}A) \leq p + 1.$$

Proof. This follows from the simple fact that

$$\rho\left(\sum_{i=0}^{p} P_i\right) \leq \left\|\sum_{i=0}^{p} P_i\right\|_A \leq \sum_{i=0}^{p} \|P_i\|_A = p + 1.$$

□

REMARK. The bound given in the previous lemma can be easily improved to:

$$\lambda_{\max}(M_{asc}^{-1}A) \leq n_c + 1 \equiv c_1,$$

where n_c is the *number of colors to color Ω_i's* in such a way that no two neighboring subdomains are colored the same color.

We next give an estimate on the lower bound for $\lambda_{\min}(M_{asc}^{-1}A)$. This estimate is based on the following *Partition Lemma* which plays a crucial role in the convergence analysis of the domain decomposition methods.

Lemma 2.3 (Partition Lemma). *(Matsokin-Nepomnyaschikh [40], Lions [41], and Dryja-Widlund [42, 26]). Assume that there exists a constant c_2 such that*

$$\min_{\substack{u=\sum u_i \\ u_i \in V_i}} \sum_{i=0}^{p} \|u_i\|_A^2 \leq c_2 \|u\|_A^2. \tag{2.5}$$

then

$$\lambda_{\min}(M_{asc}^{-1}A) \geq \frac{1}{c_2}.$$

Proof. Using the fact that the minimal eigenvalue minimizes the Rayleigh, quotient we get

$$\lambda_{\min}(M_{asc}^{-1}A) = \lambda_{\min}\left(\sum_{i=0}^{p} P_i\right) = \min_{u \neq 0} \frac{\Sigma_{i=0}^{p}(P_i u, u)_A}{(u,u)_A}.$$

Let us now consider the decomposition $u = \Sigma_{i=0}^{p} u_i$, $u_i \in V_i$. Since the P_i's are projections, we have

$$(u,u)_A = \sum_{i=0}^{p} (u_i, u)_A = \sum_{i=0}^{p} (u_i, P_i u)_A.$$

Applying now the Cauchy-Schwarz inequality, we obtain

$$\sum_{i=0}^{p} (u_i, P_i u)_A \leq \left(\sum_{i=0}^{p} (u_i, u_i)_A\right)^{\frac{1}{2}} \left(\sum_{i=0}^{p} (P_i u, P_i u)_A\right)^{\frac{1}{2}}.$$

After taking minimum over u_i, we get

$$(u,u)_A \leq c_2^{\frac{1}{2}} (u,u)_A^{\frac{1}{2}} \left(\sum_{i=0}^{p} (P_i u, P_i u)_A\right)^{\frac{1}{2}}$$

and thus

$$(u,u)_A^{\frac{1}{2}} \leq c_2^{\frac{1}{2}} \left(\sum_{i=0}^{p} (P_i u, P_i u)_A\right)^{\frac{1}{2}} = c_2^{\frac{1}{2}} \left(\sum_{i=0}^{p} (P_i u, u)_A\right)^{\frac{1}{2}}.$$

Squaring both sides and proper rearranging then yields the desired result. □

The assumption we have made in the partition lemma (equation (2.5)) means that for any given u, a stable decomposition must exist in the sense that the sum of the "energy" of all the pieces u_i lying in V_i is bounded by the global energy norm of the decomposed vector. This assumption can be viewed as a condition on the V_i's, i.e. the subspaces must not introduce oscillations (high energy components) in u_i.

Combining lemmas 2.1– 2.3, we get our main theorem:

Theorem 2.1 *If assumption (2.5) holds, then for the condition number $\kappa(M_{asc}^{-1}A)$, can be bounded by:*

$$\kappa(M_{asc}^{-1}A) \leq c_1 c_2. \tag{2.6}$$

This result suggests how to construct the decompositions in order to obtain optimal preconditioners. It is immediately seen that we want the constants c_1, c_2 to be independent of the problem parameters such as the number of subdomains, the characterisctic mesh sizes h and H, jumps in the coefficients of the underlying PDE, etc. It is also desirable to make c_1 and c_2 as small as possible in order to get a condition number close to 1. But c_1 and c_2 depend on the size of overlaps in the subspaces V_i. More overlap will decrease c_2, but the number of colors c_1 will increase. On the other hand, small overlap will lead to large c_2 and small c_1. Thus the space decompositions have to be made in such a way to ensure that the product $c_1 c_2$ is as small as possible.

2.2 Application to two-level overlapping domain decomposition methods

As an example of the application of the above theory, we will present a detailed estimate for the condition number of the two-level Schwarz method.

2.2.1 The intuitive idea

As in the previous section, we first present the basic intuitive idea using a simple 1D version of (1.1).

We want a splitting which satisfies the partition assumption (2.5). Take Ω to be a fixed open interval on the real line and cover Ω with p overlapping subdomains Ω_i, $i = 1, \ldots, p$ (see fig 2.1). Consider the partition of unity θ_i corresponding to this covering. By construction the functions θ_i satisfy

$$\sum_{i=1}^{p} \theta_i = 1, \qquad 0 \leq \theta_i \leq 1, \qquad |\theta_i|_{1,\infty} \leq \delta^{-1}, \tag{2.7}$$

where δ is the size of the overlap and $|\theta|_{s,\infty}$ denotes the maximum, norm, i.e. the maximum of the s-th derivative of θ_i. We define $u_i = \theta_i u$, so we have $u = \Sigma_{i=0}^{p} u_i$.

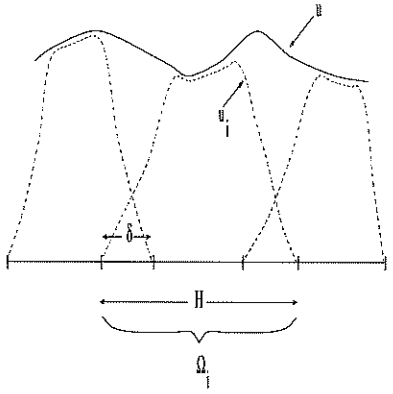


Figure 2.1: No coarse grid.

Since A corresponds to a discretization of the second order elliptic operator, $\frac{d}{dx}\alpha(x)\frac{d}{dx}$, it is easy to see that the A-norm and the H^1-seminorm are equivalent in this case:

$\|u\|_A \approx \|du/dx\|_0$. Our goal is to bound $\|u_i\|_A$ by $\|u\|_A$. Looking at Fig. 2.1, we see that the function u_i changes from $\|u\|_0$ to 0 over a distance δ and we get:

$$\|u_i\|_A^2 = \left\| \left(\frac{du}{dx} \right) \right\|_0^2 \leq c \left(\frac{\|u\|_0}{\delta} \right)^2 .$$

We still need to bound $\|u\|_0$ by $\|u\|_A$. But the function, u, satisfying homogenous Dirichlet boundary conditions, cannot change rapidly over the interval Ω if there is no significant change in the derivative. The well-known Poincaré inequality estimates the norm of the function with the norm of derivatives and its application leads to the following:

$$\left(\frac{\|u\|_0}{\delta} \right)^2 \leq \tilde{c} \left(\frac{\|u\|_A}{\delta} \right)^2 .$$

After summing over all subdomains, we get:

$$\sum_{i=1}^{p} \|u_i\|_A^2 \leq O\left(\frac{1}{\delta^2} \right) \|u\|_A^2 .$$

Therefore,

$$c_2 \;=\; O\left(\frac{1}{\delta^2} \right) = O\left(\left(\frac{H}{\delta} \right)^2 \frac{1}{H^2} \right) .$$

From these inequalities one may conclude that if the overlap is of size $O(H)$, then $\kappa(M^{-1}A) = O(H^{-2})$, which is an improvement over $O(h^{-2})$, but is still unsatisfactory. We can see that the overlapping subdomains alone cannot provide a stable partition of u.

It turns out that this dependence on H can be eliminated by using a global coarse space, V_H, which couples all the subdomains. The idea is to construct a coarse grid approximation u_H to u satisfying the following two important properties:

$$\|u_H\|_A \;\leq\; c\|u\|_A \tag{2.8}$$

$$\|u - u_H\|_0 \;\leq\; cH\|u\|_A \tag{2.9}$$

Define $w = u - u_H$ and the following partition of u:

$$u_i = \theta_i(u - u_H), \quad u = u_H + \sum_{i=1}^{p} u_i. \tag{2.10}$$

Proceeding as before and taking into account that now the pieces u_i change from 0 to $O(H)$ (not to 1 because of the approximation property (2.12)), we have

$$\|u_i\|_A^2 \leq c \left(\frac{H\|u\|_A}{\delta} \right)^2 \leq c \left(\frac{H}{\delta} \right)^2 \|u\|_A^2 .$$

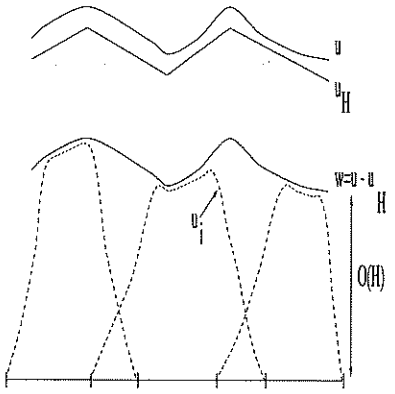


Figure 2.2: With coarse grid.

If we make the natural assumption that $\delta = O(H)$, the bound for c_2 now reads

$$c_2 = O\left(\left(\frac{H}{\delta}\right)^2\right) = O(1).$$

This estimate shows how the condition number can be improved and actually such a choice of M^{-1} gives a good preconditioner. Thus, we can see that the role of the coarse grid V_H is to make $\|u - u_H\|$ small enough ($O(H)$), so that it can be partitioned in a stable manner.

We would like to comment on the choice of u_H. As it can be seen (compare to section 1.4.2), u_H is a purely theoretical construction and there is no need of its use in the algorithm. One possible choice is $u_H = \mathcal{R}_H u$, where $\mathcal{R}_H$ is a kind of interpolation or projection operator. Of course, $\mathcal{R}_H$ must satisfy properties similar to (2.8) and (2.9) namely

$$|\mathcal{R}_H u|_1 \leq c|u|_1 \quad \text{(stability)} \tag{2.11}$$

$$\|\mathcal{R}_H u - u\|_0 \leq cH|u|_1. \quad \text{(approximation)} \tag{2.12}$$

A natural candidate for such an operator is the nodal value interpolant on the coarse grid, $u_H = I_H u$. A drawback of such a choice is that in $3D$ this interpolation does not satisfy the stability property (2.11). To see this, we take w to be the basis function ψ_i^h associated with the grid node x_i. Then we have

$$|w|_1^2 = \int_{\Delta_h} \left(\tfrac{1}{h}\right)^2 = O(h)$$

$$|I_H w|_1^2 = \int_{\Delta_H} \left(\tfrac{1}{H}\right)^2 = O(H).$$

The last estimate shows that the stability requirement is violated.

If the grid is structured then a good and stable coarse grid approximation to the elements of V_h is the *L_2-projection* Q_H from $V_h \to V_H$ and we can define $\mathcal{R}_H = Q_H$, i.e. $u_H = Q_H u$. It is known that this u_H satisfies the stability and approximation properties (2.11) and (2.12) (see Xu [18] or Dryja and Widlund [42]).

2.2.2 Proof of the condition number bound

The above intuitive explanations will now be made mathematically rigorous.

First let us assume that $\mathcal{R}_H$ satisfies stability and approximation properties (2.11) and (2.12). Further, for $u \in V_h$, we define:

$$w = u - \mathcal{R}_H u, \quad u_i = I_h(\theta_i w), \quad u_0 = \mathcal{R}_H u \quad i = 1, \ldots, p,$$

where I_h denotes the nodal value interpolant on the fine grid. In what follows we will implicitly use the following inequalities, which can be proved in a straightforward way:

$$\begin{aligned} |I_h(\theta_i w)|_1 &\leq c|wI_h\theta_i|_1, \\ |I_h\theta_i|_{s,\infty} &\leq c|\theta_i|_{s,\infty}, \quad s = 0, 1. \end{aligned}$$

We now state the main result of this section.

Theorem 2.2 *Under the above assumptions for the condition number $\kappa(M_{asc}^{-1}A)$, the following estimate holds*

$$\kappa(M_{asc}^{-1}A) = O(1 + (H/\delta)^2). \tag{2.13}$$

Proof. Using the stability assumption on $\mathcal{R}_H$, and the equivalence between $\|\cdot\|_A$ and $|\cdot|_1$, it is easy to check that the following inequalities hold for any $u \in V_h$:

$$\begin{aligned} \|w\|_0^2 \leq cH^2|u|_1^2 \qquad & \text{(approximation)} \\ |w|_1^2 \leq c|u|_1^2 \qquad & \text{(stability)} \\ \|u_0\|_A^2 \leq c|u|_1^2 \qquad & \text{(stability)} \end{aligned}$$

It follows then that the smallest eigenvalue of A can be estimated as follows:

$$\begin{aligned} \|u_i\|_A^2 &\leq c|u_i|_1^2 = c|I_h(\theta_i w)|_1^2 \\ &\leq c|\theta_i|_{1,\infty}^2\|w\|_0^2 + c\|\theta_i\|_{0,\infty}^2|w|_1^2 \\ &\leq c\delta^{-2}\|w\|_0^2 + c|w|_1^2 \\ &\leq c\delta^{-2}H^2|u|_1^2 + c|u|_1^2 \\ &\leq c\left(1 + (H/\delta)^2\right)|u|_1^2 \\ &\leq c\left(1 + (H/\delta)^2\right)\|u\|_A^2. \end{aligned}$$

To complete the estimate let us assume that each grid point belongs to at most γ subdomains Ω_i's, where γ is a fixed number independent of h and H. Then we have:

$$\sum_{i=0}^{P}\|u_i\|_A^2 \leq c\left(1 + (H/\delta)^2\right)\|u\|_A^2 \equiv c_2\|u\|_A^2.$$

Therefore the constant c_2 in the partition lemma (lemma 2.3) is $O(1 + (H/\delta)^2)$, which implies that

$$\kappa(M_{asc}^{-1}A) = O(1 + (H/\delta)^2).$$

□

2.2.3 Some extensions

Inexact subdomain solves can easily be accommodated by using $\|\cdot\|_{M_i}$ satisfying:

$$\|u\|_A \leq \omega \|u\|_{M_i} \qquad \forall u \in V_i.$$

Then the constant ω in the above inequality will be absorbed in the resulting bound for $\kappa(M^{-1}A)$.

The extension to *multilevel* Schwarz method is straightforward. An additional assumption, however is needed in this case:

$$(u_i, u_j)_A \leq \varepsilon_{ij} \|u_i\|_A \|u_j\|_A \qquad \forall u_i \in V_i, \forall u_j \in V_j.$$

Such inequalities measure the abstract angles between the subspaces and are known in the literature as "strengthened Cauchy Schwarz inequalities". For a detailed discussion of the issues concerning multilevel theory we refer to Xu [18, 14], Chan Mathew [21]. Note that $\rho(\{\varepsilon_{ij}\})$ will enter in the bound for $\kappa(M^{-1}A)$.

We now briefly comment on the convergence of the *multiplicative* Schwarz method which was described in Section 1.4.2. From (1.18) for the error e^{k+1}, we have:

$$e^{k+1} = (I - P_p) \cdots (I - P_0) e^k.$$

Since each $(I - P_i$ is a projection in the A-norm, it immediately follows that $\|e^{k+1}\|_A \leq \|e^k\|_A$. The following result gives a bound for the damping factor of the multiplicative iteration.

Theorem 2.3 *(Bramble, Pasciak, Wang, Xu [43]; and Xu [18]) Let V_i's satisfy the assumptions of the Partition Lemma. Then the following estimate is true:*

$$\|(I - P_p) \cdots (I - P_0)\|_A \leq 1 - \frac{c}{c_2}.$$

where c depends on the number of colors for coloring the Ω_i's but is independent of p.

2.3 Convergence of multigrid methods

The convergence properties of *multigrid* methods (see Section 1.4.1) depend on many parameters. One can vary the number of smoothing steps, the smoothing operators, the interpolation and restriction operators, coarse grid operators, etc. There are two main approaches in constructing multigrid preconditioners. One of them uses nested subspace splittings of V_h and the other one uses non-nested spaces or specially interpolated bilinear forms (coarse grid matrices). The discussion of the convergence in both these cases is given in [16, 43, 18, 14]. Here we give the simplest convergence result in the case of *nested* spaces and so-called full elliptic regularity assumption:

$$\|u\|_2 \leq c \|F\|_0.$$

where u is the solution, F is the right hand side of (1.1).

We first consider the case when the spaces $V_0, \ldots, V_1$ are nested, i.e. $V_0 \subset V_1 \subset \ldots V_{J-1} \subset V_J = V_h$. Again, the stability and approximation properties (2.11) and (2.12) are crucial in the convergence theory. The stability property (2.11) is automatically satisfied when the spaces are nested. It turns out that from the regularity assumption the following approximation property follows (see Xu [18, 14]):

There exists a constant, c_1, independent of the mesh parameters (i.e. of the mesh size h) such that

$$\|I - P_{k-1}v\|_A^2 \leq c_1 \frac{1}{\rho(A_k)} \|A_k v\|_2^2 \quad \forall v \in V_k, \tag{2.14}$$

where P_k denotes the elliptic projection defined by $(AP_k u, v) = (Au, v) \ \ \forall v \in V_k$.

The other operator involved in the definition of M_J^{-1} is the smoother and we make the following assumption on it

$$\frac{c_0}{\rho(A_k)}(v, v) \leq (S_{symm,k} v, v) \leq (A_k^{-1} v, v). \tag{2.15}$$

The smoother $S_{symm,k}$ is the symmetric version of S_k and is defined as: $S_{symm,k} := S^T + S - S^T A S$. Inequalities of the type (2.15) are satisfied by the Gauß-Seidel method.

An important thing to mention for the choice of the smoother is that we are trying to choose smoother which will quickly capture the high frequency components of the error, and we are not going to use it as a solver. For example if the matrix A_J corresponds to the five point finite difference stencil, it can be seen that the Jacobi method (with $\omega = 1$) is not good for a smoother. One must use the damped Jacobi method with $\omega < 1$.

The subspace correction framework presented in the previous section applies to multigrid methods as well. As long as the stability and approximation properties are verified, the following convergence result holds:

Theorem 2.4 *Under the assumptions (2.14) and (2.15), the following estimate is true:*

$$\|I - M_J^{-1} A\|_A \leq 1 - \frac{c_0}{c_1}. \tag{2.16}$$

We want to point out that a convergence result similar to Theorem 2.4 is also true in the *non-nested* case. We refer to work by Bramble et.al. [43] or Chan-Zou [44, 45] for unstructured grids. The construction of interpolation operators satisfying the stability property might be an issue. Some possible stable definitions can be found in [44] and [45] and in Section 3.2.

In the next sections, we will define subspaces satisfying the above assumptions or directly satisfying the approximation property similar to (2.14). The verification of these assumptions is easy for structured grids and can be found in many papers. On unstructured grids, however this might be rather complicated and tricky. Special attention should be paid to the construction of the spaces themselves, rather than using standard spaces and verifying the above inequalities. Thus the crucial point in designing effective subspace correction methods (such as multigrid itself) is the definition of the subspaces. When the grid is structured, this might be done in many different ways and one can obtain subspaces with excellent approximation and stability properties without

too much effort. The problem is that on unstructured grids it is not clear how to split the subspace in such a way that the low frequencies are transferred to the coarse grid successfully. Heuristically speaking, taking a "detailed" (i.e. more smooth) transfer operator sometimes results in dense coarse grid matrices. On the other hand, if the low frequencies are not well represented on the coarse grid, the convergence might be poor and the method is no longer optimal.

3 Node-nested coarse spaces

Unstructured multilevel methods for solving linear systems like (1.5) require a hierarchy of coarse grids. Grids which are node-nested have the advantage that they can be automatically generated and that efficient methods can be used to create the interpolation and restriction operators needed to transfer information from one level to the other. Disadvantages are that for complicated geometries, particularly in three dimensions, special care must be taken to ensure that the coarse grids which are produced are valid and preserve the important geometric features of the fine domain. With unstructured meshes, the grid hierarchy can allow general grids which are non-quasiuniform and coarse grids whose boundaries may be non-matching to the boundary of the fine grid, so care must be applied when constructing intergrid transfer operators for various types of boundary conditions. In this section, we will discuss some possibilities.

3.1 Maximal independent set (MIS) coarsening

An automatic approach to generating node-nested coarse grids is to take a maximal independent set (MIS) of the vertices and retriangulate the resulting vertex set [31, 32]. We will call the set of nodes in the MIS, the coarse grid nodes. A sequence of coarse grids can thus be created by repeated application of this technique. A maximal independent set of vertices in a graph is a subset of vertices which is *independent* in the sense that no two vertices in the subset are connected by an edge, and *maximal* if the addition of a vertex results in a dependent subset.

A simple technique for finding a MIS of vertices is to first choose a MIS of the boundary vertices by choosing every other boundary vertex and eliminating all its nearest neighbors, and then find a MIS of the interior vertices by selecting a random interior vertex and eliminating all its nearest neighbors, and repeating the process until all vertices are either eliminated or selected. The resulting vertex subset is then retriangulated using for example, the same triangulation routine which generated the original fine grid.

3.2 Coarse-to-fine interpolations

In general, the resulting coarse grid boundary, $\partial\Omega_H$, of the coarse grid will not match the original boundary, $\partial\Omega$, of the fine grid so the coarse space V_H is usually not a subspace of the fine space V_h. Indeed, even if $\Omega_H = \Omega$, V_H may still not be a subspace of V_h since the coarse elements are generally not the unions of some fine elements in unstructured

grids. To construct a coarse-to-fine transfer operator, one can use the standard nodal value interpolant associated with the fine space, V_h.

ALGORITHM 3.1 *(Standard nodal value interpolation)*

1. **For** *each fine grid node,*
2. Search *through all coarse grid elements until the coarse grid element which contains it is found.*
3. **If** *the fine grid node is a coarse grid node, then*
4. Set *the interpolant to be equal to that nodal value,*
5. **Else**
6. Set *it to be a linear interpolation of the 3 nodal values making up that coarse grid element (see Fig. 3.1).*

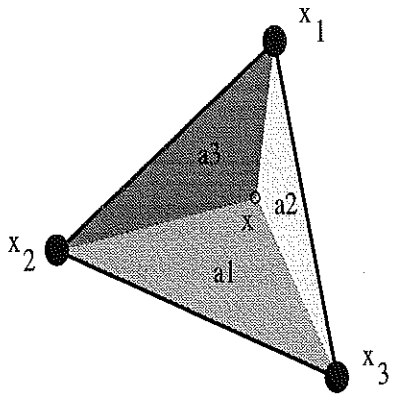


Figure 3.1: Barycentric (natural) coordinates: $\lambda_i(x) = \frac{\text{area}(a_i)}{\text{area}(\Delta_{123})}$, for $i = 1, 2, 3$, where Δ_{123} is the simplex with vertices x_1, x_2, x_3. The value of a function f at a point x is given by: $f(x) = \lambda_1(x)f(x_1) + \lambda_2(x)f(x_2) + \lambda_3(x)f(x_3)$.

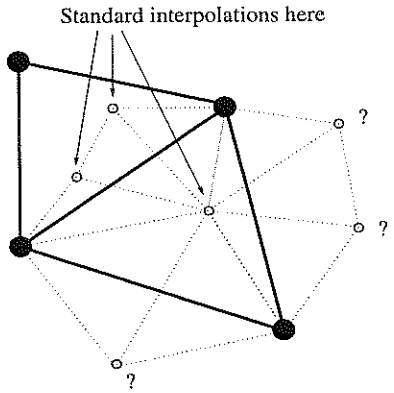


Figure 3.2: Use standard interpolation for fine grid nodes interior to a coarse element. What should be done for fine grid nodes which are not interior to any coarse grid element? One way is simply with extension by zero.

A naive implementation of this routine requires $O(n^2)$ time, but exploiting the node-nested property of the grids, one can implement this in $O(n)$ time, since only nearest coarse grid elements of a fine node need to be searched.

3.3 Interpolations on non-matching boundaries

Notice, however, that the zero extension interpolant is only well defined for those fine nodes lying also in the coarse domain $\bar{\Omega}_H$, but undefined for those fine nodes lying outside $\bar{\Omega}_H$. That is, in Step 2 of the standard nodal value interpolation (Algorithm 3.1), there is no provision for what to do if all the coarse grid elements have been searched, and none contains the fine grid node. A simple and natural way to remove this barrier is to assign those fine node values by zero. We shall denote this interpolant as the coarse-to-fine interpolant, $\mathcal{I}_h^0$.

Where coarse grid boundary conditions are of Dirichlet type, the standard nodal value interpolants with zero extensions can be accurate enough for interpolating fine grid values outside the coarse grid domain Ω_H; we refer to [32, 1] for the theoretical and numerical justifications of $\mathcal{I}_h^0$.

The zero extension interpolant, $\mathcal{I}_h^0$, works well for Dirichlet boundary conditions but will not be accurate nor stable for other boundary conditions. We provide a simple one dimensional example to illustrate why better interpolants are needed at non-matching boundaries. This example has a Dirichlet boundary condition at the left boundary point and a homogeneous Neumann boundary condition at the right boundary point. The interpolated function is the solid line and the coarse grid approximation to it is the dotted line. For Neumann boundary conditions the elements from V_h which have to be interpolated are generally not zero at the Neumann part of the boundary. Recall from Section 1.1 that V_h is a subspace of $H_0^1(\Omega, \Gamma_D)$, which elements are restricted to vanish only on Dirichlet boundary. Thus using a zero extension interpolant at a Neumann boundary will not be accurate enough and introduces a correction with high energy (no longer $O(H)$), (see Fig. 3.3).

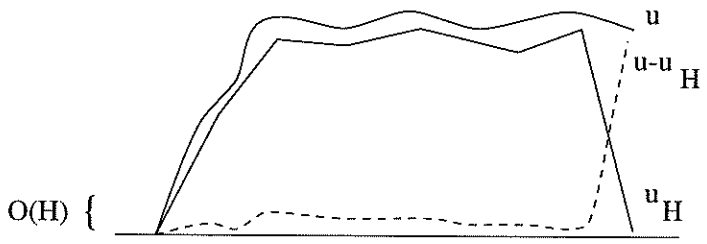


Figure 3.3: Non-matching boundaries: Zero extension interpolation is not accurate at the right end of the coarse domain.

To achieve better efficiency, we need to modify this intergrid operator to account for the Neumann condition. Two general ways to treat such boundaries:

1. Modify the coarse grid domain to cover any fine grid boundaries of Neumann type and use standard interpolation.

2. Increase the accuracy of the interpolants by accounting for the Neumann condition for those fine nodes in $\Omega \backslash \Omega^H$.

The first approach is motivated by the fact that standard nodal value interpolants can still be used with efficiency, provided the coarse grid covers the Neumann boundary part of the fine grid (see Fig. 3.4). This was first proposed and justified in [1]. We shall denote this operator as the coarse-to-fine interpolant, $\mathcal{I}_h^1$. Let us still denote the modified coarse grid domain by $\bar{\Omega}_H$. Then for all $v^H \in V^H$, the interpolant $\mathcal{I}_h^1$ is defined as:

$$\mathcal{I}_h^1 v^H(x_j^h) = \begin{cases} v^H(x_j^h) & \text{for } x_j^h \in \Omega \cap \bar{\Omega}_H, \\ 0 & \text{for } x_j^h \in \Omega \setminus \bar{\Omega}_H. \end{cases}$$

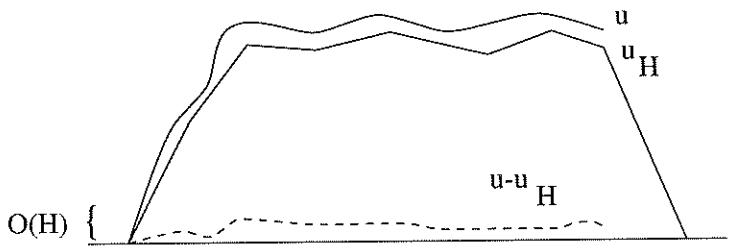


Figure 3.4: More accurate interpolation with $\mathcal{I}_h^1$ done at Neumann boundary.

This is a natural extension of v_H by zero outside the Dirichlet boundary part of the coarse grid domain. Similar zero extensions were used in Kornhuber-Yserentant [46] to embed an arbitrarily complicated domain into a square or cube in constructing multilevel methods on nested and quasi-uniform meshes for second order elliptic problems with purely Dirichlet boundary conditions.

Although the coarse-to-fine operator $\mathcal{I}_h^1$ works well for mixed boundary conditions, one has to modify the original coarse grid so that it covers the Neumann boundary part of the fine grid domain (see [3] for a description for modifying boundaries). This can be very difficult to do for complicated domains. To avoid modifying the original coarse grid, we now consider standard finite element interpolants which are modified only near Neumann boundaries. The idea is as follows: Let us consider a fine grid point, x, which lies outside the coarse grid domain. Find a nearby coarse grid triangle to x (say, τ_H with vertices x_1, x_2, x_3), and *extrapolate* $u(x)$ using the values $u(x_1), u(x_2)$ and $u(x_3)$. We will describe in more detail how this can be done in a couple of different ways. Note that such an extrapolation should depend on the type of boundary condition at x.

To do this, let us introduce some notation. Let τ_{lr}^H be any coarse boundary element in $\mathcal{T}_H$ which has an edge on the boundary $\partial\Omega_H$, denoted by $x_l^H x_r^H$. We use $\Omega(x_l^H, x_r^H)$ to denote the union of all fine elements, if any, which has a non-empty intersection with the unbounded domain formed by the edge $x_l^H x_r^H$ and two outward normal lines to $x_l^H x_r^H$ at two vertices x_l^H, x_r^H (cf. Fig. 3.5). By including a few more fine elements in some $\Omega(x_l^H, x_r^H)$, if necessary, we may assume that the fine grid part $(\Omega \setminus \Omega_H)$ is included in the union of all $\Omega(x_l^H, x_r^H)$. Moreover, we assume that the area of $\Omega(x_l^H, x_r^H)$ is bounded by the area of τ_{lr}^H:

(H1)

$$|\Omega(x_l^H, x_r^H)| \leq \mu\, |\tau_{lr}^H|,$$

where μ is a positive constant independent of H and h.

We remark that **(H1)** restricts the size of the fine grid part near the edge $x_l^H x_r^H$ but outside the coarse grid domain Ω^H, that is, each local fine grid part $\Omega(x_l^H, x_r^H)$ is not allowed to be too large compared to its nearest coarse element τ_{lr}^H. This is a reasonable requirement in applications.

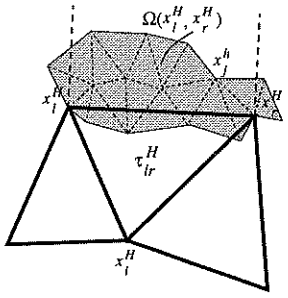


Figure 3.5: Shaded region, $\Omega(x_l^H, x_r^H)$, shows the fine grid part which is not completely covered by the coarse grid domain.

By analogy with the barycentric interpolation described in Fig. 3.1, we can now define a general nodal value interpolant near Neumann boundaries by using three given linear functions θ_1, θ_2 and θ_3 which are defined in $\bar{\Omega} \cup \bar{\Omega}_H$ but bounded in $\Omega(x_l^H, x_r^H) \cup \tau_{lr}^H$ and satisfying:

$$\theta_1(x) + \theta_2(x) + \theta_3(x) = 1, \quad \forall\, x \in \bar{\Omega} \cup \bar{\Omega}_H. \tag{3.1}$$

Then for any coarse function $v_H \in V_H$, we define an operator Θ_h by

$$\Theta_h v_H(x) = \theta_1(x) v_H(x_l^H) + \theta_2(x) v_H(x_r^H) + \theta_3(x) v_H(x_i^H), \quad \forall\, x \in \Omega(x_l^H, x_r^H) \cup \tau_{lr}^H,$$

and assume that

(H2) $$\Theta_h v_H = v_H \quad \text{on the edge } \; x_l^H x_r^H,$$

which means $\Theta_h v_H$ is indeed an extension of v_H.

With the above notation, we can now introduce the general coarse-to-fine interpolant $\mathcal{I}_h$:

Definition 3.1 *For any coarse function $v_H \in V_H$, its image under the coarse-to-fine interpolant $\mathcal{I}_h$ is specified as follows:*

(C1) *For any fine node x_j^h in $\bar{\Omega} \cap \bar{\Omega}_H$,*

$$\mathcal{I}_h v^H(x_j^h) = v^H(x_j^h);$$

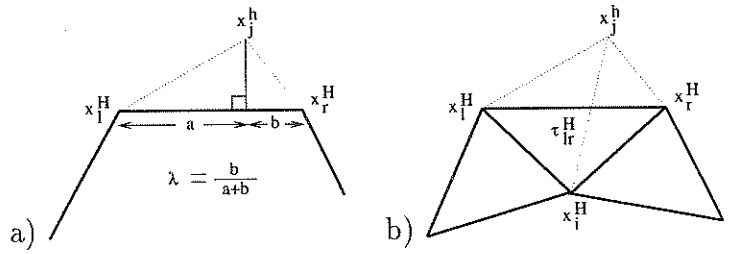


Figure 3.6: **More accurate interpolants:** a) $\mathcal{I}_h^2$: Fine nodal values outside the coarse domain are interpolated with coarse nodal values on the nearest coarse grid edge; b) $\mathcal{I}_h^3$: Fine nodal values outside the coarse domain are interpolated with nodal values on the nearest coarse element τ_{lr}^H. Thick lines represent coarse grid boundaries or elements, and dotted lines show the coarse nodes used to interpolate fine nodal value at x_j^h.

(C2) *For any fine node x_j^h in $\Omega(x_l^H, x_r^H) \setminus \bar{\Omega}_H$ with both x_l^H and x_r^H Neumann nodes,*

$$\mathcal{I}_h v^H(x_j^h) = \Theta_h v_H(x_j^h);$$

(C3) *For any fine node x_j^h in $\Omega(x_l^H, x_r^H) \setminus \bar{\Omega}_H$ with both x_l^H and x_r^H Dirichlet nodes,*

$$\mathcal{I}_h v^H(x_j^h) = 0;$$

(C4) *For any fine node x_j^h in $\Omega(x_l^H, x_r^H) \setminus \bar{\Omega}_H$ with one of x_l^H and x_r^H a Neumann node and the other a Dirichlet node,*

$$\begin{aligned} \mathcal{I}_h v^H(x_j^h) &= 0, \quad \text{if } x_j^h \text{ is a fine boundary node of Dirichlet type;} \\ \mathcal{I}_h v^H(x_j^h) &= \Theta_h v_H(x_j^h), \quad \text{otherwise.} \end{aligned}$$

The following are two specific examples of interpolants which satisfy the above definition and assumptions. We only give the corresponding forms of Θ_h's required in the definition.

We define the interpolant at x_j^h by using the nodes of the coarse boundary edge closest to x_j^h (see Fig. 3.6):

$$\mathcal{I}_h^2 v^H(x_j^h) = \lambda(x_j^h) v^H(x_l^H) + (1 - \lambda(x_j^h)) v^H(x_r^H),$$

where λ is the ratio of the lengths of two segments of the edge $x_l^H x_r^H$ cut off by the normal line passing through x_j^h to the edge (see Fig. 3.6). This kind of interpolation was also used by Bank and Xu [47] in their construction of a hierarchical basis on a unstructured mesh.

We define a non-zero extension by extrapolation using barycentric functions (see Fig. 3.6):

$$\mathcal{I}_h^3 v^H(x_j^h) = \lambda_l(x_j^h) v^H(x_l^H) + \lambda_r(x_j^h) v^H(x_r^H) + \lambda_i(x_j^h) v^H(x_i^H),$$

where $\lambda_l, \lambda_r, \lambda_i$ are the three barycentric coordinate functions corresponding to τ_{lr}^H.

REMARK. Note that the functions λ_l, λ_r and λ_i used in the definition of $\mathcal{I}_h^3$ satisfies $\lambda_l, \lambda_r, \lambda_i \geq 0$ for $x_j^h \in \tau_{lr}^H$, but not so for $x_j^h \notin \tau_{lr}^H$. The barycentric coordinates may still be defined, provided we consider the area of a simplex to be orientation-dependent. That is, area is > 0 for "right-handed" triangles (clockwise) and area is < 0 for "left-handed" triangles (counter-clockwise). In the case as shown in Fig. 3.6b, we have $x_j^h \notin \tau_{lr}^H$, $\lambda_l(x) \geq 0, \lambda_r(x) \geq 0$, but $\lambda_i(x) \leq 0$ and still $\lambda_l(x) + \lambda_r(x) + \lambda_i(x) = 1$. By **(H1)**, we always have

$$|\lambda_l(x)| \leq \mu_1, \quad |\lambda_r(x)| \leq \mu_1, \quad \text{and} \quad |\lambda_i(x)| \leq \mu_1, \quad \forall\, x \in \Omega(x_l^H, x_r^H) \cup \tau_{lr}^H,$$

where μ_1 is a constant independent of h and H but depending only on the constant μ in **(H1)**.

We summarize the various interpolants:

$\mathcal{I}_h^0$: Zero extension with unmodified coarse boundaries,

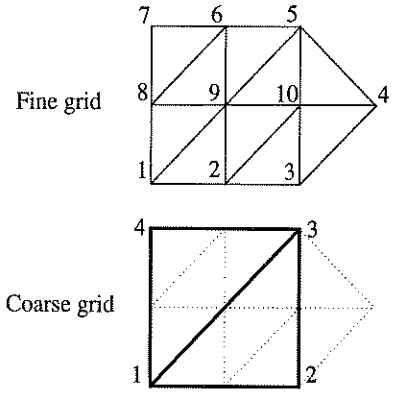
$\mathcal{I}_h^1$: Zero extension with modified coarse Neumann boundaries,

$\mathcal{I}_h^2$: Nearest edge interpolation, and

$\mathcal{I}_h^3$: Nearest element interpolation.

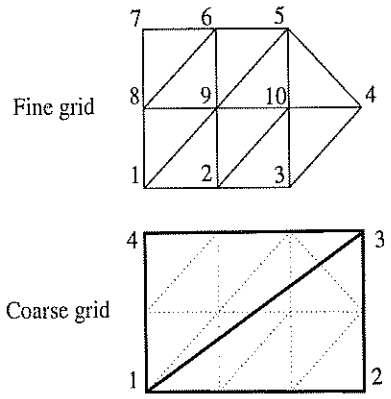
3.3.1 A simple example

To illustrate the differences among the four different interpolants $(\mathcal{I}_h^0, \mathcal{I}_h^1, \mathcal{I}_h^2, \mathcal{I}_h^3)$, let us consider a simple fine grid with 10 nodes, and a 4-noded MIS coarse grid in which the coarse grid boundary does not match the fine grid boundary. In particular, the MIS coarse grid that is automatically generated will not contain the fine grid node, v_4 (see, e.g. Fig. 3.7). The coarse grids and the corresponding interpolants are shown in Figs. 3.7–3.10 below:



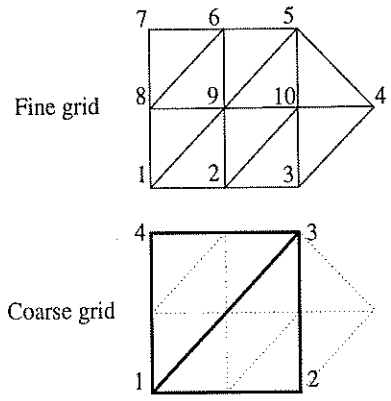
$$\mathcal{I}_h^0 = \begin{bmatrix} 1.0 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 \\ 0 & 1.0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 1.0 \\ 0.5 & 0 & 0 & 0.5 \\ 0.5 & 0 & 0.5 & 0 \\ 0 & 0.5 & 0.5 & 0 \end{bmatrix} \Leftarrow$$

Figure 3.7: Fine grid and MIS coarse grid and the corresponding linear interpolant, $\mathcal{I}_h^0$.



$$\mathcal{I}_h^1 = \begin{bmatrix} 1.0000 & 0 & 0 & 0 \\ 0.6667 & 0.3333 & 0 & 0 \\ 0.3333 & 0.6667 & 0 & 0 \\ 0 & 0.5000 & 0.5000 & 0 \\ 0 & 0 & 0.6667 & 0.3333 \\ 0 & 0 & 0.3333 & 0.6667 \\ 0 & 0 & 0 & 1.0000 \\ 0.5000 & 0 & 0 & 0.5000 \\ 0.5000 & 0 & 0.3333 & 0.1667 \\ 0.3333 & 0.1667 & 0.5000 & 0 \end{bmatrix} \Leftarrow$$

Figure 3.8: Fine grid and modified coarse grid and the corresponding linear interpolant, $\mathcal{I}_h^1$.

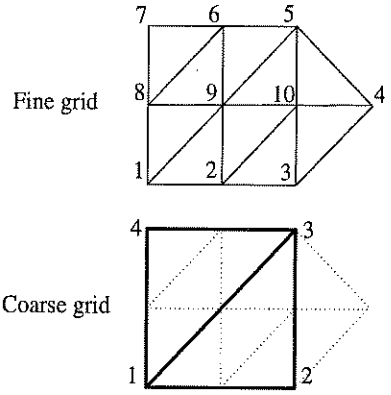


$$\mathcal{I}_h^2 = \begin{bmatrix} 1.0 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 \\ 0 & 1.0 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 1.0 \\ 0.5 & 0 & 0 & 0.5 \\ 0.5 & 0 & 0.5 & 0 \\ 0 & 0.5 & 0.5 & 0 \end{bmatrix} \Leftarrow$$

Figure 3.9: Fine grid and MIS coarse grid and the corresponding linear interpolant, $\mathcal{I}_h^2$.

Note that $\mathcal{I}_h^0, \mathcal{I}_h^2$, and $\mathcal{I}_h^3$ are identical except for the way that the value at fine grid node, v_4, is interpolated (row 4). Interpolant $\mathcal{I}_h^0$ results in a zero value extension, interpolant $\mathcal{I}_h^2$ extends linearly in the normal direction with the nearest coarse boundary edge, and interpolant $\mathcal{I}_h^3$ extends by barycentric coordinates of the nearest coarse element (note here that the entries are not necessarily non-negative, but still sum to one). Because all the coarse boundary nodes are also fine boundary nodes, the definition of the coarse boundary conditions can simply be taken to be the same as that of the corresponding fine boundary nodes.

Interpolant $\mathcal{I}_h^1$ differs in that the coarse grid domain was modified to contain the fine grid domain. As a result, the grids are not element nested and some of the coarse boundary nodes are not nodes from the fine grid. Standard linear interpolation is done everywhere in this case with no special extensions required. However, choosing the proper boundary conditions for the coarse boundary points which are not in the fine grid must be determined somehow (see Def. 3.1).



$$\mathcal{I}_h^3 = \begin{bmatrix} 1.0 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 \\ 0 & 1.0 & 0 & 0 \\ -0.5 & 1.0 & 0.5 & 0 \\ 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 1.0 \\ 0.5 & 0 & 0 & 0.5 \\ 0.5 & 0 & 0.5 & 0 \\ 0 & 0.5 & 0.5 & 0 \end{bmatrix} \Leftarrow$$

Figure 3.10: Fine grid and MIS coarse grid and the corresponding linear interpolant, $\mathcal{I}_h^3$.

3.4 Stability and approximation of the non-nested interpolation

The convergence theory for overlapping multilevel domain decomposition and multigrid methods require the coarse-to-fine grid transfer operator to possess the local optimal L^2-approximation and local H^1-stability properties as introduced in Sec. 2. The locality of these properties is essential to the effectiveness of these methods on highly non-quasi-uniform unstructured meshes.

Because the spaces are non-nested (they are *node-nested*, but still *non-nested*, as coarse grid elements are not unions of fine grid elements), in the theory discussed in Section 2, the u_H coarse space approximation to u should be defined as:

$$u_H = \mathcal{I}_h \mathcal{R}_H u.$$

Since $\mathcal{R}_H \in V_H \not\subset V_h$, we need to use the interpolation operator $\mathcal{I}_h$ to map $\mathcal{R}_H u$ back to V_h. The convergence theory now requires both $\mathcal{I}_h$ and $\mathcal{R}_H$ to possess the stability and approximation properties. When the mesh is quasi-uniform, the usual L^2-projection, Q_H, can be used for $\mathcal{R}_H$. But when the mesh is highly non-quasi-uniform, the constant in the approximation property (2.12) can deteriorate if we use Q_H. The trick then is to use a localized version of the L^2-projection, i.e. the so-called Clément's projection. It is known that this projection provides local stable and good approximations. We refer to Clément [48] for its definition and Chan-Zou [44] and Chan-Smith-Zou [1] for its use in domain decomposition contexts.

For $\mathcal{I}_h$, we can take the coarse-to-fine interpolants introduced in Def. 3.1 The key step then is proving the stability and approximation properties for $\mathcal{I}_h$. The proof that the non-nested standard interpolation used in the interior is stable and accurate can be found in Cai [49] and Chan-Smith-Zou [1]. The proof for the boundary-specific interpolations can be found in [4].

3.5 Numerical results

In this section, we provide some numerical results of domain decomposition and multigrid methods for elliptic problems on several unstructured meshes: see Figure 3.11. The well-known NASA airfoil mesh was provided by T. Barth and D. Jesperson of NASA Ames, and a fine, unstructured square and annulus were generated using Barth's 2-dimensional Delaunay triangulator. All numerical experiments were performed using the Portable, Extensible Toolkit for Scientific Computation (PETSc) [50], running on a Sun SPARC 20. Piecewise linear finite elements were used for the discretizations and the resulting linear system was solved using either multilevel overlapping Schwarz or V-cycle multigrid as a preconditioner with full GMRES as an outer accelerator.

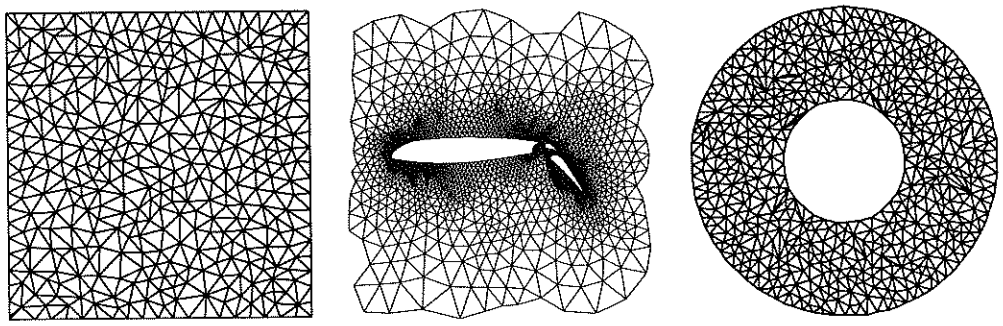


Figure 3.11: Some fine grids: an unstructured square with 385 nodes (left), NASA airfoil with 4253 nodes (center) and an annulus with 610 nodes (right).

We shall present numerical results for Schwarz solvers and multigrid methods. For partitioning, all the domains (except the coarsest) were partitioned using the recursive spectral bisection method [51], with exact solves for both the subdomain problems and the coarse grid problem. To generate overlapping subdomains, we first partition the domain into nonoverlapping subdomains and then extend each subdomain by some number of elements.

In all the experiments, the initial iterate is set to be zero and the iteration is stopped when the discrete norm of the residual is reduced by a factor of 10^{-5}.

For our first experiment, we use additive Schwarz to solve the Poisson problem on a unit square with homogeneous Dirichlet boundary conditions. Because the fine domain is so simple and Dirichlet boundary conditions are given, non-matching boundaries are not an issue here and no special interpolants are used. Table 1 shows the number of GMRES iterations to convergence with varying fine grid problem and varying number of levels. Providing a coarse grid greatly improved convergence, and without it the method is not scalable to larger problems. Interesting things to notice are that for a fixed number of levels, multilevel Schwarz is mesh-size independent, but that the number of iterations increases with the number of levels for a fixed problem size. This had also been previously observed for structured meshes using a multilevel diagonal scaling method in [22] and is due to the additive nature of the method. Also, increasing the amount of overlap improved convergence, but in practice, a one-element overlap was sufficient.

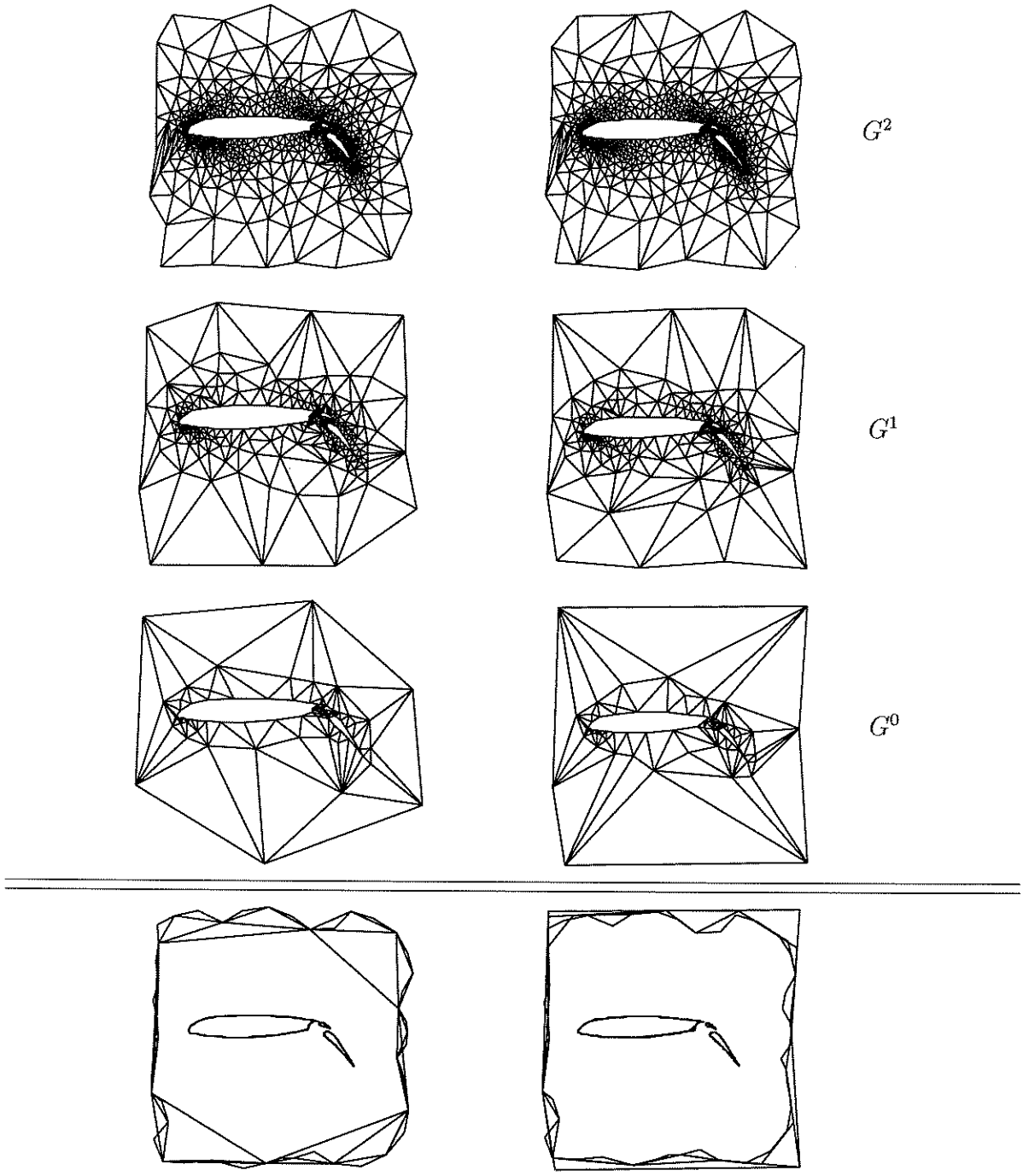


Figure 3.12: *Airfoil* grid hierarchy with unmodified boundaries (left) and modified boundaries (right).

Table 1: **Additive multilevel Schwarz** iterations for the Poisson problem on a unit square grid. All grids (except coarsest) were partitioned using RSB. Tables show the number of GMRES iterations to convergence.

Dirichlet boundary conditions

# of levels	# of nodes	# of subdomains	# overlap elements 0	1	2
1	6409	256	84	63	50
	1522	64	45	36	27
	385	16	26	19	16
2	1522	64	19	16	16
	385	1			
	385	16	19	15	15
	102	1			
	102	4	17	15	15
	29	1			
3	6409	256	28	24	25
	1522	64			
	385	1			
	1522	64	32	25	26
	385	16			
	102	1			
	385	16	31	26	26
	102	4			
	29	1			
4	6409	256	43	37	37
	1522	64			
	385	16			
	102	1			
	1522	64	42	37	37
	385	16			
	102	4			
	29	1			

In our second experiment, we solve a mildly varying coefficient problem on the airfoil:

$$\frac{\partial}{\partial x}((1+xy)\frac{\partial u}{\partial x}) + \frac{\partial}{\partial y}((\sin(3y))\frac{\partial u}{\partial y}) = (4xy+2)\sin(3y) + 9x^2\cos(6y)$$

with either a purely Dirichlet boundary condition or a mixed boundary condition: Dirichlet for $x \leq 0.2$ and homogeneous Neumann for $x > 0.2$. For this problem, the non-homogeneous Dirichlet condition is $u = 2 + x^2\sin(3y)$. Table 3.13 shows the number of GMRES iterations to convergence using additive multilevel Schwarz with the different boundary treatments. We see the deterioration in the method when Neumann conditions

are not properly handled.

In Table 3.14, we show results for the same problem, but solved using a hybrid multiplicative-additive Schwarz (multiplicative between levels but additive among subdomains on the same level). As in the additive case, deterioration of the method occurs when mixed boundary conditions are present. It can be seen that the multiplicative method (both on the subdomains and between levels) behaves much like multigrid (see Fig. 3.15 and Table 2). In fact, this is nothing more than standard V-cycle multigrid with a block smoother, used as a preconditioner. A V-cycle multigrid method with pointwise Gauss-Seidel smoothing and 2 pre- and 2 post-smoothings per level was used to produce the results in Table 2.

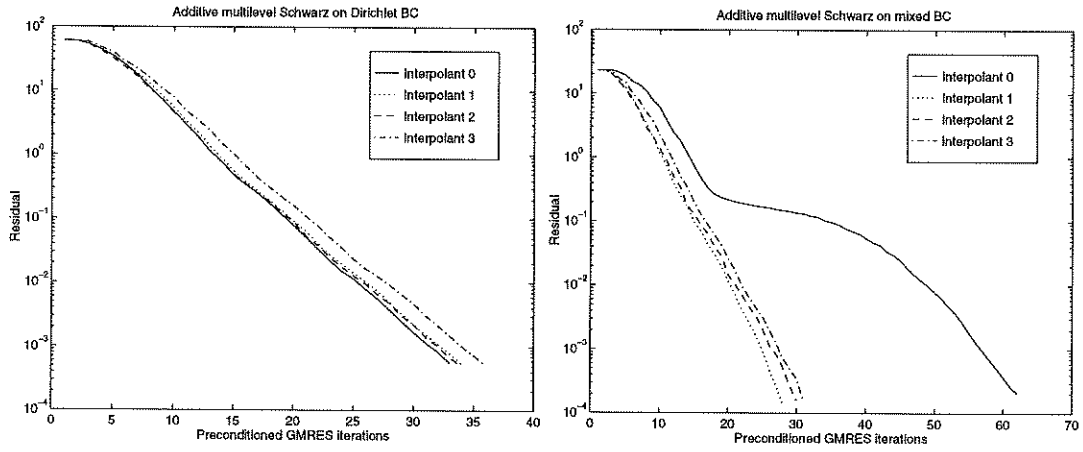


Figure 3.13: **Additive 4-level Schwarz** iterations for the elliptic problem with mildly varying coefficients on the *airfoil* grid (G^3) with 4253 unknowns. All grids (except coarsest) were partitioned using RSB, with 1 element overlap.

Table 3 shows some multigrid results for the Poisson equation on an annulus. The forcing function is set to be one and both kinds of boundary conditions were tested. A V-cycle multigrid method with pointwise Gauss-Seidel smoothing and 2 pre- and 2 post-smoothings per level was used. When mixed boundary conditions are present, the deterioration in using the less accurate interpolation $\mathcal{I}_h^0$ is less pronounced in the multigrid method than in the Schwarz methods, but it still exists.

3.6 Concluding remarks

When using general unstructured meshes, the coarse grid domain may not necessarily match that of the fine grid. For the parts of the fine grid domain which are not contained in the coarse domain, special treatments must be done to handle different boundary conditions. The transfer operators using linear interpolation with a zero extension is the most natural to implement and is effective for problems with Dirichlet boundary conditions.

For problems where Neumann boundary conditions exist however, zero extension is no longer appropriate and special interpolants should be sought. Our numerical results

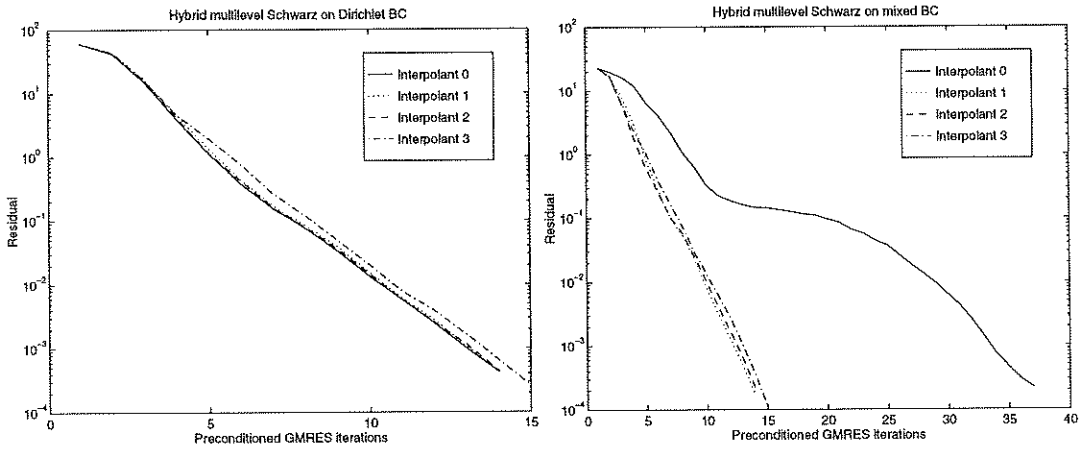


Figure 3.14: **Hybrid multiplicative-additive 4-level Schwarz** iterations for the elliptic problem with mildly varying coefficients on the *airfoil* grid (G^3) with 4253 unknowns. All grids (except coarsest) were partitioned using RSB, with 1 element overlap.

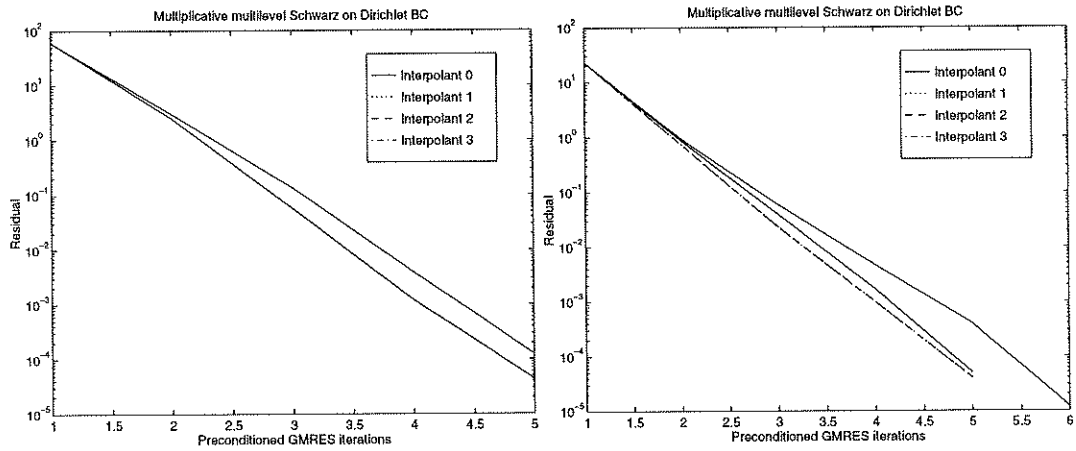


Figure 3.15: **Multiplicative 4-level Schwarz** iterations for the elliptic problem with mildly varying coefficients on the *airfoil* grid (G^3) with 4253 unknowns. All grids (except coarsest) were partitioned using RSB, with 1 element overlap.

show the significance of the assumption that when standard interpolations with zero extension are used, the coarse grid must cover the Neumann boundaries of the fine grid problem, otherwise deterioration of the methods indeed occurs. The deterioration is most significant when using additive multilevel methods, but can still be seen for the multiplicative methods. When coupled with highly stretched elements, the deterioration can be very significant, even for multiplicative methods.

Although modifying the coarse grid domains to ensure that this assumption is satisfied is effective, this approach can be problematic to implement for particularly complicated domains or can sometimes generate coarse grid domains which deviate significantly from the fine domain. In addition, the modified coarse boundaries will no longer

Table 2: **Multigrid** iterations for the elliptic problem with mildly varying coefficients on the *airfoil*. Tables show the number of GMRES iterations to convergence.

Dirichlet boundary conditions

# of fine grid nodes	MG levels	# of coarse grid nodes	Interpolant Used			
			$\mathcal{I}_h^0$	$\mathcal{I}_h^1$	$\mathcal{I}_h^2$	$\mathcal{I}_h^3$
4253	2	1170	4	4	4	4
	3	340	4	4	4	4
	4	101	4	4	4	4

Mixed Dirichlet/Neumann boundary conditions

# of fine grid nodes	MG levels	# of coarse grid nodes	Interpolant Used			
			$\mathcal{I}_h^0$	$\mathcal{I}_h^1$	$\mathcal{I}_h^2$	$\mathcal{I}_h^3$
4253	2	1170	6	5	4	4
	3	340	6	4	5	5
	4	101	7	5	5	5

Table 3: **Multigrid** iterations for the Poisson problem on an *annulus*. The exit condition was decreased to 10^{-6} from 10^{-5}. Tables show the number of GMRES iterations to convergence.

Dirichlet boundary conditions

# of fine grid nodes	MG levels	# of coarse grid nodes	Interpolant Used			
			$\mathcal{I}_h^0$	$\mathcal{I}_h^1$	$\mathcal{I}_h^2$	$\mathcal{I}_h^3$
2430	2	610	4	4	4	4
	3	160	4	4	4	4
	4	47	4	4	4	4

Mixed Dirichlet/Neumann boundary conditions

# of fine grid nodes	MG levels	# of coarse grid nodes	Interpolant Used			
			$\mathcal{I}_h^0$	$\mathcal{I}_h^1$	$\mathcal{I}_h^2$	$\mathcal{I}_h^3$
2430	2	610	6	5	4	4
	3	160	7	5	4	4
	4	47	7	5	4	4

be node-nested to the fine boundaries, so some way of defining boundary conditions on those nodes must be made.

An alternative is to modify the interpolants so that non-zero extensions be used on those fine grid boundaries which have Neumann conditions and which are not contained within the coarse grid domain. Since we are using the multilevel methods only as pre-conditioners, the extension need not be particularly accurate.

4 Agglomerated coarse spaces

4.1 Agglomerated multigrid methods on unstructured grids

In this section, we will consider a general approach for constructing nested coarse spaces and transfer operators. The difference between this technique and node-nested coarse spaces from the previous section is that here we want to produce a nested sequence of spaces to be used in the multigrid method. The common point will be that our construction must satisfy the approximation and stability properties mentioned in the subspace correction framework, section 2.

4.2 Coarse points and construction of macroelements

The agglomeration technique is based on the construction of a coarse grid with "macro-elements" consisting of unions of fine grid elements (triangles). An example of such a coarse grid is given on Fig. 4.1. Then as in the standard finite element method, the basis functions in each coarse grid macroelement are appropriately defined. The coarse space V_H is then determined as the space spanned by these functions. If the coarse grid basis functions are defined as linear combinations of fine grid basis (i.e. the usual finite element basis), then V_H is a proper subspace of V_h, i.e. we obtain nested spaces by construction.

The construction of a basis in V_H is equivalent to the definition of the restriction matrix R_H, because the coordinates of these basis functions with respect to the fine grid basis form the rows of the restriction matrix. Thus, the basis defined, we have the restriction R_H, the interpolation (or prolongation) R_H^T, the coarse grid operator $R_H A R_H^T$ and we can apply the V-cycle algorithm from section 1.4.1.

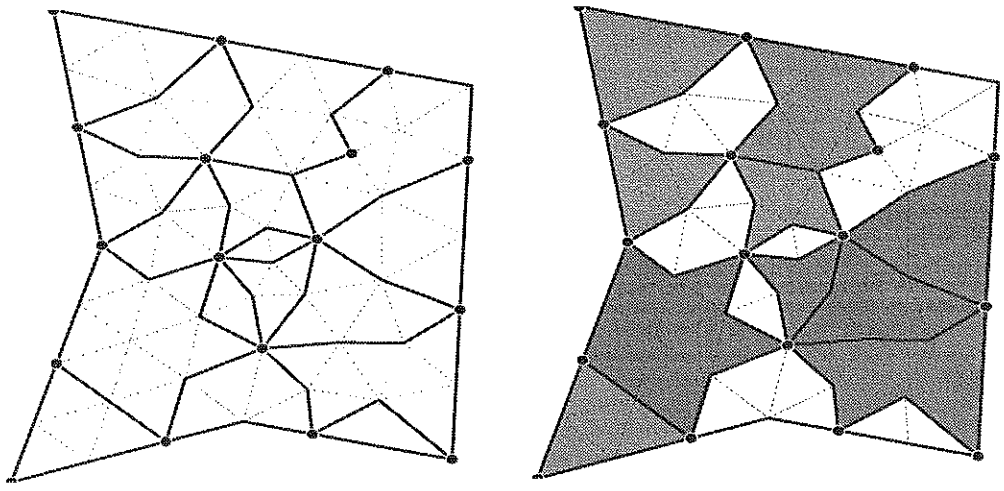


Figure 4.1: An example of macroelements

Although the nestedness is assured by construction, there are some important rules

which have to be followed in order to ensure good convergence rate of the resulting multilevel methods. These rules are summarized below:

- *Smoothness:* The basis functions have to be smooth enough. This requirement is needed because the elements from V_H have to satisfy the stability property (2.11), which involves the A-norm of the coarse grid function.

- *Approximation:* The functions in V_H have to satisfy the approximation property (2.12). An implication of this is that the individual basis function in V_H cannot be independently chosen, and there must be some global relation which couples the basis functions.

- *Small supports:* The functions in V_H must have compact support. This requirement is based on the fact that once the basis $\{\Phi_i\}$ in V_H is given, then the coarse grid matrix elements are defined to be $a(\Phi_i, \Phi_j)$ (see (1.4)). Thus, if the basis functions have large supports, the coarse grid matrix will be dense, and the coarse grid problem is expensive to solve.

- *Conformity:* For finite element discretizations it is desirable that the resulting coarse grid is formed by conformal macroelements, an analogue of conforming triangulations in finite element methods. This facilitates the analysis and constructions of the algorithms.

- *Recursion:* The coarse grid should have the same properties as the fine grid, allowing the recursive application of the algorithm to construct a multilevel method.

A careful look at these rules shows that it is difficult (even impossible) to satisfy all of them simultaneously. Usually some of them have to be weakened in order to satisfy others. For example, to have conforming macroelements of size $\approx 2h$ on an unstructured grid (which is a desirable choice in multigrid) is almost impossible. Taking smoother basis functions will increase the supports, and make the coarse grid operator more dense.

In this section, we will present algorithms for definition the coarse grid and constructing of the coarse basis functions on it. We will use a MIS (see section 3.1) for the coarse grid nodes.

A recent paper by Koobus, Lallemand and Dervieux [37] deals with agglomeration with the finite volume discretizations. The basis functions are piecewise constant on each cell, and the coarse grid cells are formed as unions of fine grid cells. A drawback of this algorithm is that the stability of the coarse grid basis functions is not easy to control.

An algebraic agglomeration algorithm can be found in the recent papers by Mandel, Vaněk, Brezina (see [38]) and Vaněk, Křížková [39]. This approach uses an algebraically smoothed basis functions, and the coarse grid nodes are not explicitly defined. This allows the process of the basis construction more automatic, but it is more difficult to control the sparsity of the coarse grid operators.

Our approach is based first on the definition of the coarse grid points (MIS) and then using them to define the macroelements. We will first explain our algorithm for grouping

the triangles on the fine grid using the MIS, following as closely as possible the rules given at the beginning of this section.

Let us consider a triangulation of Ω as the one on Fig. 4.2. To this triangulation corresponds a graph $G = (V, E)$, where V is the set of vertices (grid nodes) and E is the set of edges (boundaries of the triangles). We want to group neighboring elements on this grid together. This is equivalent to grouping together vertices in the so-called dual graph of G. The dual graph for this triangulation is drawn on Fig. 4.2 with dashed line. For our presentation, it will be enough to keep in mind that to each triangle in the triangulation corresponds a vertex in the dual graph, and to each edge in the triangulation corresponds an edge in the dual. Taking vertices in the dual together and separating the formed group from the rest is equivalent to removing some of the edges in the dual graph. In other words, if we form a macroelement by grouping together some neighboring triangles and consider its boundary, all the edges in the *dual* graph corresponding to this boundary are removed. In summary, forming the macroelement boundary is equivalent to removing some of the edges in the dual.

The algorithm, which is given later is fully based on this observation. Roughly it works in the following way:

ALGORITHM SKETCH:

- Remove some of the edges in the dual, selected by a reasonable criteria.
- Find the *connected components* in the dual graph. These connected components correspond to macroelements in G, see Fig. 4.3.

By *connected component* in the above algorithm, we mean a sub-graph, such that there is a *path* between every two vertices in it. A path between two vertices is any collection of edges connecting these two vertices.

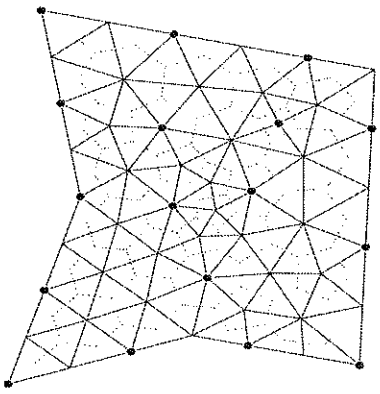
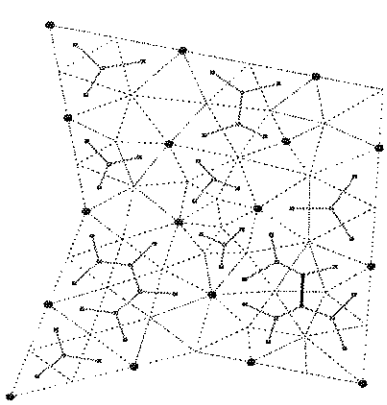


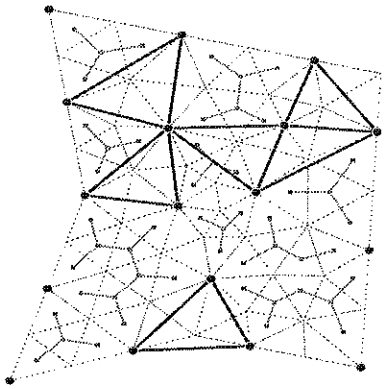
Figure 4.2: A triangulation and its dual

First we will try to give an intuitive idea of what we meant by the reasonable criteria mentioned in the algorithm sketch above.

Let us look at the rules we give at the beginning of this section. We will focus on the relationship between the macroelements and the supports of the basis functions. Clearly, each support will be a union of macroelements. The situation here is very similar to the finite element discretization on the fine grid, except that the elements have different shape. The nonzeroes in the coarse grid operator appear whenever we have $a(\Phi_i, \Phi_j) \neq 0$, i.e. whenever a point of the fine grid belongs to both the supports $supp(\Phi_i)$ and $supp(\Phi_j)$. If we now reverse this point of view and look at the fine grid nodes (i.e. the nodes not in the MIS), we see that each such node will create a number of nonzeroes exactly equal to the number of supports it belongs to. But because of the approximation requirement, a fine grid node should belong to as many supports as possible, to assure accurate data transfer from coarse to fine grid. This conflict can be avoided if we take into account that, if the fine grid node lies on the macroelement boundary it will belong to only two supports. Consequently the nodes interior to the macroelements would create more non-zeroes in the coarse grid operator. Our goal in accordance with this observation will be to create as many macroelement boundaries as we can and in addition to keep the size of macroelements reasonably small.



Connected components.



Connected components of the dual and the fictitious macroelements.

Figure 4.3:

Because of the conformity requirement, we want to avoid more than two macroelement boundaries intersecting each other only at coarse grid nodes. Thus, through each coarse grid node we will try to pass a maximum number of macroelement boundaries. The correspondence between the dual graph and the triangulation suggests the following algorithm.

- **For** all coarse grid points **do**:
- Take all the triangles attached to a coarse grid point.
- Remove the edges in the dual between these triangles.

- **endFor.**
- Proceed with the second step of the ALGORITHM SKETCH.

Before we present the algorithm in detail, we want to focus our attention on two more issues. The first one is that such an algorithm might produce large macroelements. To avoid this, we break in pieces any of these large elements. Another issue is that removing the edges might create isolated vertices in the dual (isolated triangles). The approximation requirement might be unsatisfiable if a fine grid node (not in MIS) is surrounded by such triangles. We avoid this by creating a "fictitious" macroelement with the closest coarse grid nodes as vertices, see Fig. 4.4

The new graph for the recursive call of the same algorithm is prepared by cutting each macroelement with more than three coarse nodes as vertices into triangles; see Fig. 4.4.

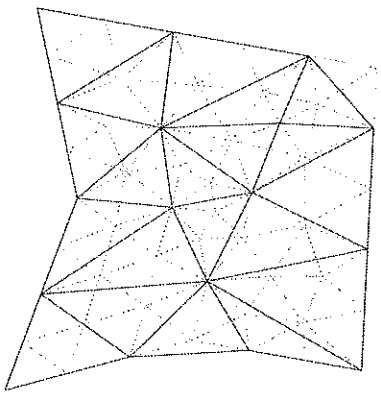


Figure 4.4: G_{next}

The detailed algorithm is given below:

ALGORITHM 4.1 *(Macroelement construction)*

Step 0: *Consider the graph corresponding to a triangulation in the plane:* $G = (V, E)$, *and its dual* $G^* = (V^*, E^*)$.*(see Fig.4.2).*

Step 1: *Coarse nodes. Choose a maximal independent set (coarse nodes)* $M(G)$ *in* G. *Designate as* blocked triangles, *those triangles which have a coarse node as a vertex. Call the other triangles* free triangles.

Step 2: *Initial dual splitting. Remove the edges in the dual of the form* blocked—blocked *and obtain the new graph* $G_1^* = (V^*, E_1^*)$.

Step 3: *Domain splitting. Find the connected components in* G_1^*. *The corresponding triangles in* G *are called* macroelements.

Step 3*: *Refined splitting of "large" macroelements. During* **Step 3**, *after forming each macroelement, we pick all the edges from* $G \cap$ macroelement *which are not incident with the boundary of that macroelement. Then a maximum cardinality matching [52] in this set is found, and the corresponding edges in* G_1^* *are removed. Thus, every macroelement in which we are able to find a non-boundary maximum cardinality matching is divided in pieces.*

Step 4: *Next graph* $G_{next} = (M(G), E_{next})$. *The next graph for* multilevel coarsening *we is defined as follows:*

- *Macroelements obtained from step* 3 *and* 3* *are faces in this graph.*
- *Additional faces are obtained considering all the vertices which are NOT in these macroelements, and are NOT coarse points. These points are either surrounded by triangles, which are not visited during* Step 3, *or they lie on the boundaries of the macroelements. For each such node i, we consider the neighboring vertices from* $M(G)$. *If the number of such neighbors is greater than TWO we put a fictitious face there (i.e. node i might be viewed as an internal node for this face). (see Fig.4.3)*

It can be verified that, the above algorithm fully recovers the coarse structure of a "structured", topologically rectangular mesh. If the grid is unstructured, but is obtained by regular refinement, and the maximal independent set coincides with the real coarse grid vertices, then the macroelements will form exactly the underlying coarse grid.

4.3 Coarse space basis functions.

Given the set of macroelements, we will now introduce three different ways to define the coarse grid basis functions. Recall that we want to meet the first two rules given in the previous section. We need to define smooth basis functions so that the coarse space satisfy the approximation and stability properties.

To assure the approximation property, we have to take a basis which preserves at least the constant function, i.e. the constant function must be always in the coarse space V_H. To do this, we first define basis functions possessing this property on the the macroelement boundaries, and after that we extend them into the interior of the macroelement as discrete harmonic functions. This extension obviously will not destroy the constant preserving property, because the constant function is harmonic.

We define the coarse basis functions on these edges as linear functions minimizing some quadratic functional. The $H^{1/2}$ norm on the boundary is one good choice for the quadratic functional, as it is the interface analogue of the A-norm.

Let the macroelement boundary be formed by ℓ edges from the fine grid, (see Fig 4.5) connecting two coarse grid points, x_0 and x_ℓ, and let this path contains the vertices $x_0, \ldots, x_\ell$. We define the basis function corresponding to the coarse grid node x_0 as follows: Φ_0 is a linear function in $\mathbb{R}^2$:

$$\Phi_0 = a\xi + b\zeta + c, \quad (\xi, \zeta) \in \mathbb{R}^2.$$

We want : $\Phi_0(x_0) = 1$, $\Phi_0(x_\ell) = 0$. These are only two conditions and we have three parameters. To complete the set of conditions, we require that the function Φ_0 minimizes the functional (discrete $H^{1/2}$ norm):

$$F_{1/2}(\Phi_0) = \sum_{i=1}^{\ell} \sum_{j=i+1}^{\ell} \frac{h_i h_j}{h_{ij}^2} (\Phi_0(x_i) - \Phi_0(x_j))^2, \tag{4.1}$$

where h_i is the length of the edge (x_i, x_{i+1}) and $h_{ij} = |x_i - x_j|$. After using the first two conditions, this minimization is equivalent to minimization of a simple quadratic function of one variable which can be easily proved analytically.

In the interior of the macroelements, we extend the basis functions by solving the equation:

$$a(\Phi_0, \phi) = 0, \qquad \text{for any} \quad \phi \in V_h. \tag{4.2}$$

In this way, we define the basis in V_H and thus also defining the restriction operator R_H.

ALGORITHM 4.2 *(Geometric coarsening-Harmonic extension I)*

1. *The boundary values minimizing the $H^{1/2}$-discrete norm.*
2. *For the nodes internal to the macroelements, the values of the basis functions are obtained via the harmonic extension, see Fig. 4.5.*

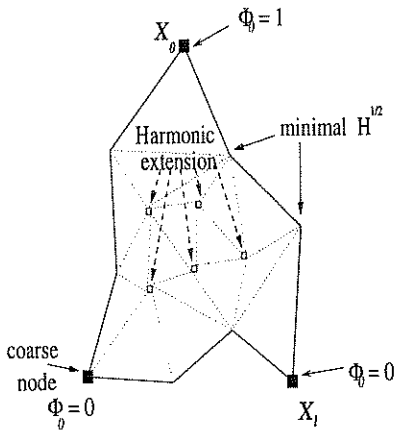


Figure 4.5: Minimizing $H^{1/2}$ on the boundary and harmonic extension in the interior

We will now give two simpler variants of this algorithm. The first one uses simpler boundary conditions: The function is defined on the boundary using the graph distance (see Fig 4.6). The graph distance $dist(i, j)$ is equal to the number of edges forming the shortest path connecting the vertices i and j, (in the case we consider these vertices are x_0 and x_ℓ).

ALGORITHM 4.3 *(Geometric coarsening-Harmonic extension II)*

1. *The boundary values are taken using the graph distance interpolation, see Fig. 4.6.*
2. *For the nodes internal to the macroelements, the values of the basis functions are obtained via the harmonic extension, see Fig. 4.5.*

The second variant, which does not include harmonic extension is as follows:

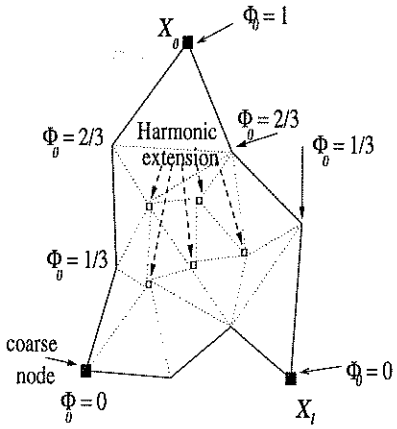


Figure 4.6: Graph distance on the boundary and smooth extension in the interior

ALGORITHM 4.4 *(Geometric coarsening)*

1. *On the macroelement boundaries the graph distance interpolation is used.*
2. *For the nodes internal to the macroelements and fictitious faces (see* Step 4*), the values of all basis functions whose supports form the macroelement take one and the same value: the reciprocal of the number of coarse points forming the macroelement, see Fig. 4.7.*

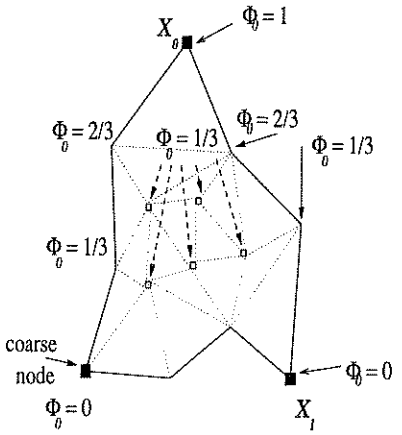


Figure 4.7: Graph distance on the boundary and in the interior. c denote the coarse grid points

The approximation and stability properties of the agglomerated spaces given above are assured by the next lemma. A detailed proof will be included in the paper [5].

Given a triangulation $\mathcal{T}_h$ and the corresponding linear finite element space $V_h \subset H^1(\Omega)$. Let $V_H \subset V_h$ be obtained by the agglomeration algorithm described in the above section. Let $Q_H : H^1(\Omega) \to V_H$ be the L^2-projection.

Lemma 4.1 *Assume that the constructed basis preserves the constant function. Then the following stability and approximation properties hold for the agglomerated subspaces:*

$$\|Q_H v\|_{1,\Omega} \leq c\|v\|_{1,\Omega}, \tag{4.3}$$

$$\|v - Q_H v\|_{0,\Omega} \leq cH|v|_{1,\Omega}, \quad \forall v \in H^1(\Omega). \tag{4.4}$$

4.4 Numerical examples

We consider an elliptic equation of following type:

$$\begin{cases} -\dfrac{\partial}{\partial x}\left(a(x,y)\dfrac{\partial u}{\partial x}\right) - \dfrac{\partial}{\partial x}\left(b(x,y)\dfrac{\partial u}{\partial x}\right) = 0, \quad (x,y) \in \Omega \subset \mathbb{R}^2, \\ u(x,y) = 1, \quad (x,y) \in \partial\Omega. \end{cases} \tag{4.5}$$

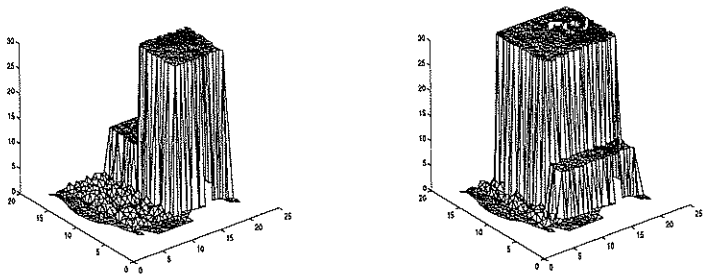


Figure 4.8: Surface plot of the coefficients for Example 2 — rapidly varying coefficients.

We use three types of coefficients for the equation (4.5) on three different grids. As a standard example we take $a = b = 1$, i.e. the Laplace operator.

In the next example, *Example 1*, the coefficients are mildly varying: $a(x,y) = (x^2 + y^2 + 1 + sin(x+y))$ and $b(x,y) = (x^2 + y^2 + 1 + cos(x+y))$. In *Example 2* the coefficients are varying in the range $[10^{-3}, 30]$. For the grid given in Fig. 4.9 the surface plot of the coefficients for Example 2, can be seen in Fig. 4.8. For all the grids the coefficients change in the same range. In these experiments we use the standard V-cycle preconditioner and the outer acceleration is done by CG method. In the V-cycle we use 1-pre and 1-post smoothing steps. The smoothing operator is forward Gauß-Seidel. The PCG iterations are terminated when the relative residual is less then 10^{-6}.

In Figures 4.9–4.11, the macroelements are shown for different unstructured grids and different number of levels. Figure 4.12 shows the convergence histories for the different types of coefficients and different grids. All these experiments in are done using the simplest interpolation algorithm 4.4. Figure 4.13 shows the convergence histories for a varying number of unknowns on two types of grids. One of these (one-element airfoil) has one internal boundary, the other one has four internal boundaries. The numerical experiments are done using Algorithm 4.3.

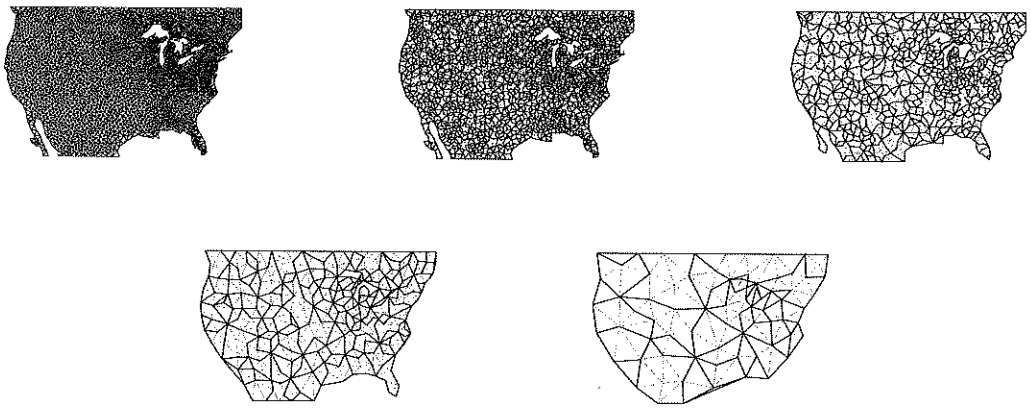


Figure 4.9: Macroelements for an unstructured grid level= 4 $N_h = 3422$; level= 3 $N_H^1 = 938$; level= 2 $N_H^2 = 268$; level= 1 $N_H^3 = 77$; level= 0 $N_H^4 = 21$.

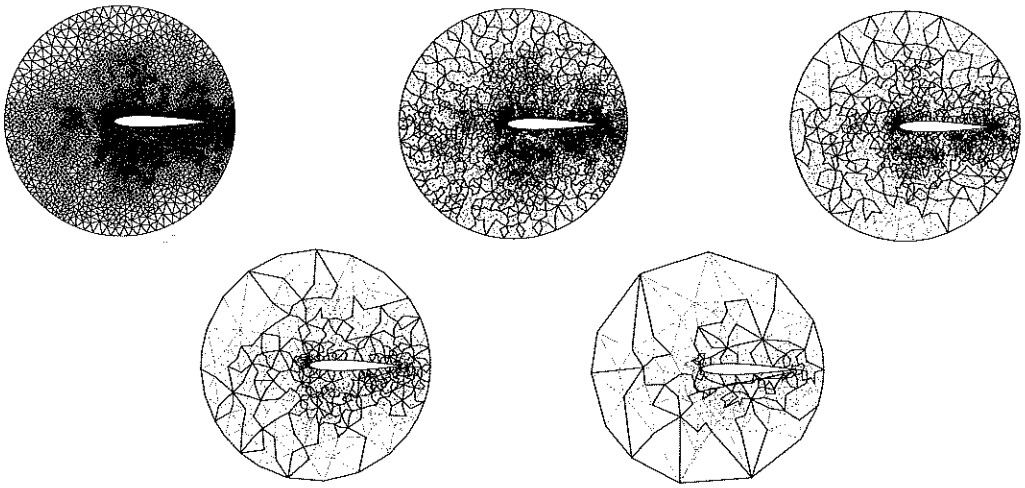


Figure 4.10: Macroelements for one element airfoil: level= 5 $N_h = 12665$; level= 4 $N_H^1 = 3404$; level= 3 $N_H^2 = 928$; level= 2 $N_H^3 = 257$; level= 1 $N_H^4 = 74$; level= 0 $N_H^5 = 24$.

These computational results show that the convergence is uniform with respect to the mesh size h. The convergence in the experiments shown on figure 4.13 is a little better because the aspect ratio of the grids is better and Algorithm 4.3 was used instead of Algorithm 4.4. The behavior for roughly varying coefficients is not as good, as it can be seen on Fig. 4.12. The grid on Fig. 4.9 is made with the Portable, Extensible Toolkit for Scientific Computation (PETSc) [50]. All other finite element grids grids used for the experiments in this section were produced using Barth's SIMPLEX2D mesh generator.

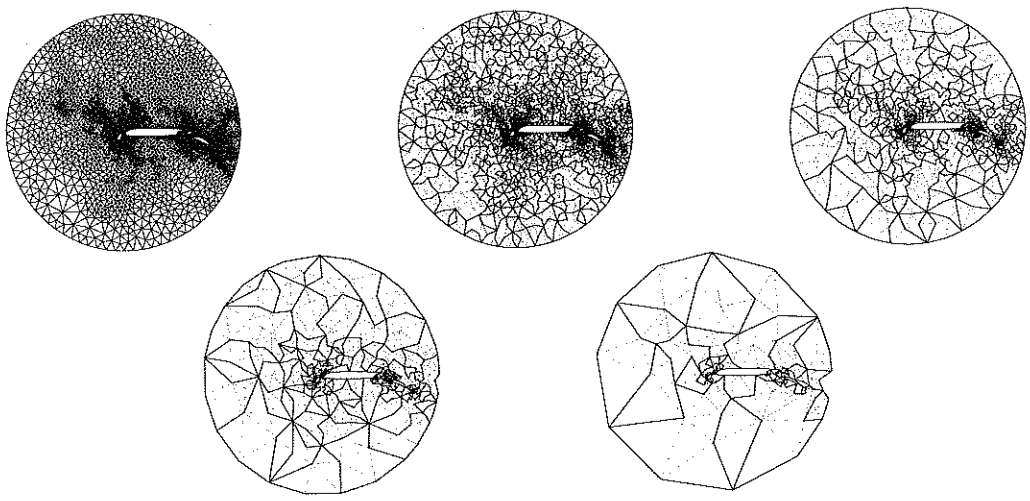


Figure 4.11: Macroelements for four element airfoil: level= 5 $N_h = 12850$; level= 4 $N_H^1 = 3444$; level= 3 $N_H^2 = 949$; level= 2 $N_H^3 = 270$; level= 1 $N_H^4 = 80$; level= 0 $N_H^5 = 26$.

Grid-Fig. 4.9

Example	Reduction factor
Laplace	0.11754
Example 1	0.12298
Example 2	0.24887

Grid-Fig. 4.10

Example	Reduction factor
Laplace	0.20701
Example 1	0.20724
Example 2	0.42600

Grid-Fig. 4.11

Example	Reduction factor
Laplace	0.21144
Example 1	0.21454
Example 2	0.46994

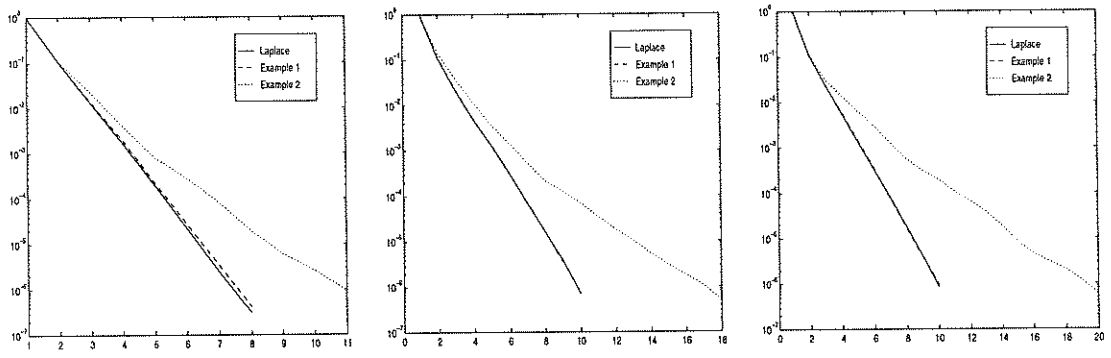


Figure 4.12: Convergence history and average reduction per iteration for Laplace equation, Example 1 and Example 2 on the different grids

4.5 Extensions

More "algebraic extensions" of the algorithm presented here can be derived. One of them is based generally on the graph of the matrix and does not use the dual graph. In this

1-element airfoil

nodes	Reduction factor
$N_h = 72139$	0.16834
$N_h = 18152$	0.14236
$N_h = 4683$	0.14911
$N_h = 1315$	0.14087

4-element airfoil

nodes	Reduction factor
$N_h = 72233$	0.15454
$N_h = 18328$	0.13836
$N_h = 4870$	0.15488
$N_h = 1502$	0.14727

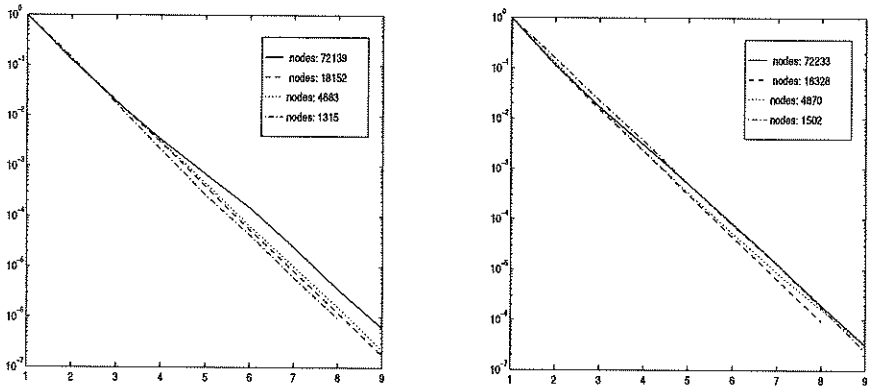


Figure 4.13: Convergence history and average reduction per iteration for varying number of unknowns

way each fine grid node belongs to macroelement with vertices few of the closest (3 or 4) coarse grid nodes. Depending on the matrix graph see [53], this algorithm might produce a huge number of non-zeros in the coarse grid operators after two or three levels. In the next numerical experiment such a type of agglomeration is called *algebraic coarsening.*

Smooth interpolations can also be done for such a method in an algebraic way. They are known as matrix dependent prolongations (see M. Griebel in [54]). In this case the prolongation operator is defined by $R_H^T = [A_{11}^{-1}A_{12}, I]^T$. Here A_{11} is the block in A_J formed by the natural splitting of the unknowns into two non overlapping subsets, corresponding to fine and coarse grid unknowns respectively:

$$A_J = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}. \tag{4.6}$$

The block A_{11} corresponds to the contributions $fine - fine$. Then it is straightforward to see that the coarse grid matrix is equal to the Schur complement of A, $S = A_{J-1} = A_{22} - A_{21}A_{11}^{-1}A_{12}$. Unfortunately block A_{11}^{-1} is generally a dense matrix, which is a serious drawback of using this approach. There are ways of defining approximations to this type of matrix dependent prolongations as proposed by A. Reusken [36] and M. Griebel in

[54].

The agglomeration coarsening algorithm presented here and the aggregation given in [38] also might be viewed as ways of approximating the first entry $A_{11}^{-1}A_{12}$ in the prolongation with a sparse matrix.

4.5.1 Anisotropic problems

Another issue we would like to comment on are the *anisotropic problems*. The problem in applying multigrid methods for such problems is that the smoother does not smooth the proper range of the high frequencies. A semi-coarsening (i.e. coarsening only in one direction) is often used to remedy this.

For anisotropic problems, the relevant changes in the agglomeration algorithm are straightforward. A dropping strategy can be used for the small off-diagonal elements in A_k on each level. A new coarse grid operator $\tilde{A}_k$ is then obtained and this matrix corresponds to a new graph which is disconnected. Different dropping strategies can be applied (see [13] for example). Here we apply a simple one:

$$\text{If } \sqrt{\frac{a_{ij}a_{ji}}{a_{ii}a_{jj}}} \leq 0.0001 \quad \text{then set } \; a_{ii} := a_{ii} + a_{ij};\; a_{jj} := a_{jj} + a_{ji};\quad a_{ij} := 0;\; a_{ji} := 0.$$

Then the algorithm is exactly the same as the *algebraic coarsening* algorithm. In the next example, the algorithm which uses the dropping strategy is called *reduced graph algorithm*. Similar approach for handling anisotropic problems can be found in [38].

The last numerical example in this section is the Laplace equation. The grid is given on Fig. 4.14. The anisotropy here is introduced by the geometry and the geometrical aspect ratio is of order 10^4. This grid was also generated by Barth's triangulator. It can be seen that the algorithm which uses the anisotropic agglomeration is faster than the others.

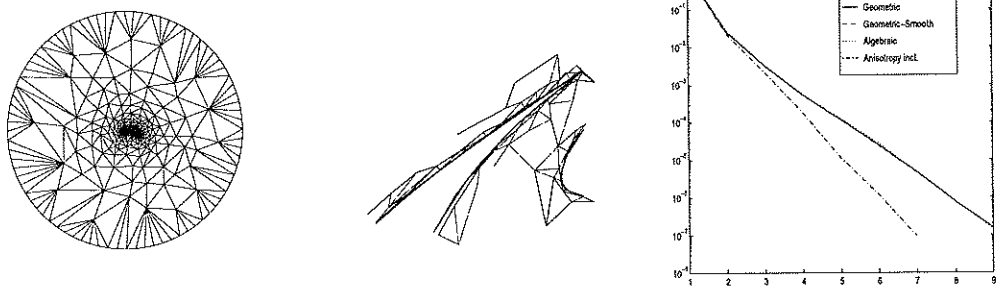


Figure 4.14: Macroelements (geometrical algorithm), second level geometric coarsening, zoomed 200 times near the airfoil. Convergence history for the stretched mesh: Geometrical coarsening; Geometrical coarsening-harmonic extension II; Algebraic coarsening; Reduced graph coarsening (anisotropy).

4.6 Concluding remarks

The agglomeration algorithms can provide a good approach for developing multilevel methods on unstructured grids. We presented here a general technique for constructing basis for the coarse space satisfying stability and approximation properties. We have to point out that the general theory for the construction of agglomerated spaces on unstructured grids is not still developed. On the other hand, numerically these methods have good performance and can be applied to a large set of problems, including elliptic, anisotropic and convection dominated problems. For such experiments, we refer to Koobus, Lallemand and Dervieux [37], Mandel, Vaněk, Brezina (see [38]) and the experiments presented in this section.

We presented three different types of basis construction over agglomerated macroelements: Algorithm 4.2, Algorithm 4.3, Algorithm 4.4. We prefer to use Algorithm 4.4, because the convergence rate is as good as with the other two algorithms and this algorithm is simpler. The last numerical experiment we performed shows that for more complicated problems, such as anisotropic problems, the interpolation must be done depending of the direction of the anisotropy. In this case, the algorithms for constructing the coarse grid need to be done very carefully, following this direction.

5 An algebraic nonoverlapping domain decomposition method for convection-diffusion problems

In this section, we will describe some recent joint work of the first author with Tim Barth of NASA Ames and Wei-Pei Tang of the University of Waterloo on using nonoverlapping Schur complement domain decomposition methods for flow problems [6].

The starting point of this approach is the block Gaussian elimination Schur complement framework described earlier in Section 2. The main feature, however, is that we will be making several approximations, both in the form of inexact subdomain solvers, as well as in setting up the coarse problem and solving it approximately. Unlike in the overlapping case described in Sections 3 and 4, the coarse space is constructed algebraically through an appropriate approximation of the Schur complement of the interface unknowns in the global stiffness matrix. The approximations are based on matrix element dropping strategies, localized Schur complement computations and *supersparse* matrix computations. Even though a rigorous theory for such an algebraic approach is lacking at this point, we are motivated in pursuing this approach because both the implementation and the performance are not sensitive to the type of PDE we are solving and works equally well for purely diffusive elliptic problems and convection dominated problems.

Our main objectives are to develop robust and efficient parallel solvers for the compressible Navier-Stokes flows about general geometries discretized on simplicial unstructured meshes. Our focus is on algorithms that possess coarse grain parallelism and our target architectures are parallel systems with a significant but not massive number of processors. A sample of such systems available at NAS at NASA Ames includes the IBM SP2 (160 processors), the SGI Array (40 processors) and the Cray J90 (20 processors).

We will first describe the fluid flow context in which the discretized PDE problems arise before moving on to describe the details of the domain decomposition preconditioner itself.

5.1 A model convection-diffusion problem

Let us consider the model convection diffusion equation:

$$u_t + div(\lambda u) = \epsilon \Delta u, (x,t) \in \Omega \times [0,T]$$

with

$$u(x,0) = u_0(x),\ x \in \Omega$$
$$u(x,t) = g(x,t),\ x \in \Gamma,$$

where λu represent the flux of the transported quantity.

For the discretization, we shall consider the Streamline Upwind Petrov-Galerkin (SUPG) finite element method of Johnson, Hughes, et. al. [10]: Find $u^h \in V_h$ such that $\forall\ w \in V_h$

$$\begin{aligned} \frac{\partial}{\partial t}\int_\Omega w\, u^h\, d\Omega &+ \int_\Omega w\, div(\lambda u) d\Omega + \int_\Omega \epsilon(\nabla w \cdot \nabla u)\, d\Omega \\ &+ \int_\Omega (div(\lambda u) - \epsilon\Delta u)\, \tau\, (div(\lambda w) - \epsilon \Delta w)\, d\Omega \\ &+ \text{Discontinuity Capturing Operator} = 0 \end{aligned}$$

This stabilized numerical method extends naturally to systems of conservation laws such as the Euler and Navier-Stokes equations governing fluid flow. Inserting standard C^0 finite element spatial approximations yields coupled ordinary differential equations of the form:

$$Du_t = R(u), \quad R(u) : \mathbf{R}^n \to \mathbf{R}^n,\ D > 0, D \in \mathbf{R}^{n \times n}$$

Linearizing with Newton's method and using a backward Euler time integration gives:

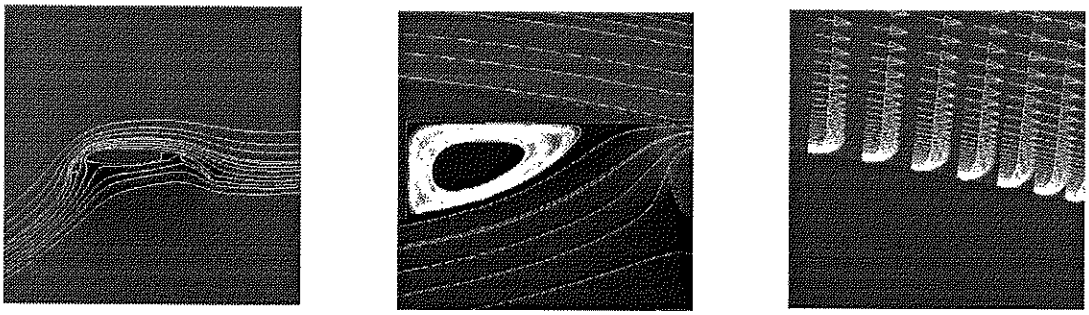
$$\left[\frac{1}{\Delta t} D - \left(\frac{\partial R}{\partial u}\right)^n\right](u^{n+1} - u^n) = R(u^n).$$

The above equation can also be viewed as a *modified* Newton method for solving the steady state equation $R(u) = 0$. In practice, we let Δt vary with $\|R(u)\|$ so that Newton's method is approached as $\|R(u)\| \to 0$.

Consider a representative matrix problem $Ax = b$ taken from one step of (5.1). We shall be describing preconditioners for this linear system. We shall allow nonstationary preconditioning and hence we shall be using the *flexible* GMRES method described in Section 1.

It is well known that many preconditioning techniques for elliptic problems are not robust for convection dominated problems. In particular, their performance can be quite sensitive to the nature of the characteristic directions of the flow field. In Figure 5.1, we show three different kinds of flows that are known to cause convergence problems for

many preconditioning techniques. The entrance/exit flow is probably the most benign of the three but still care must be taken in the preconditioner to "follow" the characteristic direction (from left to right in this case). The boundary layer flow has high anisotropy which can cause severe problems for many preconditioners. The recirculation flow is probably the most problematic of the three. In some sense, there is a stronger global coupling than the other two flows.



Entrance/exit flow | Recirculation flow | Boundary layer flow

Figure 5.1:

Figure 5.2 shows the convergence history of an ILU (with Cuthill-McKee ordering) preconditioned GMRES iteration for an entrance/exit flow and a recirculation flow with and without dissipation. It can be seen that the convergence rate can be much worse for the non-dissipative recirculation flow. Dissipation (perhaps artificially added) can alleviate the problem somewhat, but can adversely affect the quality of the numerical solution.

We now look at some potential candidate preconditioners and consider their pro's and con's:

1. The ILU family of preconditioners are purely algebraic and can be generally applicable but their convergence rate can be sensitive to the mesh size[1] (See Figure 5.3) and they are not easily parallelizable without some further deterioration in convergence rate [55]. Furthermore, ILU preconditioners can be sensitive to how well the ordering of the unknown can follow the flow characteristics.

2. Another possibility is the classical multigrid methods as preconditioners. While they are unquestionably successful for elliptic problems, their behavior for convection dominated flow problems are still poorly understood and remains a research frontier.

3. Additive Schwarz variants of ILU on overlapping subdomains can alleviate somewhat the sensitivity to the flow characteristics but to be scalable they need a coarse grid (or more generally a coarse space) and an approximation of the PDE on it (see Sections 3 and 4) which may not be as easy to derive for convection dominated problems as for standard elliptic problems. Moreover, as stated earlier, tetrahedral coarsening is problematic in 3-D.

[1]This behavior is usually called *non-optimal* in the literature.

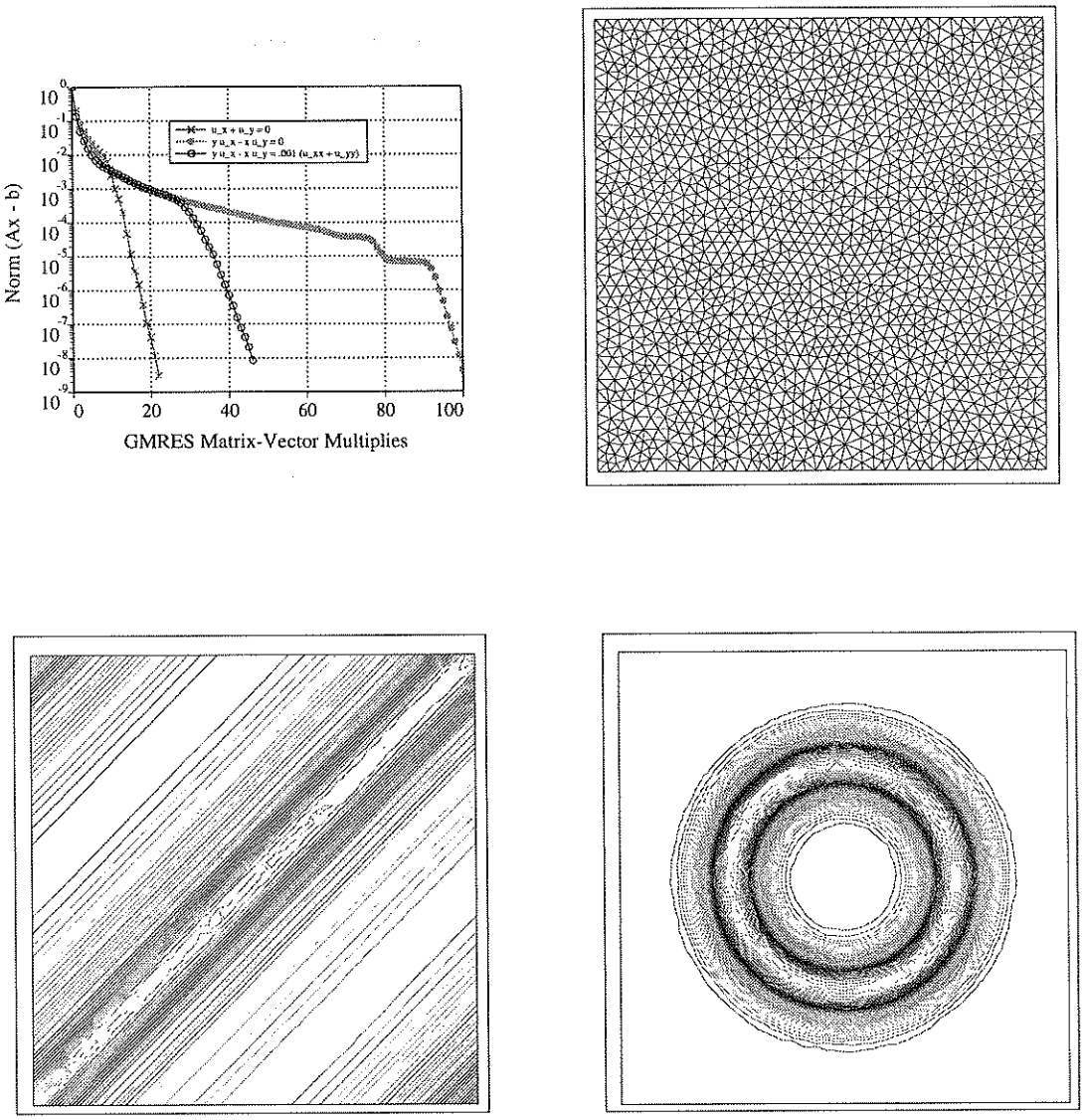


Figure 5.2: Performance of ILU for two different flows.

4. This leads us to the subject of this section, namely nonoverlapping Schur complement domain decomposition preconditioners. We shall be using an algebraically constructed coarse space to provide the global coupling, which makes them less sensitive to the ordering of the unknowns or the particular types of flow characteristics. Finally, like the overlapping methods, they are well suited to coarse grain parallel architectures.

Figures 5.4 and 5.5 taken from [56] show the performance of the additive Schwarz algorithm used as a preconditioner for GMRES. The test matrix was taken from one step of Newton's method applied to an upwind finite volume discretization of the Euler equations at low Mach number ($M_\infty = .2$). These calculations were performed without coarse mesh correction. As expected, the graphs show a degradation in quality with

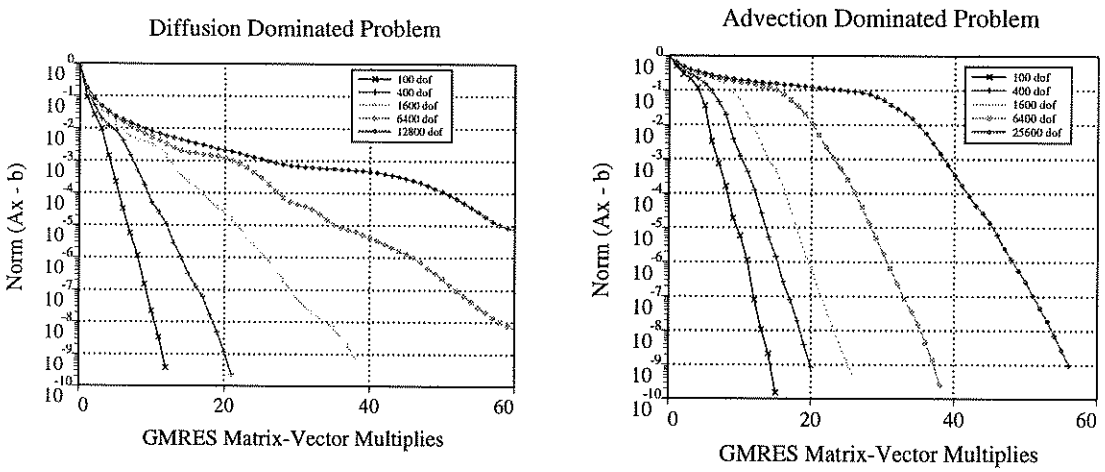


Figure 5.3: Dependence of ILU on mesh size for diffusion and advection dominated problems using scalar SUPG discretization and Cuthill-McKee ordering.

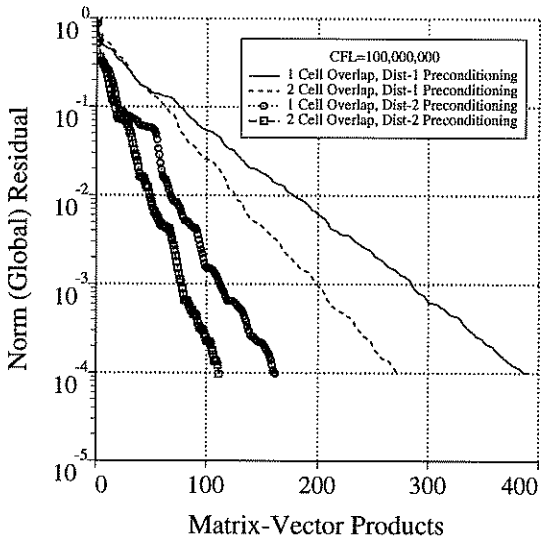


Figure 5.4: Effect of increased overlap and matrix fill.

decreasing overlap and increasing number of mesh partitions.

5.2 Nonoverlapping domain decomposition via Schur complement

Consider the subdomain partition as illustrated in Fig. 5.6. As discussed earlier in Sec. 1, if we order the subdomain variables before the interface variables, the discrete equations can be written in a 2×2 partitioned form as illustrated in Fig. 5.6.

We can write the partitioned linear system in the following form:

$$Ax = \begin{bmatrix} A_{II} & A_{IB} \\ A_{BI} & A_{BB} \end{bmatrix} \begin{pmatrix} x_I \\ x_B \end{pmatrix} = \begin{pmatrix} f_I \\ f_B \end{pmatrix},$$

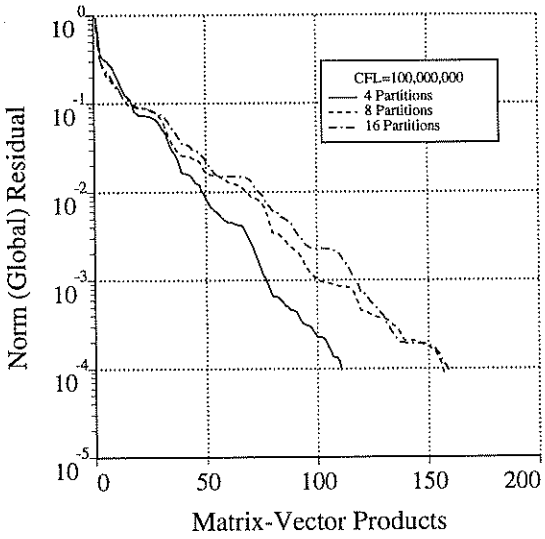


Figure 5.5: Effect of increasing the number of subdomains.

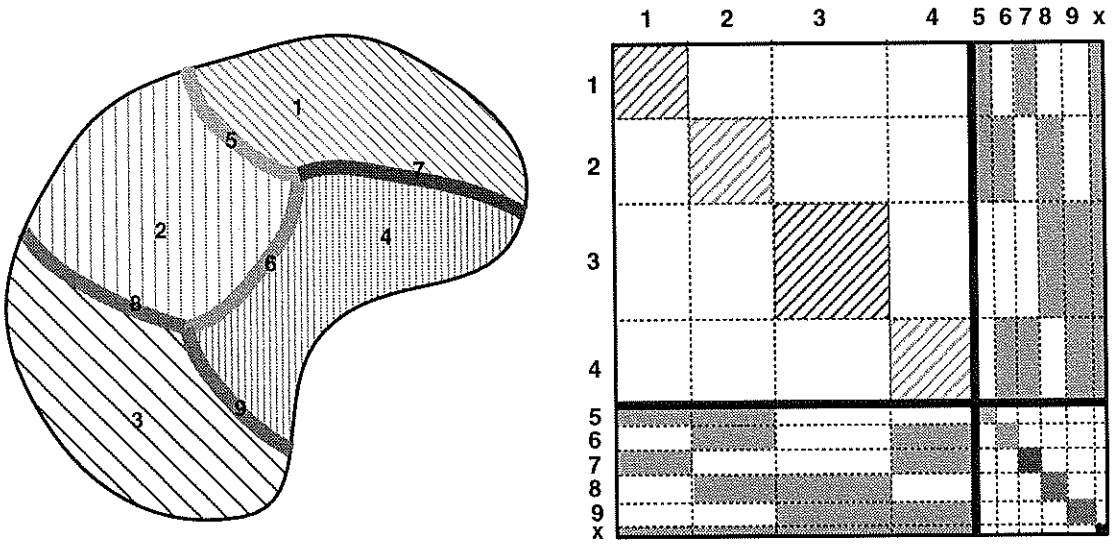


Figure 5.6: Domain decomposition and induced 2×2 matrix partitioning.

where x_I, x_B denote the interior and the boundary variables respectively. By using the following block LU factorization of A,

$$A = LU = \begin{bmatrix} A_{II} & 0 \\ A_{BI} & I \end{bmatrix} \begin{bmatrix} I & A_{II}^{-1} A_{IB} \\ 0 & S \end{bmatrix}.$$

we can eliminate x_I by

$$x_I = A_{II}^{-1}(f_I - A_{IB} x_B),$$

and solve for x_B from the reduced equations:

$$S x_B = f_B - A_{BI} A_{II}^{-1} f_I$$

where $S = A_{BB} - A_{BI} A_{II}^{-1} A_{IB}$ is the *Schur Complement* of A_{BB} in A.

From this block partitioning, we can easily derive an *exact* Schur Complement solver for A system as follows. First, in a preprocessing step, calculate the Schur complement

$$S = A_{BB} - \sum_i (A_{BI})_i (A_{II})_i^{-1} (A_{IB})_i,$$

where the $(A_{II})_i$'s are the diagonal blocks of A_{II} corresponding to the i-th subdomain and the $(A_{IB})_i$'s and $(A_{BI})_i$'s are the corresponding blocks of A_{IB} and A_{BI} respectively. Then we can solve the system $Ax = r$ via the following steps:

Step (1)	$u_i = (A_{II})_i^{-1} r_i$	(parallel)
Step (2)	$v_i = (A_{BI})_i u_i$	(parallel)
Step (3)	$w_b = r_b - \sum v_i$	(communication)
Step (4)	$x_b = S^{-1} w_b$	(serial,communication)
Step (5)	$y_i = (A_{IB})_i x_b$	(parallel)
Step (6)	$x_i = u_i - (A_{II})_i^{-1} y_i$	(parallel)

The above approach requires exact solvers for the subdomain problems A_i's and the Schur complement S. As explained earlier in Sec. 1, the cost can be dominated by just forming S alone, not even counting the cost associated with solving a linear system with it. Instead, we shall use the setup above to derive a family of preconditioners for A. Our general approach is to implement a flexible GMRES outer iteration with a preconditioner for A built from inner iterations for solving *approximately* the S and A_{II} linear systems in the above setup. The various preconditioners differ in how the approximations for the A_{II} and S solves are constructed. We shall describe three such preconditioners next. As indicated above, both the vector dot products and the matrix-vector products can be easily parallelized.

5.3 Preconditioner I: Inexact Subdomain Solve

The main idea behind the first preconditioner is to replace the exact solves with the A_i's and S by a fixed number of ILU preconditioned GMRES iterations. Specifically, in the "exact" algorithm above, we make the following replacements:

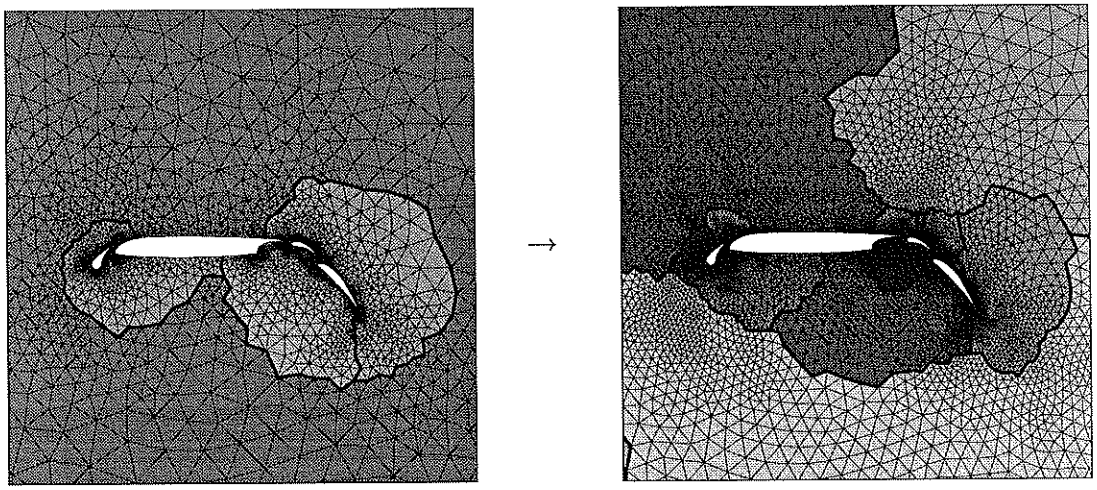
1. $(A_{II})_i^{-1}(A_{IB})_i \rightarrow (\tilde{A}_{II})_i^{-1}(A_{IB})_i$ using m_1 steps of ILU+GMRES so that $S \rightarrow \tilde{S}$.
2. $(A_{II})_i^{-1} z_i \rightarrow \widehat{A}_{II}^{-1} z_i$ using m_2 steps of ILU+GMRES.
3. $\tilde{S}^{-1} w_b \rightarrow \widehat{S}^{-1} w_b$ using m_3 steps of ILU+GMRES.

In the preprocessing step, we calculate the *approximate* Schur complement:

$$\tilde{S} = A_{BB} - \sum_i (A_{BI})_i (\tilde{A}_{II})_i^{-1} (A_{IB})_i.$$

Now the preconditioner step becomes (solving $Mx = r$):

Step (1)	$u_i = (\widehat{A}_{II})_i^{-1} r_i$	(parallel)
Step (2)	$v_i = (A_{BI})_i u_i$	(parallel)
Step (3)	$w_b = r_b - \sum v_i$	(communication)
Step (4)	$x_b = \widehat{S}^{-1} w_b$	(serial,communication)
Step (5)	$y_i = (A_{IB})_i x_b$	(parallel)
Step (6)	$x_i = u_i - (\widehat{A}_{II})_i^{-1} y_i$	(parallel)



Next we evaluate the performance of the baseline Schur complement strategy for scalar diffusion and advection dominated test problems using Galerkin and SUPG finite element schemes respectively. We consider a scalability experiment for which the mesh size and number of processors are successively quadrupled. This implies in 2-D that the number of solution unknowns for each subdomain is held roughly constant. In these calculations the adjustable parameters are all chosen identical $m_1 = m_2 = m_3 = 2$. Tables 4 shown below gives the number of global GMRES iterations needed to solve the matrix problem to a 10^{-9} iterative stopping criteria. The timing columns give the time needed to form the Schur complement preconditioner and the time needed to apply the Schur complement preconditioner to vectors supplied by the outer (global) GMRES iteration. Observe that the time to apply the Schur preconditioner is proportional to the number of global GMRES iterations needed to solve the matrix problem. The number of iterations starts at 12 for a single domain solve, drops to only 3 iterations for 4 subdomains, and grows slowly as the mesh size is increased. The initial drop in number of iterations when solving the single and 4 subdomain problems can be understood as the result of breaking the single domain ILU recurrence into four smaller recurrences recombined via Schur

complement. Recall that the ILU factorization introduces an error which propagates with the recurrence. The shorter recurrences have less error propagation before being recombined by the Schur complement. This can dramatically improve the quality of the factorization. The slow growth in the number of iterations is a direct consequence of our choice of parameters m_1, m_2, and m_3. Clearly, we could reduce the number of global GMRES iterations simply by increasing the value of these parameters. Unfortunately, this may result in larger execution times. From our experience choosing small values of the parameters m_1, m_2, and $m_3 < 3$ often leads to minimum CPU times. Turning to the fourth column of the tabulated results, we observe a linear growth in the time needed to form the preconditioner. This growth is easily explained given that we have chosen to assign a single processor to the task of forming the Schur complement and individual processors for each subdomain. In our scalability experiment, the number of elements in each subdomain is held roughly constant so that the execution time is also nearly constant and scalability of the subdomain work is obtained. Unfortunately, a number of seemingly innocuous operations performed by the Schur complement processor are proportional to size of the Schur complement, e.g. matrix assembly of the interface. Since the size of the interface and Schur complement grow linearly in the scalability experiment, eventually the work becomes imbalanced between subdomains and Schur complement processor. In more recent work (not presented here), we have further decomposed the Schur complement matrix among additional processors so that scalability can be approximately obtained.

Table 4: Scalability experiments for the baseline Schur complement preconditioner.

Pure Diffusion Problem:

Mesh Size	Number of Subdomains	Iter	Time Form	Time Apply
100	1	12	0	.02
400	4	3	.03	.02
1600	16	4	.04	.03
6400	64	5	.12	.04
25600	256	7	.44	.06

Pure Advection Problem:

Mesh Size	Number of Subdomains	Iter	Time Form	Time Apply
100	1	12	0	.02
400	4	3	.02	.02
1600	16	4	.03	.03
6400	64	5	.11	.04
25600	256	5	.44	.04

5.4 Preconditioner II: Drop Tolerance Approximation

It is clear that in Preconditioner I, the major bottleneck is the cost in forming and solving the S system. First, we note that S is generally much denser than A: A particular variable on the interface is coupled to all other variables sharing a common subdomain. This increased density of $\tilde{S}$ increases the cost of preconditioning and solving its associated linear systems. We thus look for ways to further approximate $\tilde{S}$ by a sparser approximation. We are helped in this regard by the well-known fact that, for elliptic problems at least, the size of the entries of S are rapidly decaying away from the main diagonal. The decay is faster than the decay of the corresponding entries of $(A_{II})_i^{-1}$ [57]. Therefore, a natural idea is to drop all elements of $\tilde{S}$ whose size is less than a given tolerance: i.e. drop elements of the Schur complement matrix, $\tilde{S}_{ij}$ if:

(1) $|S_{ij}| < tol$ (scalar matrix entries)

(2) According to a prescribed sparsity pattern

The preprocessing step now becomes: Calculate the approximate Schur complement:

$$\tilde{S} = A_{BB} - \sum_i (A_{BI})_i \tilde{A}_i^{-1} (A_{IB})_i, \quad \widehat{S} = DROP(\tilde{S}),$$

where $DROP$ represents the specific dropping strategy for $\tilde{S}$.

The preconditioning step (solving $Mx = r$) is now:

Step (1)	$u_i = (\widehat{A}_{II})_i^{-1} r_i$	(parallel)
Step (2)	$v_i = (A_{BI})_i u_i$	(parallel)
Step (3)	$w_b = r_b - \sum v_i$	(communication)
Step (4)	$x_b = \widehat{S}^{-1} w_b$	(serial,communication)
Step (5)	$y_i = (A_{IB})_i x_b$	(parallel)
Step (6)	$x_i = u_i - (\widehat{A}_{II})_i^{-1} y_i$	(parallel)

5.5 Preconditioner III: Wireframe Approximation

The main cost in Preconditioner II is the formation of S itself: one subdomain solve is needed for *every* variable on the interface. The main idea in Preconditioner III is to approximate S by the (approximate) Schur complement of a relative thin *wireframe* region surrounding the interface variables. In matrix terms, what we do is to take a principal submatrix of A corresponding to the variables within the wireframe and compute the (approximate) Schur complement of the interface variables in this principal submatrix instead of A itself. The main advantage is that the subdomain solves to form this approximation Schur complement is much cheaper because the number of variables within each subdomain is much reduced. It is well known in domain decomposition

theory that the exact Schur complement of the wireframe region is spectrally equivalent to the Schur complement of the whole domain.

The preprocessing step is now: Calculate the approximate Schur complement on the localized subdomains:

$$\tilde{S} = A_{BB} - \sum_i (\tilde{A}_{BI})_i (\tilde{A}_{II})_i^{-1} (\tilde{A}_{IB})_i, \quad \widehat{S} = DROP(\tilde{S})$$

where $(\tilde{A}_{II})_i, (\tilde{A}_{IB}), \tilde{A}_{BI}$ are algebraic restrictions of A_i, B, C to the localized Schur subdomain(s).

The preconditioning step (solving $Mx = r$) is now:

Step (1)	$u_i = (\widehat{A}_{II})_i^{-1} r_i$	(parallel)
Step (2)	$v_i = (A_{BI})_i u_i$	(parallel)
Step (3)	$w_b = r_b - \sum v_i$	(communication)
Step (4)	$x_b = \widehat{S}^{-1} w_b$	(serial,communication)
Step (5)	$y_i = (A_{IB})_i x_b$	(parallel)
Step (6)	$x_i = u_i - (\widehat{A}_{II})_i^{-1} y_i$	(parallel)

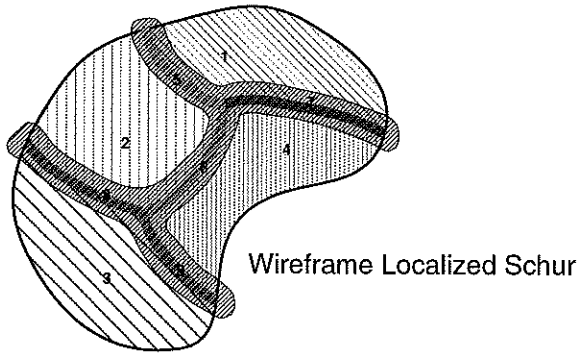


Figure 5.7: Wireframe Localized Schur Complement Approximation

The wireframe technique introduces a new adjustable parameter into the preconditioner. For simplicity, we usually specify the width of the wireframe strip in terms of graph distance on the mesh triangulation. Table 5 gives performance results for a fixed mesh size (1600 vertices) while varying the width of the wireframe. For sake of comparison, we have chosen all other parameters to be identical to the baseline results presented earlier. As expected, the quality of the preconditioner improves rapidly with increasing wireframe width with baseline-like results obtained using modest wireframe widths. As a consequence of the wireframe construction the time taken form the Schur complement

has dropped by approximately 50%. Further improvements in cost/performance might be obtainable by choosing the shape of the wireframe to better represent the PDE being solved.

Table 5: Performance results for a fixed mesh size (1600 vertices) while varying the width of the wireframe.

Pure Diffusion Problem:

Mesh Size	Number of Subdomains	Support	Iter	Time Form*	Time Apply
1600	16	2	15	.012	.11
1600	16	3	11	.017	.08
1600	16	4	6	.020	.04

Pure Advection Problem:

Mesh Size	Number of Subdomains	Support	Iter	Time Form*	Time Apply
1600	16	2	8	.012	.09
1600	16	3	6	.017	.06
1600	16	4	4	.019	.03

5.6 Preconditioner IV: Supersparse Matrix Approximation

Although the wireframe construction significantly reduces the cost of forming the Schur complement matrix, it is possible to introduce further approximations which improve on the overall efficiency of the Schur complement preconditioner. To see this, begin by noting the following observations:

1. Iterative calculation of $A_{II}^{-1}A_{IB}$ is expensive.
2. Columns of A_{IB} are usually very sparse.
3. Unpreconditioned Krylov subspace methods require computation of the vector sequence $[p, Ap, A^2p, \ldots, A^mp]$, $p = col(B)$ for small m.
4. ILU preconditioning destroys sparsity of the preconditioned Krylov subspace vectors $[p, MAp, (MA)^2p, \ldots, (MA)^mp]$.

From these observations we have developed the following cost saving strategy:

1. Eliminate wasted flop's in matrix-vector products by storing *vectors* as well as matrices in sparse form.
2. Approximate the ILU backsolves based on a vector fill level strategy to maintain sparsity.

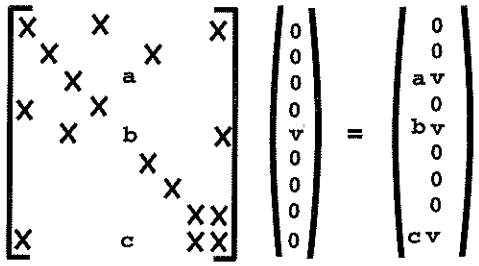


Figure 5.8:

The motivations should be clear. By storing vectors in sparse form, we avoid multiplying nonzero matrix entries by zero vector entries, thus saving a significant number of flop's. The idea of approximate ILU backsolves based on fill level is motivated by the known decay phenomena associated with elements of A^{-1} for elliptic problems. To carry out the preconditioning step, we first compute a sparsity pattern to be imposed onto the resulting vector by symbolically computing and truncating the fill level structure produced during the lower and upper triangular solves (sweeps). Next we perform the actual numerical backsolve truncating the computation to nonzero elements imposed on the resulting vector.

Table 6 shows the performance of the Schur complement preconditioner with supersparse arithmetic for a 2-D test problem consisting of Euler flow past a multi-element airfoil geometry partitioned into 4 subdomains with 1600 mesh vertices in each subdomain. Computations were performed on the IBM SP2 parallel computer using MPI message passing protocol. Various values of backsolve fill level were chosen while monitoring the number of global GMRES iterations needed to solve the matrix problem and the time taken to form the Schur preconditioner. Results for this problem indicate preconditioning performance comparable to exact backsolves using backsolve fill levels of only 2 or 3 with a 60-70% reduction in cost. Note that this technique can be combined with the previous wireframe strategy with combined 5-7 fold performance gains.

Table 6: Performance of the Schur complement preconditioner with supersparse arithmetic for a 2-D test problem consisting of Euler flow past a multi-element airfoil geometry partitioned into 4 subdomains with 1600 mesh vertices in each subdomain.

Fill Level k	Global GMRES Iter	Time(k)/Time(∞)
0	26	0.299
1	22	0.313
2	21	0.337
3	20	0.362
4	20	0.392
∞	20	1.000

5.7 Concluding Remarks

Experience with our nonoverlapping domain decomposition method with an algebraically generated coarse problem shows that we can successfully trade off some of the robustness of the exact Schur complement method for increased efficiency by making appropriately designed approximations. In particular, the localized wireframe approximation and the supersparse matrix-vector approximation together result in dramatically lowering the cost without giving up very much in convergence rate.

Much work remains to be done. For example, our current implementation solves the interface system in one single processor. As the number of subdomains increase, the growth of the Schur complement matrix necessitates further work which should be distributed among all the available processors. We shall report on these developments and their applications to various flow problems in the future.

References

[1] T. F. Chan, B. Smith, and J. Zou. Overlapping Schwarz methods on unstructured meshes using non-matching coarse grids. *Numerische Mathematik*, 73(2):149–167, April 1996.

[2] T. F. Chan, B. Smith, and J. Zou. Multigrid and domain decomposition methods for unstructured meshes. In I.T. Dimov, Bl. Sendov, and P. Vassilevski, editors, *Proc. of the 3rd Int'l Conf. on Advances in Numerical Methods and Applications, Sofia, Bulgaria.*, pages 53–62. World Scientific, August 1994.

[3] T. F. Chan, S. Go, and J. Zou. Multilevel domain decomposition and multigrid methods for unstructured meshes: Algorithms and theory. Technical Report CAM-95–24, Department of Mathematics, University of California at Los Angeles, May 1995. also in Proceedings of the Eighth International Conference on Domain Decomposition, May 1995, Beijing.

[4] T. F. Chan, S. Go, and J. Zou. Boundary treatments of multilevel methods on unstructured meshes. Technical Report CAM 96–30, Department of Mathematics, University of California at Los Angeles, September 1996. To appear in SIAM J. Sci. Comp.

[5] Tony F. Chan, J. Xu, and Ludmil Zikatanov. Agglomeration strategies in multigrid method. in preparation.

[6] T. J. Barth, T. F. Chan, and W.-P. Tang. A parallel algebraic non-overlapping domain decomposition method for flow problems. In preparation.

[7] G. Strang and G. Fix. *An Analysis of the Finite Element Method.* Prentice Hall, 1973.

[8] P. Ciarlet. *The finite element method for elliptic problems.* North-Holland, 1978.

[9] S. Brenner and L.R. Scott. *The mathematical theory of finite element methods.* Springer-Verlag, 1994.

[10] Claes Johnson. *Numerical solution of partial differential equations by the finite element method.* Cambridge University Press, Cambridge, 1990.

[11] D.L. Brown, G. Chesshire, W.D. Henshaw, and H.-O. Kreiss. On composite overlapping grids. In T.J. Chung and G.R. Karr, editors, *Finite Element Analysis in Fluids*, pages 544–559. UAH Press, 1989.

[12] G. Chesshire and W. D. Henshaw. Composite overlapping meshes for the solution of partial differential equations. *J. Comp. Phys.*, 90(1):1–64, 1990.

[13] Yousef Saad. *Iterative methods for sparse linear systems.* PWS Publishing company, 1996.

[14] J. Xu. *An introduction to multilevel methods.* To be published by Oxford University Press, 1996. Lecture notes: VIIth EPSRC Numerical analysis summer school, Leicester University, UK.

[15] William Briggs. *A multigrid tutorial.* Society for Industrial and Applied Mathematics, Philadelphia, 1987.

[16] J. Bramble. *Multigrid methods.* Pitman, Notes on Mathematics, 1994.

[17] W. Hackbusch. *Multi-grid methods and applications.* Springer Verlag, New York, 1985.

[18] J. Xu. Iterative methods by space decomposition and subspace correction. *SIAM Review*, 34:581–613, December 1992.

[19] J. H. Bramble and J. E. Pasciak. New convergence estimates for multigrid algorithms. *Math. Comp.*, 49:311–329, 1987.

[20] J. H. Bramble, J. E. Pasciak, and J. Xu. The analysis of multigrid algorithms with nonnested spaces or noninherited quadratic forms. *Math. Comp.*, 56:1–34, 1991.

[21] T. F. Chan and T. Mathew. Domain decomposition algorithms. *Acta Numerica*, pages 61–143, 1994.

[22] B. Smith, P. Bjorstad, and W. Gropp. *Domain decomposition: parallel multilevel methods for elliptic partial differential equations.* Cambridge University Press, Cambridge, 1996.

[23] H. A. Schwarz. Gesammelte mathematische abhandlungen. *Vierteljahrsschrift der Naturforschenden Gesellschaft*, 15:272–286, 1870.

[24] G. Kron. A set of principles to interconnect the solutions of physical systems. *J. of Applied Physics*, 24(8):965, 1953.

[25] J. S. Przemieniecki. Matrix structural analysis of substructures. *AIAA Journal*, 1:138–147, 1963.

[26] Maksymilian Dryja and Olof B. Widlund. Some domain decomposition algorithms for elliptic problems. In Linda Hayes and David Kincaid, editors, *Iterative Methods for Large Linear Systems*, pages 273–291, San Diego, California, 1989. Academic Press.

[27] Maksymilian Dryja and Olof B. Widlund. Additive Schwarz methods for elliptic finite element problems in three dimensions. In Tony F. Chan, David E. Keyes, Gérard A. Meurant, Jeffrey S. Scroggs, and Robert G. Voigt, editors, *Fifth Conference on Domain Decomposition Methods for Partial Differential Equations*, Philadelphia, PA, 1992. SIAM.

[28] J. Bramble, J. Pasciak, and A. Schatz. The construction of preconditioners for elliptic problems by substructuring, I. *Math. Comp.*, 47:103–134, 1986.

[29] B. Smith. An optimal domain decomposition preconditioner for the finite element solution of linear elasticity problems. *SIAM J. Sci. Stat. Comput.*, 13(1):364–378, January 1992.

[30] D.J. Mavriplis. Unstructured mesh algorithms for aerodynamic calculations. Technical Report 92-35, ICASE, NASA Langley, Virginia, July 1992.

[31] H. Guillard. Node-nested multi-grid method with Delaunay coarsening. Technical Report RR-1898, INRIA, Sophia Antipolis, France, March 1993.

[32] T. F. Chan and Barry Smith. Domain decomposition and multigrid methods for elliptic problems on unstructured meshes. In David Keyes and Jinchao Xu, editors, *Domain Decomposition Methods in Science and Engineering, Proceedings of the Seventh International Conference on Domain Decomposition, October 27-30, 1993, The Pennsylvania State University.* American Mathematical Society, Providence, 1994. also in Electronic Transactions on Numerical Analysis, v.2, (1994), pp. 171–182.

[33] R. E. Bank and J. Xu. An algorithm for coarsening unstructured meshes. Technical report, Dept. of Math., Univ. of Calif. at San Diego, 1994.

[34] W. Hackbusch. *Iterative solution of large sparse systems of equations.* Springer Verlag, Heidelberg, 1993. Applied Mathematical Sciences, Vol. 95.

[35] D. Braess. Towards algebraic multigrid for elliptic problems of second order. *Computing*, 55:379–393, 1995.

[36] A. A. Reusken. A multigrid method based on incomplete Gaussian elimination. Technical Report RANA 95-13, Eindhoven University of Technology, October 1995.

[37] B. Koobus, M. H. Lallemand, and A. Dervieux. Unstructured volume-agglomeration MG: solution of the Poisson equation. *International Journal for Numerical Methods in Fluids*, 18(1):27–42, 1994.

[38] P. Vaněk, J. Mandel, and M. Brezina. Algebraic multi-grid by smoothed aggregation for second and forth order elliptic problems. *Computing*, 1995. to appear.

[39] P. Vaněk and J. Křižkovà. Two-level method on unstructured meshes with convergence rate independent of the coarse space size. Technical Report 33, University of Colorado at Denver, January 1995.

[40] A. M. Matsokin and S. V. Nepomnyaschikh. A Schwarz alternating method in a subspace. *Soviet Mathematics*, 29(10):78–84, 1985.

[41] P. Lions. On the Schwarz alternating method. I. In Roland Glowinski, Gene H. Golub, Gérard A. Meurant, and Jacques Périaux, editors, *First International Symposium on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, 1988.

[42] M. Dryja and O. Widlund. An additive variant of the Schwarz alternating method for the case of many subregions. Technical Report 339, also Ultracomputer Note 131, Department of Computer Science, Courant Institute, 1987.

[43] James H. Bramble, Joseph E. Pasciak, Junping Wang, and Jinchao Xu. Convergence estimates for multigrid algorithms without regularity assumptions. *Math. Comp.*, 57(195):23–45, 1991.

[44] T. F. Chan and J. Zou. Additive Schwarz domain decomposition methods for elliptic problems on unstructured meshes. *Numerical Algorithms*, 8:329–346, 1994.

[45] T. F. Chan and J. Zou. A convergence theory of multilevel additive Schwarz methods on unstructured meshes. Technical Report 95-16, Department of Mathematics, University of California at Los Angeles, March 1995. to appear in Numerical Algorithms.

[46] Ralf Kornhuber and Harry Yserentant. Multilevel methods for elliptic problems on domains not resolved by the coarse grid. In David Keyes and Jinchao Xu, editors, *Domain Decomposition Methods in Science and Engineering, Proceedings of the Seventh International Conference on Domain Decomposition, October 27-30, 1993, The Pennsylvania State University*, volume 180, pages 49–60. American Mathematical Society, Providence, 1994.

[47] R. Bank and J. Xu. An algorithm for coarsening unstructured meshes. *Numer. Math.*, 73:1–23, 1996.

[48] P. Clément. Approximation by finite element functions using local regularization. *R.A.I.R.O. Numer. Anal.*, R-2:77–84, 1975.