A dissertation presented to the faculty of the Graduate School of Yale University in candidacy for the degree of Doctor of Philosophy.

Domain Decomposition Algorithms for Elliptic Partial Differential Equations

Diana Cristina Resasco

YALEU/DCS/RR-776
May 1990

Work supported in part by " Consejo Nacional de Investigaciones Científicas y Técnicas", from Argentina, by the Department of Energy under contract DE-AC02-81ER 0996, by the Office of naval research under contract N00014-86-K-0310, and by IBM under grant P00038437.

Contents

List of Figures

List of Tables

ABSTRACT

Domain Decomposition Algorithms for Elliptic Partial Differential Equations

Diana Cristina Resasco

Yale University

1990

The technique of *Domain Decomposition* or *Substructuring* is applied to the solution of elliptic partial differential equations when the domain is divided into a number of subdomains. The decomposition may be motivated, for example, by the differential operator taking different parameters on each subdomain, or there might be a natural decomposition of the domain into regular pieces such as rectangles. The idea also provides a very natural approach to the solution of elliptic problems on multiprocessor systems. In this thesis we are interested, in particular, in a class of domain decomposition techniques whose unifying feature is the use of the preconditioned conjugate gradient method in solving a system for the unknowns on the separator set, given by the gridpoints at the interfaces between subdomains. In some cases, preconditioners for the problem on the whole domain can be derived from preconditioners for the interface system. Since each iteration involves solving problems on the subdomains, it is essential to keep the number of iterations low. For this reason, much effort has been devoted recently to the construction of good preconditioners for the conjugate gradient method.
This thesis presents an algebraic framework in which many of the preconditioners given in the literature can be analyzed and compared. In particular, two issues will be addresed: the construction of preconditioners that take the aspect ratios of the subdomains into account and the coupling between interfaces. For a few simple geometries, small numerical bounds on the condition number of the preconditioned interface system are derived. These bounds are independent of the subdomain aspect ratios. Some of the applications to multiprocessor systems include a parallel domain decomposed fast Poisson solver in two and three dimensions and parallel preconditioners for variable coefficients problems. Experiments were carried out on an Intel Hypercube multiprocessor system.

Chapter 1

Introduction

The terms *Domain Decomposition* and *Substructuring* refer to techniques for solving partial differential equations (PDE's) by first decomposing the domain into smaller regions or subdomains and then reducing the solution of the original problem to solving problems on these subdomains. The decomposition is sometimes motivated by the physics of the problem – e..g., when the equation takes different forms or it depends on parameters which take different values on the various subdomains. In other cases, the domain is decomposed in order to simplify the computation of the solution, for example, the domain is divided into regular pieces such as rectangles, where more efficient numerical techniques can be applied, or an out-of-core model of computation is used to solve large problems which do not fit in the main memory of a given machine. Finally, the idea is well suited to the numerical solution of PDE's in a parallel environment, since the subproblems can be solved independently and communication is limited to the boundaries of the subdomains.

The general concept of domain decomposition is not new. It can be traced to Schwarz's alternating procedure, which proves the existence of a solution for a boundary value problem by an iteration which involves overlapping subdomains [31]. The rate of convergence of this procedure depends on the amount of overlapping: more overlapping implies faster convergence, but more work per iteration, since the work is duplicated on the overlapping region.

The methods considered in this thesis apply to non-overlapping subdomains. The basic idea is to reduce the differential operator on the whole domain to an operator on the interfaces. After solving for the interface values, the solution at the interior of the subdomains can be computed by solving independent problems with the computed interface values as boundary conditions.

1.1 Description of the Method

The domain is discretized and partitioned into several subregions; then, by applying block elimination to the discretized equations, a system is derived for the unknowns on the interfaces between subregions. The interface system is sometimes called the *capacitance system.* Once this system is solved and the solution is known at the interfaces, the original problem is decoupled and the solution at the interior of the subdomains can be found by solving independent problems on the subdomains. Forming the right hand side for the interface system requires the solution of independent elliptic problems on the subdomains. For certain constant coefficient problems on regular domains, fast direct methods can be applied to the solution of this system. Such is not the case, however, for more general operators on irregular domains. For efficiency reasons, the interface system must therefore be solved by iterative methods, such as the preconditioned conjugate gradient method (PCG).

The method described above is particularly suited to problems for which the subproblems can be solved efficiently, for example, when the operator has separable coefficients and the domain is a union of rectangles. On the other hand, when the subdomain problems cannot be solved efficiently but they can

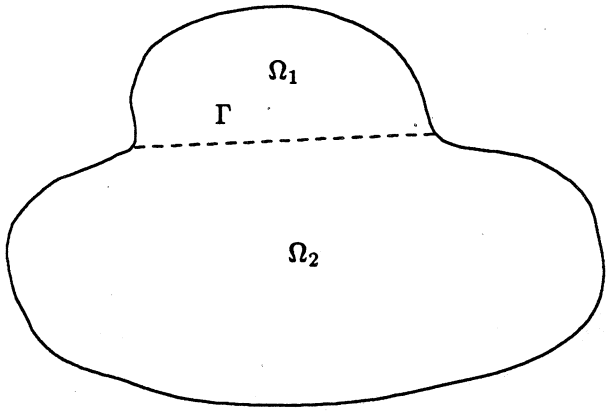

Figure 1.1: The domain Ω and its partition

be approximated by simpler operators, it is possible to derive block preconditioners for the original system based on preconditioners for the capacitance matrix [7, 28].

In order to illustrate the domain decomposition method, consider the problem:

$$\begin{aligned} Lu &= f \quad \text{in} \quad \Omega \\ u &= u_b \quad \text{on} \quad \partial\Omega \quad , \end{aligned} \tag{1.1}$$

where L is a linear elliptic operator and Ω is the domain in Fig. 1.1, which can be decomposed into two subdomains Ω_1 and Ω_2, separated by the interface Γ.

Let the linear system

$$Au = f \tag{1.2}$$

represent a discretization of (1.1), where u now represents the discrete solution vector. If we order the unknowns for the internal points of the subdomains first and then those on the interface Γ, then the linear system (1.2) can be expressed in block form as:

$$\begin{pmatrix} A_{11} & 0 & A_{13} \\ 0 & A_{22} & A_{23} \\ A_{13}^T & A_{23}^T & A_{33} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} \quad . \tag{1.3}$$

Using the following block decomposition of the matrix A,

$$A = \begin{pmatrix} A_{11} & 0 & 0 \\ 0 & A_{22} & 0 \\ A_{13}^T & A_{23}^T & C \end{pmatrix} \begin{pmatrix} I & 0 & A_{11}^{-1}A_{13} \\ 0 & I & A_{22}^{-1}A_{23} \\ 0 & 0 & I \end{pmatrix} \quad , \tag{1.4}$$

where C is the Schur complement of A_{33} in A, i.e.,

$$C = A_{33} - A_{13}^T A_{11}^{-1} A_{13} - A_{23}^T A_{22}^{-1} A_{23} \quad , \tag{1.5}$$

system (1.3) can be solved by block-Gaussian elimination as follows:

ALGORITHM BGE:

Step 1: Compute $g = f_3 - A_{13}^T A_{11}^{-1} f_1 - A_{23}^T A_{22}^{-1} f_2$.

Step 2: Compute u_3 by solving $Cu_3 = g$.

Step 3: Compute u_1 and u_2 by solving

$$A_{11}u_1 = f_1 - A_{13}u_3$$
$$A_{22}u_2 = f_2 - A_{23}u_3$$

Note that, except for Step 2, the algorithm only requires the solution of problems with A_{11} and A_{22}, which corresponds to solving independent problems on Ω_1 and Ω_2. The Schur complement C is also called the *capacitance matrix*.

Solution of the Interface System

In the continuous case, the interface equation corresponds to an equation for u on Γ derived from the condition that the solutions of problems on the subdomains have the same traces and normal derivatives on the interface (see [1]).

The computation of the right hand side g requires the solution of one problem on each subdomain and a sparse matrix-vector product. In general, the explicit computation of the matrix C is expensive, since it requires the solution of $2n$ subproblems on each subdomain Ω_i, where n is the number of gridpoints on Γ. Also, since C is a dense n by n matrix, the solution of $Cu_3 = g$ by Gaussian elimination would require $\mathcal{O}(n^3)$ operations, which exceeds the complexity of solving the subdomain problems, when fast direct solvers of complexity $\mathcal{O}(n^2)$ or $\mathcal{O}(n^2 \log n)$ are applicable.

Instead of using a direct method for solving the interface system, iterative methods such as preconditioned conjugate gradients can be applied, for which only matrix-vector products of the form Cw for a given vector $w \in R^n$ are needed. From (1.5), we can see that the evaluation of Cw will take time comparable to computing the right hand side g, since it also requires the solution of two subdomain problems with boundary conditions on Γ given by w.

Since each iteration involves the solution of problems on the subdomains, keeping the number of iterations small is very important for the efficiency of the method. This can be achieved by choosing a good preconditioner for C. As is well known, a good preconditioner is an operator M that is a good approximation of C and can be easily inverted. In particular, if the condition number of $M^{-1}C$ is small, then M is a good preconditioner, because a bound on the rate of convergence of the iterative method depends on such condition number. Several preconditioners for C are given in the literature. Many have the desirable property of being spectrally equivalent to C. In other words, the condition number of $M^{-1}C$ is bounded *independent of the mesh size* $h = 1/(n+1)$, which means that the rate of convergence will not be affected when the mesh is refined. These preconditioners are described in Chapter 3.

In [12], Chan gives an exact eigenvalue decomposition of the capacitance matrix for the Laplacian operator, when the domain Ω is a rectangle divided into two strips. In this case, the capacitance matrix has the form

$$W \Lambda W^T \quad , \tag{1.6}$$

where Λ is diagonal and W is an orthogonal matrix such that Wu and $W^T u$, for $u \in R^n$, can be computed efficiently by using fast Fourier transforms. Chan [12] also noticed that the aspect ratios of the subdomains play an important role in the performance of the various earlier preconditioners. In general, the convergence properties of a preconditioner may deteriorate when the subdomains become narrow. This is not the case, however, for the decomposition (1.6), which is always exact for a rectangular domain divided into two – not necessarily equal – rectangular subdomains. For more general operators or domains, such decomposition of C is not exact, but it can still be applied as a preconditioner for C. This preconditioner takes aspect ratios into account and it does not deteriorate as one or both subdomains become narrow. For example, when this preconditioner is applied to the Poisson equation on an L-shaped domain, the condition number of the preconditioned capacitance system is bounded and the bound does not depend on the mesh size or the aspect ratios of the subdomains (see Chapter 5).

1.2 Thesis Outline

Much of the work of this thesis is based on the decomposition (1.6). The domain Ω is first approximated by the union of simpler domains and the differential operator is approximated by simpler operators, for which the interface system can be easily described and solved. Then, preconditioners for the original problem are designed based on the solution of the interface and the subdomain systems for the simplified problem. Consider, for example, the Poisson equation on the union of two rectangular regions, such as the T-shaped region of Fig. 1.2a. A preconditioner M_C for C is defined by first approximating Ω by a rectangle

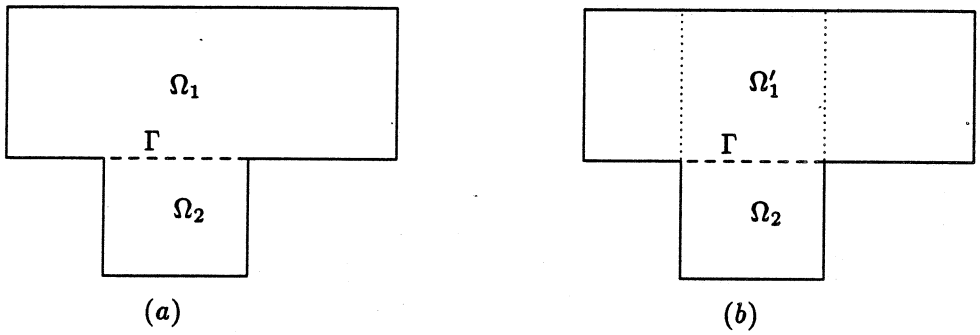

Figure 1.2: Decomposition of a T-shaped region

containing the interface Γ. Then, M_C is the capacitance matrix for the discrete Laplacian operator on the union of the two rectangles Ω_1' and Ω_2 of Fig. 1.2b. M_C has an exact eigenvalue decomposition of the form (1.6). Fast direct methods can be applied to solve the subproblems on Ω_1 and Ω_2. Since the rate of convergence of the iterative method with preconditioner M_C is bounded by a constant independent of h, $\mathcal{O}(\log \frac{1}{h})$ iterations are sufficient to achieve an accuracy of $\mathcal{O}(h^2)$, and therefore, the asymptotic complexity of this domain decomposed solver for the Poisson equation on the union of two rectangles is the same as the complexity of the subdomain solvers, multiplied by $\log \frac{1}{h}$.

It has been observed [13] that the method described is equivalent to the Schwarz's alternating procedure with maximum overlapping, when the iterative method is used to accelerate convergence, in the sense that at each iteration, both methods compute the same interface values on Γ. The non-overlapping method, however, does not need to double the work at the region of overlap.

In Chapter 5, we analyze the behavior of operators of the form (1.6) as preconditioners for irregular domains, in particular, L-shaped and C-shaped regions.

Variable coefficient operators can also be handled by approximating the coefficients by functions which may take different constant values on each subdomain. A preconditioner for the entire matrix A is given by an operator which has a decomposition of the form (1.4) with the constant coefficient approximations replacing the blocks A_{ij} and M_C replacing the Schur complement C. Exact decompositions in terms of Fourier modes will be derived in Chapter 4 for the interface system for a number of simple operators and domain splittings.

Multiple subdomains

The case of two subdomains and one interface is an interesting one, both from the theoretical and from the practical point of view. But if one wants to apply the idea of domain decomposition to the implementation of elliptic solvers on a parallel machine, one will often need to solve problems with many subdomains and many interfaces. It will, therefore, be necessary to define preconditioners for the interface system for the case of multiple subdomains and interfaces, without losing the good convergence properties that hold for the case of one interface. As we will show in future chapters, the extension is non-trivial, because a new ingredient is introduced with multiple interfaces: the coupling between the various interfaces becomes stronger as more subdomains are added and the interfaces become closer together. Suppose, for example, that we want to solve problem (1.1) on a rectangular domain Ω using a parallel system with k processing units, where $k > 2$. It is natural to consider the decomposition of the domain Ω into k subdomains and map each subdomain into one processor, so that the subdomain problems can be solved in parallel. We can consider two different ways of decomposing the domain: horizontal strips and boxes. Each type of splitting has its advantages and disadvantages. These will be discussed later in this thesis. Here we just mention the following: in the past, some authors have recommended against using stripwise decompositions, because the preconditioners proposed for that type of splittings did not take the coupling between interfaces into account [8, 28]. They therefore noticed that the preconditioners deteriorated as more subdomains were

added – obviously an undesirable property, when considering parallel implementations. Box-splitting, on the other hand, introduces a new class of unknowns: the cross-points. Bramble, Pasciak and Schatz [8] proposed a preconditioner for this kind of splitting. Although their preconditioner is quite efficient, it does not inherit the property of spectral equivalence from the case of one interface. Also, the rate of convergence improves when more subdomains are added, but at the cost of solving a coarse-grid problem for the cross-points which approximates the complexity of the original problem. The coupling between interfaces is only considered at these cross-points. In Chapter 4, we show that, for the case of constant or stepwise-constant coefficient operators and multistrip decompositions, the capacitance matrix can be described in terms of Fourier modes and it can be efficiently solved by fast direct methods. In other words, for some simple problems, the coupling is solved exactly. All operators and domain splittings for which we can derive exact decompositions are summarized in Chapter 4.

Parallel applications

As a direct consequence of the exact eigenvalue decomposition of the interface system, we present in Chapter 6, domain decomposed fast Poisson solvers on regular regions in two and three dimensions which can be easily implemented in parallel. Results on an Intel IPSC2 Hypercube multiprocessor system are shown and discussed.

In Chapter 7, a spectrally equivalent domain decomposed preconditioner is presented for a general self adjoint elliptic operator on a rectangular domain divided into strips. This preconditioner does not deteriorate when the subdomains become smaller and it can be efficiently implemented in parallel.

Chapter 2

Notation and Preliminary Definitions

In this chapter, we introduce the notation and definitions that will be used in the following chapters.

2.1 Preliminary Notation

Given a square non-singular matrix A, the *condition number* of A is defined as

$$\kappa(A) = \|A\|\|A^{-1}\| \quad .$$

If A is symmetric and the 2-norm is used, κ is called the spectral condition number and then we have:

$$\kappa(A) = \frac{|\lambda_{max}|}{|\lambda_{min}|} \quad ,$$

where λ_{max} and λ_{min} denote the eigenvalues of A with the largest and the smallest absolute values, respectively. The spectral radius of A is defined to be $\rho(A) = |\lambda_{max}|$ and $\sigma(A)$ denotes the spectrum of A.

If A is symmetric and positive definite (SPD) of dimension n, the A-norm of a vector $v \in R^n$ is defined as

$$\|v\|_A = \sqrt{v^T A v} \quad .$$

The matrix of sine modes of dimension n is the orthogonal matrix W whose elements are given by:

$$w_{ij} = \sqrt{\frac{2}{n+1}} \sin \frac{ij\pi}{n+1} \quad . \tag{2.1}$$

Given a vector $v \in R^n$, the product Wv is called the *discrete Sine Transform* of v and it can be computed with $\mathcal{O}(n \log n)$ operations by the application of Fast Fourier Transforms (FFT). Since $W^{-1} = W^T = W$, a system of the form $Wv = u$ can also be solved in $\mathcal{O}(n \log n)$ time.

2.2 Variational Iterative Methods

Next we describe the class of variational methods considered in this thesis. For every integer $m \geq 0$, define $\mathcal{P}_m$, the set of polynomials $P(x)$ of degree at most m such that $P(0) = 1$.

The iterative methods we will consider for the solution of a linear system $Ax = f$, where A is a SPD $n \times n$ matrix, are defined as follows (see, for example [11]): Given an initial guess x_0, at the i-th step compute x_i such that the $A^{\mu-2}$-norm of the residual $r_i = f - Ax_i$ is minimized over all vectors of the form $P(A)r_0$, with $P \in \mathcal{P}_i$, where r_0 is the initial residual.

The integer parameter μ characterizes the various methods in this family. For example, $\mu = 1$ gives the *conjugate gradient* (CG) method and $\mu = 2$ gives the *conjugate residual* (CR) method.

The preconditioned versions of these iterative methods solve the problem

$$\hat{A}\hat{x} = \hat{f} \quad ,$$

where $\hat{A} = M^{-1/2}AM^{-1/2}, \hat{x} = M^{1/2}x$, $\hat{f} = M^{-1/2}f$ and the matrix M — called the preconditioner — approximates A. At the i-th step, the preconditioned iterative method minimizes the $\hat{A}^{\mu-2}$-norm of the preconditioned residual $\hat{r}_i = \hat{f} - \hat{A}\hat{x}_i$.

Preconditioned Variational Algorithm:
Given an initial guess x_0, define the initial residual as
$r_0 = f - Ax_0$
and let
$\tilde{r}_0 = M^{-1}r_0$
$p_0 = \tilde{r}_0$
For $i = 0, 1, \ldots$ until convergence, do:
$a_i = \frac{(r_i,(M^{-1}A)^{\mu-1}\tilde{r}_i)}{(p_i,A(M^{-1}A)^{\mu-1}p_i)}$
$x_{i+1} = x_i + a_i p_i$
$r_{i+1} = r_i - a_i A p_i$
$\tilde{r}_{i+1} = \tilde{r}_i - a_i M^{-1} A p_i$
$b_i = \frac{(r_{i+1},(M^{-1}A)^{\mu-1}\tilde{r}_{i+1})}{(r_i,(M^{-1}A)^{\mu-1}\tilde{r}_i)}$
$p_{i+1} = \tilde{r}_{i+1} + b_i p_i$
enddo

If K denotes the condition number of the preconditioned matrix $M^{-1/2}AM^{-1/2}$ (assuming M symmetric and positive definite), then the iterates x_i satisfy (see e.g. [11]):

$$\|x_i - x\|_{A(M^{-1}A)^{\mu-1}} \leq 2\left(\frac{\sqrt{K}-1}{\sqrt{K}+1}\right)^i \|x_0 - x\|_{A(M^{-1}A)^{\mu-1}} \quad . \tag{2.2}$$

2.3 Finite Difference Discretizations

Consider the Dirichlet problem

$$Lu = f \qquad \text{in} \quad \Omega \tag{2.3}$$

$$u = u_b \qquad \text{on} \quad \partial\Omega \tag{2.4}$$

where L is the following linear self-adjoint elliptic differential operator:

$$L \equiv -\sum_{i=1}^{d} \frac{\partial}{\partial x_i}\left(a_i(x_1,\ldots,x_d)\frac{\partial}{\partial x_i}\right) + c(x_1,\ldots,x_d)$$

and Ω will typically be a regular two-dimensional ($d = 2$) or three-dimensional ($d = 3$) domain or a union of regular domains (rectangles or boxes). When $a_i(x) = 1$ and $c(x) = 0$, L is the Laplacian operator.

In this thesis we will only consider five-point finite-difference discretizations on regular grids for two dimensional problems and seven-point discretizations for three dimensional problems. For a given mesh width h, the terms of L are approximated at a gridpoint $(x_1,\ldots,x_d)$ by

$$\frac{\partial}{\partial x_i}\left(a_i(x_1,\ldots,x_d)\frac{\partial}{\partial x_i}u\right) \approx \frac{1}{h^2}\left(a_{i+}\left(u_{i+} - u\right) - a_{i-}\left(u - u_{i-}\right)\right) \quad , \tag{2.5}$$

where $a_{i\pm} = a_i(x_1,\ldots x_i \pm \frac{h}{2}, x_d)$, u denotes $u(x_1,\ldots x_i, x_d)$ and $u_{i\pm}$ denotes $u(x_1,\ldots x_i \pm h, x_d)$.

By applying (2.5) and after multiplying by h^2, the discretization of (2.4) on a finite grid leads to a linear system of the form

$$Au = f \quad , \tag{2.6}$$

where u denotes now the vector of approximated values for the solution $u(x,y)$ at the gridpoints and the vector f is computed using the values of $f(x,y)$ and the boundary conditions u_b.

The discretization of the one dimensional Laplacian operator on the interval $(0,1)$ leads to the matrix $-K$, where

$$K = \begin{pmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix} \quad . \tag{2.7}$$

The eigenvalue decomposition of K is given by $K = W^T \mathrm{diag}(\sigma_j) W$, where $\sigma_j = 4 \sin^2 j\frac{\pi}{2}h$.

When $d = 2$ and Ω is a rectangle, a five-point discretization of (2.4) on an n by m grid leads to a block-tridiagonal system with m blocks of dimension n. For the constant coefficient case, where

$$L \equiv -a \frac{\partial^2}{\partial x^2} - b \frac{\partial^2}{\partial y^2} + c \quad , \tag{2.8}$$

we have

$$A = \begin{pmatrix} T & -bI & & & \\ -bI & T & -bI & & \\ & \ddots & \ddots & \ddots & \\ & & -bI & T & -bI \\ & & & -bI & T \end{pmatrix} \quad , \tag{2.9}$$

where $T = (2b + h^2 c)I + aK$. The diagonal blocks and the off-diagonal blocks of A are diagonalizable by the matrix W. In particular, the five-point discretization for the two-dimensional Laplacian operator on a rectangular n by m grid has the following block tridiagonal form (m blocks):

$$\begin{pmatrix} -2I - K & I & & & \\ I & -2I - K & \ddots & & \\ & \ddots & \ddots & & \\ & & & -2I - K & I \\ & & & I & -2I - K \end{pmatrix} \quad .$$

We will also use five-point discretization formulas for cases where the coefficients are piecewise continuous, with jumps along horizontal or vertical interfaces. Continuity conditions require that the solution $u(x, y)$ be continuous in Ω, that $b(x, y)u_y(x, y)$ be continuous along horizontal interfaces and that $a(x, y)u_x(x, y)$ be continuous along vertical interfaces. Following Varga's [35, pp. 190-191] description of five-point approximations, we can see that the same formulas given for continuous coefficients can be applied in this case, as long as the coefficients are redefined by arithmetic averages at the interfaces. If (x, y) is a point on a horizontal interface, for instance, we redefine:

$$a(x, y) \equiv \frac{1}{2}(\lim_{y \to y^-} a(x, y) + \lim_{y \to y^+} a(x, y)) \quad , \tag{2.10}$$

similarly, on a vertical interface,

$$b(x, y) \equiv \frac{1}{2}(\lim_{x \to x^-} b(x, y) + \lim_{x \to x^+} b(x, y)) \tag{2.11}$$

and if (x, y) is the cross point between a horizontal and a vertical interface, we have:

$$c(x, y) \equiv \frac{1}{4}(\lim_{\substack{x \to x^- \\ y \to y^-}} c(x, y) + \lim_{\substack{x \to x^- \\ y \to y^+}} c(x, y) + \lim_{\substack{x \to x^+ \\ y \to y^-}} c(x, y) + \lim_{\substack{x \to x^+ \\ y \to y^+}} c(x, y)) \quad . \tag{2.12}$$

Suppose, for example, that Ω is a rectangle partitioned into two strips Ω_1 and Ω_2 and the coefficients of L take constant values a_i, b_i and c_i inside each strip. The discretization matrix for this problem has the form

$$\begin{pmatrix} T_1 & -b_1 I & & & & & & & \\ -b_1 I & T_1 & \ddots & & & & & & \\ & \ddots & \ddots & -b_1 I & & & & & \\ & & -b_1 I & T_1 & -b_1 I & & & & \\ & & & -b_1 I & T_3 & -b_2 I & & & \\ & & & & -b_2 I & T_2 & -b_2 I & & \\ & & & & & b_2 I & T_2 & \ddots & \\ & & & & & & \ddots & \ddots & -b_2 I \\ & & & & & & & -b_2 I & T_2 \end{pmatrix}$$

with $T_1 = (2b_1 + h^2c_1)I + a_1K$, $T_2 = (2b_2 + h^2c_2)I + a_2K$ and

$$T_3 = \left(b_1 + b_2 + h^2\frac{(c_1 + c_2)}{2}\right)I + \frac{(a_1 + a_2)}{2}K \quad .$$

2.4 Notation Concerning the Domain Decomposition Method

Consider the system (2.6) which represents the discretization of an elliptic PDE on a domain Ω divided into k subdomains Ω_i.

The number of gridpoints on a given interface will generally be denoted by n for one dimensional interfaces and n by p or n by m for two dimensional interfaces, unless explicitly stated otherwise. The mesh width will be denoted by h and usually, $h = 1/(n+1)$.

Let Γ be the union of all separators between subdomains. If the vectors u and f are reordered such that the points in the interior of the subdomains are numbered first, then (2.6) can be written in block form as:

$$\begin{pmatrix} A_\Omega & P \\ P^T & A_\Gamma \end{pmatrix}\begin{pmatrix} u_\Omega \\ u_\Gamma \end{pmatrix} = \begin{pmatrix} f_\Omega \\ f_\Gamma \end{pmatrix} \quad . \tag{2.13}$$

For the discretizations considered in this thesis, there is no coupling between the interior points of two different subdomains. Therefore, the submatrix A_Ω is block diagonal:

$$A_\Omega = \begin{pmatrix} A_{\Omega_1} & & & \\ & A_{\Omega_2} & & \\ & & \ddots & \\ & & & A_{\Omega_k} \end{pmatrix} \quad ,$$

where each block A_{Ω_i} represents the discretization of the differential operator on the interior of the i-th subdomain. A_Γ represents the restriction of the operator to the interfaces and P represents the coupling between subdomains and interfaces. The capacitance matrix or interface matrix is the Schur complement of A_Γ in A, i.e.:

$$C = A_\Gamma - P^T A_\Omega^{-1} P \quad . \tag{2.14}$$

By applying block elimination to (2.13), we have the following domain decomposition algorithm (note that for the case of two subdomains with one interface, this algorithm is equivalent to Algorithm BGE of page 2):

ALGORITHM DD:
Step 1: Compute $g = f_\Gamma - P^T A_\Omega^{-1} f_\Omega$.
Step 2: Solve $Cu_\Gamma = g$.
Step 3: Solve $A_\Omega u_\Omega = f_\Omega - Pu_\Gamma$.

2.5 Interaction Between Boundaries of a Subdomain

In this section we describe the operators which contribute to the term $P^T A_\Omega^{-1} P$ in (2.14). Generally speaking, they represent the interaction between two interfaces of a given subdomain.

Consider the rectangular region R of Fig. 2.1, with edges Γ_N, Γ_E, Γ_S and Γ_W. This region R represents a generic rectangular subdomain in the domain Ω. Let n_1 be the number of gridpoints in Γ_N (or Γ_S) and n_2 the number of gridpoints in Γ_E (or Γ_W). The corner points are not included in the edges, since they may or may not be interior to Ω.

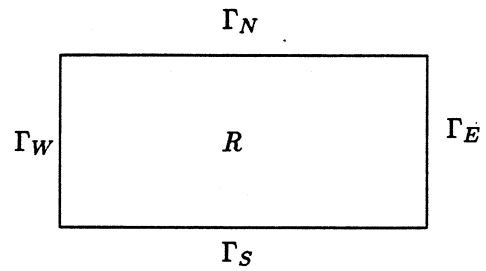

Figure 2.1: Interaction between interfaces: a rectangular subdomain.

Let (2.6) be the linear system which represents a five-point discretization of the differential equation in Ω. Let A_R be the submatrix of A corresponding to the interior points of R, where the gridpoints are numbered according to the natural ordering, i.e., row-wise. A_R is the restriction of the discrete differential operator to the region R. Suppose that the interface Γ_k, for $k = N, S, E$ or W, is interior to Ω and define P_k to be the submatrix of A that represents the coupling between gridpoints of R and gridpoints on Γ_k. In Fig. 1.1, for example, if R was given by subdomain Ω_1, then Γ_S would correspond to the interface Γ in the picture, A_R would be the submatrix A_{11} in (1.3) and $P_S = A_{13}$.

When block elimination is applied, the Schur complement for the interface contains non-zero blocks corresponding to the interaction between edges. Define the operator Q_{kl} as:

$$Q_{kl} = P_k^T A_R^{-1} P_l \tag{2.15}$$

In the case of the Poisson equation, for example, the operator Q_{kl} takes boundary values on the edge Γ_l and computes the solution to the Poisson equation on the gridpoints adjacent to the edge Γ_k. For constant coefficient operators of the form (2.8), it is possible to describe Q_{kl} in terms of Fourier modes.

Given $\mu \geq 0$, define

$$\gamma(\mu) = \left(1 + \frac{\mu}{2} - \sqrt{\frac{\mu^2}{4} + \mu}\right)^2 . \tag{2.16}$$

It is easy to show that the function γ has the following properties, for all $\mu \geq 0$:

$$\frac{1}{\sqrt{\gamma(\mu)}} = 1 + \frac{\mu}{2} + \sqrt{\frac{\mu^2}{4} + \mu} \tag{2.17}$$

$$\sqrt{\gamma(\mu)} + \frac{1}{\sqrt{\gamma(\mu)}} = 2 + \mu \tag{2.18}$$

$$\sqrt{\gamma(\mu)} - \frac{1}{\sqrt{\gamma(\mu)}} = -2\sqrt{\frac{\mu^2}{4} + \mu} \tag{2.19}$$

The following lemma will be used in the proofs that follow:

Lemma 2.1 *Let $\beta > 2$ and consider the following tridiagonal system of size n:*

$$\begin{pmatrix} -\beta & 1 & & \\ 1 & -\beta & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -\beta \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix} .$$

The solution x is given by

$$x_k = -\sqrt{\gamma}^{\,n+1-k} \frac{1 - \gamma^k}{1 - \gamma^{n+1}} \tag{2.20}$$

where $\gamma = \gamma(\beta - 2)$.

Proof: The elements of x satisfy the following difference equation

$$x_{k-1} - \beta x_k + x_{k+1} = 0$$

with boundary conditions $x_0 = 0$ and $x_{n+1} = -1$. Therefore,

$$x_k = c_1 r_+^k + c_2 r_-^k \quad ,$$

where $r_\pm$ are the roots of the characteristic polynomial $z^2 - \beta z + 1$, i.e.

$$r_\pm = \frac{\beta}{2} \pm \sqrt{\frac{\beta^2}{4} - 1}$$

and the constants c_1 and c_2 are determined by the boundary conditions, therefore $c_1 + c_2 = 0$ and $c_1 = \frac{-1}{r_+^{n+1} - r_-^{n+1}}$. Since $r_- = \sqrt{\gamma(\beta-2)}$ and $r_+ = 1/\sqrt{\gamma(\beta-2)}$, we get (2.20). ∎

In order to simplify the notation, we will use direct (or tensor) products to define the various operators and we will apply the following two properties:

$$i \;) \quad (X \otimes Y)^T = X^T \otimes Y^T \tag{2.21}$$

$$ii \;) \quad (X_1 \otimes Y_1)(X_2 \otimes Y_2) = (X_1 X_2) \otimes (Y_1 Y_2) \tag{2.22}$$

Let I_{n_i}, for $i = 1, 2$ denote the identity matrix of dimension n_i and $e_j^{(i)}$, the j-th column of I_{n_i}. Also, given n define

$$\sigma_j(n) = 4 \sin^2 \frac{j\pi}{2(n+1)} \tag{2.23}$$

for $j = 1, \ldots, n$.

Suppose that A_R has the following block-tridiagonal form:

$$A_R = \begin{pmatrix} T & -dI_{n_1} & & \\ -dI_{n_1} & T & \ddots & \\ & \ddots & \ddots & -dI_{n_1} \\ & & -dI_{n_1} & T \end{pmatrix} \tag{2.24}$$

with n_2 blocks of dimension n_1, where $W_{n_1} T W_{n_1} = \mathrm{diag}(t_1, \ldots, t_{n_1})$. For example, for the case of the operator (2.8), A_R is given by (2.9), i.e. $T = (2b + h^2 c)I + aK$, $d = b$ and the eigenvalues of T are $t_j = 2b + h^2 c + a\sigma_j(n_1)$. The expression (2.24) corresponds to a row-wise numbering of the gridpoints. Suppose also that, for a column-wise numbering of the gridpoints, A_R has a block-tridiagonal form with n_1 blocks of dimension n_2, with $\tilde{T} = W_{n_2} \mathrm{diag}(\tilde{t}_1, \ldots, \tilde{t}_{n_2}) W_{n_2}$ on the diagonal and $-\tilde{d} I_{n_2}$ on the off-diagonals. For the case of the operator (2.8), we have $\tilde{d} = a$ and $\tilde{t}_i = 2a + h^2 c + b\sigma_i(n_2)$.

Also, let the submatrices P_k be given by

$$P_N = -d_N \, e_1^{(2)} \otimes I_{n_1} \tag{2.25}$$

$$P_S = -d_S \, e_{n_2}^{(2)} \otimes I_{n_1} \tag{2.26}$$

$$P_W = -d_W \, I_{n_2} \otimes e_1^{(1)} \tag{2.27}$$

$$P_E = -d_E \, I_{n_2} \otimes e_{n_1}^{(1)} \tag{2.28}$$

Interaction between parallel edges

When Γ_k and Γ_l are opposite edges or when $k = l$, the operator Q_{kl} is diagonalizable by Fourier modes.

Lemma 2.2 *Let A_R, P_N and P_S be given by (2.24), (2.25) and (2.26) respectively and let Q_{NS} represent the coupling between interfaces Γ_N and Γ_S, i.e. $Q_{NS} = P_N^T A_R^{-1} P_S$. Then the matrix*

$$W_{n_1} Q_{NS} W_{n_1} = D_{NS}$$

of dimension n_1, is diagonal with diagonal entries given by

$$d_{jj}^{NS} = \frac{d_N d_S}{d} \sqrt{\gamma(\mu_j)^{n_2}} \left(\frac{1-\gamma(\mu_j)}{1-\gamma(\mu_j)^{n_2+1}} \right) \tag{2.29}$$

where $\mu_j = \frac{t_j}{d} - 2$. Similarly, $W_{n_2} Q_{EW} W_{n_2} = D_{EW}$ is diagonal, with

$$d_{ii}^{EW} = \frac{d_E d_W}{\tilde{d}} \sqrt{\gamma(\tilde{\mu}_i)^{n_1}} \left(\frac{1-\gamma(\tilde{\mu}_i)}{1-\gamma(\tilde{\mu}_i)^{n_1+1}} \right) \quad ,$$

where $\tilde{\mu}_i = \frac{\tilde{t}_i}{\tilde{d}} - 2$.

Also, $W_{n_1} Q_{NN} W_{n_1} = D_{NN}$, $W_{n_1} Q_{SS} W_{n_1} = D_{SS}$, $W_{n_2} Q_{EE} W_{n_2} = D_{EE}$ and $W_{n_2} Q_{WW} W_{n_2} = D_{WW}$ are diagonal, with

$$d_{jj}^{NN} = \frac{d_N^2}{d} \sqrt{\gamma(\mu_j)} \left(\frac{1-\gamma(\mu_j)^{n_2}}{1-\gamma(\mu_j)^{n_2+1}} \right) \quad , \tag{2.30}$$

$$d_{jj}^{SS} = \frac{d_S^2}{d} \sqrt{\gamma(\mu_j)} \left(\frac{1-\gamma(\mu_j)^{n_2}}{1-\gamma(\mu_j)^{n_2+1}} \right) \quad , \tag{2.31}$$

$$d_{ii}^{EE} = \frac{d_E^2}{\tilde{d}} \sqrt{\gamma(\tilde{\mu}_i)} \left(\frac{1-\gamma(\tilde{\mu}_i)^{n_1}}{1-\gamma(\tilde{\mu}_i)^{n_1+1}} \right) \quad , \tag{2.32}$$

$$d_{ii}^{WW} = \frac{d_W^2}{\tilde{d}} \sqrt{\gamma(\tilde{\mu}_i)} \left(\frac{1-\gamma(\tilde{\mu}_i)^{n_1}}{1-\gamma(\tilde{\mu}_i)^{n_1+1}} \right) \quad . \tag{2.33}$$

Proof: These expressions were originally derived in [12] and [15] for the Poisson equation, with $d_N = d_S = d_E = d_W = d = 1$, $t_j = 2 + \sigma_j(n_1)$ and $\tilde{t}_i = 2 + \sigma_i(n_2)$.

By using properties (2.21) and (2.22), we can prove that

$$\begin{aligned} W_{n_1} P_N^T &= -d_N (e_1^{(2)} \otimes I_{n_1})^T (I_{n_2} \otimes W_{n_1}) \\ P_S^T W_{n_1} &= -d_S (I_{n_2} \otimes W_{n_1})(e_{n_2}^{(2)} \otimes I_{n_1}) \quad . \end{aligned}$$

Then we can see that

$$W_{n_1} Q_{NS} W_{n_1} = -\frac{d_N d_S}{d} (e_1^{(2)} \otimes I_{n_1})^T \begin{pmatrix} D & I_{n_1} & & \\ I_{n_1} & D & & \\ & & \ddots & I_{n_1} \\ & & I_{n_1} & D \end{pmatrix}^{-1} (e_{n_2}^{(2)} \otimes I_{n_1})$$

where $D = \mathrm{diag}(-t_j/d)$. If the equations in the last expression are rearranged according to a column-wise ordering of the gridpoints in R, we get:

$$W_{n_1} Q_{NS} W_{n_1} = -\frac{d_N d_S}{d} (I_{n_1} \otimes e_1^{(2)})^T \begin{pmatrix} T_1 & & & \\ & T_2 & & \\ & & \ddots & \\ & & & T_{n_1} \end{pmatrix}^{-1} (I_{n_1} \otimes e_{n_2}^{(2)}) \quad ,$$

where $T_j = \mathrm{tridiag}(1, -t_j/d, 1)$. Therefore, $W_{n_1} Q_{NS} W_{n_1}$ is diagonal and its diagonal elements are given by

$$-\frac{d_N d_S}{d} e_1^{(2)^T} T_j^{-1} e_{n_2}^{(2)} \quad .$$

Since $e_1^{(2)^T} T_j^{-1} e_{n_2}^{(2)}$ corresponds to the first element x_1 of the solution to $T_j x = e_{n_2}^{(2)}$, by applying Lemma 2.1 we can prove that

$$x_1 = -\sqrt{\gamma(\mu_j)}^{\,n_2} \frac{1-\gamma(\mu_j)}{1-\gamma(\mu_j)^{n_2+1}}$$

Therefore, we have (2.29).

Similarly, we can prove that $W_{n_1} Q_{NN} W_{n_1}$ is diagonal and its diagonal elements are given by

$$-\frac{d_N^2}{d} e_{n_2}^{(2)^T} T_j^{-1} e_{n_2}^{(2)} \quad ,$$

and $e_{n_2}^{(2)^T} T_j^{-1} e_{n_2}^{(2)}$ corresponds to the last element of x, thus proving (2.30).

The expressions involving Γ_E and Γ_W can be derived analogously, by using the block form for A_R corresponding to a column-wise numbering of the nodes. ∎

Interactions between perpendicular edges

Operators like Q_{NW} or Q_{NE}, on the other hand, which represent the interaction between perpendicular edges, are not diagonalizable by Fourier modes. Moreover, they are, in general not square, but n_1 by n_2 rectangular matrices. It is possible, however, to describe the elements of the matrices $W_{n_1} Q_{NW} W_{n_2}$ and $W_{n_1} Q_{NE} W_{n_2}$ for constant coefficient cases.

Lemma 2.3 *If A_R, P_N and P_S are given by (2.24), (2.25) and (2.26) respectively and Q_{NW}, defined as in (2.15), represents the coupling between interfaces Γ_N and Γ_W, then the elements of*

$$\hat{Q}_{NW} = W_{n_1} Q_{NW} W_{n_2}$$

are given by

$$q_{ij}^{NW} = \frac{2 d_N d_W}{\sqrt{(n_1+1)(n_2+1)}} \frac{\sin\frac{i\pi}{n_1+1} \quad \sin\frac{j\pi}{n_2+1}}{d(2+\sigma_j(n_2)) + t_i} \tag{2.34}$$

Similarly, the elements of the matrix

$$\hat{Q}_{NE} = W_{n_1} Q_{NE} W_{n_2}$$

are given by

$$q_{ij}^{NE} = \frac{2 d_N d_E}{\sqrt{(n_1+1)(n_2+1)}} \frac{\sin\frac{n_1 i\pi}{n_1+1} \quad \sin\frac{j\pi}{n_2+1}}{d(2+\sigma_j(n_2)) + t_i}$$

Proof: If u_i is an eigenvector of T corresponding to the eigenvalue t_i and v_j is the j-th column of W_{n_2}, then we can see that $A_R(v_j \otimes u_i) = \lambda(v_j \otimes u_i)$, with $\lambda = d(2+\sigma_j(n_2)) + t_i$. Therefore, the eigenvalue decomposition of A_R is given by

$$A_R = (W_{n_2} \otimes W_{n_1}) \quad \Lambda \quad (W_{n_2} \otimes W_{n_1}) \tag{2.35}$$

where Λ is the $n_1 n_2 \times n_1 n_2$ diagonal matrix:

$$\Lambda = I_{n_2} \otimes \mathrm{diag}(t_i) + \mathrm{diag}(d(2+\sigma_j(n_2))) \otimes I_{n_1} \quad .$$

in other words, the eigenvalues of A_R are

$$\lambda_J = d(2+\sigma_j(n_2)) + t_i \quad ,$$

with $J = (j-1)n_1 + i$. It is easy to verify, for example, that the eigenvalues for the discrete Laplacian are given by $-\sigma_i(n_1) - \sigma_j(n_2)$.

By substituting (2.35) and (2.27) in (2.15) and then applying (2.21) and (2.22), we have

$$Q_{NW} = d_N d_W \left((e_1^{(2)T} W_{n_2}) \otimes W_{n_1} \right) \Lambda^{-1} \left(W_{n_2} \otimes (W_{n_1} e_1^{(1)}) \right)$$

and therefore,

$$\hat{Q}_{NW} = d_N d_W \left((e_1^{(2)T} W_{n_2}) \otimes I_{n_1} \right) \Lambda^{-1} \left(I_{n_2} \otimes (W_{n_1} e_1^{(1)}) \right) \quad .$$

Then, the j-th column of $\hat{Q}_{NW}$ is given by

$$d_N d_W \sqrt{\frac{2}{n_2+1}} \sin\frac{j\pi}{n_2+1} \left(d\sigma_j(n_2)\, I_{n_1} + \mathrm{diag}(t_i/d) \right)^{-1} W_{n_1}\, e_1^{(1)} \quad ,$$

and the i-th element of this column is

$$q_{ij}^{NW} = d_N d_W \sqrt{\frac{2}{n_2+1}} \sin\frac{j\pi}{n_2+1} \frac{1}{d(2+\sigma_j(n_2)) + t_i} \sqrt{\frac{2}{n_1+1}} \sin\frac{i\pi}{n_1+1} \quad ,$$

from which (2.34) follows.

Similarly, q_{ij}^{NE} can be derived by using (2.28) instead of (2.27). ∎

2.6 About Schur Complements and Preconditioned Iterative Methods

Consider the following symmetric positive definite system, written in block form:

$$\begin{pmatrix} A & B \\ B^T & C \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix} \quad , \tag{2.36}$$

where the blocks A and C are square SPD matrices, and the reduced systems:

$$(A - BC^{-1}B^T)x = f - BC^{-1}g \tag{2.37}$$

$$(C - B^T A^{-1} B)y = g - B^T A^{-1} f \quad . \tag{2.38}$$

Suppose that efficient solvers exist for the blocks A and C. Then A can be used as preconditioner in the solution of (2.37) and y can then be computed by solving $Cy = g - B^T x$. Or we could instead use C as preconditioner in the solution of (2.38) and then solve $Ax = f - By$. Although in general the two procedures involve the iterative solution of systems of different dimensions, we will see that they have equivalent convergence properties. We will first show this result from the point of view of fixed point iterations and then, for the case of PCG like methods.

Theorem 2.1 *Consider the solution of (2.37) by the following fixed-point iteration, with splitting matrix A: given an initial guess x^0, define the i-th iterate as the solution to:*

$$Ax^i = f - BC^{-1}g + BC^{-1}B^T x^{i-1} \quad . \tag{2.39}$$

Similarly, given y^0, define the i-th iterate of a fixed-point iteration for solving (2.38) with splitting matrix C, as:

$$Cy^i = g - B^T A^{-1} f + B^T A^{-1} B y^{i-1} \quad . \tag{2.40}$$

Then, the two iterations converge. Moreover, they are equivalent *in the sense that for any given initial guess x^0 for (2.39), there exists an initial guess y^0 for (2.40), such that for all i we have:*

$$y^i = C^{-1}(g - B^T x^i) \tag{2.41}$$

and

$$\|e_x^{i+1}\|_A \leq \|e_y^i\|_C \leq \|e_x^i\|_A \quad , \tag{2.42}$$

where $e_x^i = x^i - x$ and $e_y^i = y^i - y$.

Completely analogous results hold for x^i and e_x^i, given an initial guess y^0 for (2.38).

Proof: Given x^0, define $y^0 = C^{-1}g - C^{-1}B^T x^0$. By induction, we can see that (2.41) satisfies (2.40) for every $i \geq 1$. The blocks A and C are SPD, then so are their corresponding Schur complements. It follows that $\|A^{-1}BC^{-1}B^T\|_2 < 1$ and $\|C^{-1}B^T A^{-1}B\|_2 < 1$. Therefore, the two iterations converge. Also,

$$\|A^{-1/2}BC^{-1/2}\|_2 < 1 \quad . \tag{2.43}$$

Moreover,

$$\begin{aligned} Ae_x^{i+1} &= BC^{-1}B^T e_x^i \\ e_y^i &= -C^{-1}B^T e_x^i \quad . \end{aligned}$$

Therefore, we have

$$A^{1/2}e_x^{i+1} = -(A^{-1/2}BC^{-1/2})C^{1/2}e_y^i \tag{2.44}$$

and

$$C^{1/2}e_y^i = -(C^{-1/2}B^T A^{-1/2})A^{1/2}e_x^i \quad . \tag{2.45}$$

(2.42) follows from (2.43), (2.44) and (2.45). ∎

Suppose that an iterative method such as PCG is applied to (2.37) with preconditioner A and to (2.38) with preconditioner C. Then the last theorem suggests that we get the same convergence rate.

The preconditioned Schur complements can be written as

$$\hat{S}_A \equiv A^{-1/2}(A - BC^{-1}B^T)A^{-1/2} = I - XX^T \tag{2.46}$$

$$\hat{S}_C \equiv C^{-1/2}(C - B^T A^{-1}B)C^{-1/2} = I - X^T X \quad , \tag{2.47}$$

where $X = A^{-1/2}BC^{-1/2}$. By (2.43) we know that $\|X\|_2 < 1$. Then, we can easily prove the following lemma:

Lemma 2.4 *The condition numbers of the preconditioned Schur complements $\hat{S}_A$ and $\hat{S}_C$ are both bounded by*

$$\frac{1}{1 - \|X\|^2} \quad .$$

∎

Furthermore, we will show that $\hat{S}_A$ and $\hat{S}_C$ are also equivalent in their eigenvalue structure. If X is square, it is easy to see that $X^T X$ and XX^T have the same eigenvalues. If X is not square, $X^T X$ and XX^T have the same eigenvalues, except for $\lambda = 0$, which may be an eigenvalue of one of them, but not of the other. According to the next lemma, this implies that both iterations are equivalent.

Lemma 2.5 *Consider the systems*

$$Mx = f \tag{2.48}$$
$$Ny = g \quad , \tag{2.49}$$

where the matrices M, of order m and N, of order n, are symmetric and positive definite.
i) *Suppose that $m = n$ and*

$$M = U\Lambda U^T$$
$$N = V\Lambda V^T \quad ,$$

where U and V are orthogonal matrices (i.e., M and N have the same eigenvalues). Then, given an initial residual r_0^M for (2.48) there exists an initial residual r_0^N for (2.49) such that, at each iteration,

$$\|r_i^M\|_{M^{\mu-2}} = \|r_i^N\|_{N^{\mu-2}} \quad . \tag{2.50}$$

In particular, initial guesses can always be found such that the same number of iterations will be required for solving systems (2.48) and (2.49) to any specified accuracy.
ii) *Suppose that $\rho(M) < 2$, $m < n$ and $\sigma(N) = \sigma(M) \bigcup \{1\}$. Then, given an initial residual r_0^M for (2.48), there exists an initial residual r_0^N for (2.49) such that, at each iteration*

$$\|r_i^M\|_{M^{\mu-2}} = \|r_i^N\|_{N^{\mu-2}} \quad . \tag{2.51}$$

And given r_0^N, there exists r_0^M such that, for all i,

$$\|r_{i+1}^N\|_{N^{\mu-2}} \leq \|r_i^M\|_{M^{\mu-2}} \leq \|r_i^N\|_{N^{\mu-2}} \quad . \tag{2.52}$$

Proof:
i) We can write M as $M = UV^T NVU^T$ and also, for every polynomial P we have $P(M) = UV^T P(N)VU^T$. Given $r_0^M \in R^m$, define $r_0^N = VU^T r_0^M$. Then, for every P we have:

$$\begin{aligned} \|P(M)r_0^M\|^2_{M^{\mu-2}} &= (r_0^M)^T P(M)^T M^{\mu-2} P(M) r_0^M \\ &= (r_0^M)^T UV^T P(N)^T N^{\mu-2} P(N) VU^T r_0^M \\ &= (r_0^N)^T P(N)^T N^{\mu-2} P(N) r_0^N = \|P(N)r_0^N\|^2_{N^{\mu-2}} \quad . \end{aligned}$$

Therefore, we can say that a polynomial P_i minimizes $\|P(M)r_0^M\|^2_{M^{\mu-2}}$ over all $P \in \mathcal{P}_i$ if and only if it minimizes $\|P(N)r_0^N\|^2_{N^{\mu-2}}$ and moreover, $\|P_i(M)r_0^M\|^2_{M^{\mu-2}} = \|P_i(N)r_0^N\|^2_{N^{\mu-2}}$, which proves (2.50).

ii) By i, two matrices with the same eigenvalues have equivalent convergence properties. Therefore, without loss of generality we may assume that:

$$N = \begin{pmatrix} M & 0 \\ 0 & I \end{pmatrix} \quad ,$$

where I is the identity matrix of dimension $n - m$.

Given $r_0^M \in R^n$, we define the initial residual r_0^N to be:

$$r_0^N = \begin{pmatrix} r_0^M \\ 0 \end{pmatrix} \quad ,$$

where 0 represents a zero vector of dimension $n - m$. Then, for every polynomial $P \in \mathcal{P}_i$, we have

$$P(N)r_0^N = \begin{pmatrix} P(M)r_0^M \\ 0 \end{pmatrix}$$

and

$$\|P(N)r_0^N\|_{N^{\mu-2}} = \|P(M)r_0^M\|_{M^{\mu-2}} \quad .$$

Therefore, P minimizes $\|P(N)r_0^N\|_{N^{\mu-2}}$ if and only if it minimizes $\|P(M)r_0^M\|_{M^{\mu-2}}$, which proves equation (2.51).

On the other hand, given an initial residual $r_0^N \in R^n$ for (2.49) such that

$$r_0^N = \begin{pmatrix} r_{01}^N \\ r_{02}^N \end{pmatrix}$$

with $r_{01}^N \in R^m$ and $r_{02}^N \in R^{n-m}$, define the initial residual for (2.48) as $r_0^M = r_{01}^N$. For all i there exists a polynomial $Q \in \mathcal{P}_i$ such that

$$r_i^M = Q(M)r_0^M \quad .$$

Define $\tilde{Q}(x) = (1-x)Q(x)$, then $\tilde{Q} \in \mathcal{P}_{i+1}$ and

$$\|\tilde{Q}(N)r_0^N\|_{N^{\mu-2}}^2 = \|\tilde{Q}(M)r_0^M\|_{M^{\mu-2}}^2 = \|Q(M)r_0^M\|_{M^{\mu-2}}^2 - y^T(2I-M)y \quad ,$$

for $y = M^{(\mu-1)/2}Q(M)r_0^M$. Since all eigenvalues of M are bounded by two, $2I - M$ is positive definite. Then,

$$\|\tilde{Q}(N)r_0^N\|_{N^\mu}^2 \leq \|Q(M)r_0^M\|_{M^\mu}^2 = \|r_i^M\|_{M^\mu}^2 \quad . \tag{2.53}$$

Since r_{i+1}^N minimizes the $N^{\mu-2}$-norm of the residual over all vectors of the form $R(N)r_0^N$ for $R \in \mathcal{P}_{i+1}$, we have

$$\|r_{i+1}^N\|_{N^{\mu-2}} \leq \|\tilde{Q}(N)r_0^N\|_{N^{\mu-2}} \quad . \tag{2.54}$$

Also, there exists a polynomial $P \in \mathcal{P}_i$ such that

$$r_i^N = P(N)r_0^N \quad .$$

Then,

$$\begin{aligned} \|P(M)r_0^M\|_{M^{\mu-2}}^2 &= \|P(N)r_0^N\|_{N^{\mu-2}}^2 - \|P(I)r_0^{N^2}\|_2^2 \\ &\leq \|P(N)r_0^N\|_{N^{\mu-2}}^2 = \|r_i^N\|_{N^{\mu-2}}^2 \quad . \end{aligned} \tag{2.55}$$

(2.52) follows from (2.53), (2.54) and (2.55). ∎

This lemma proves that initial residuals always exist such that the number of iterations required to solve (2.48) and (2.49) to a specified accuracy differ at most by one.

Since the preconditioned Schur complements (2.46) and (2.47) satisfy hypotheses i or ii of Lemma 2.5, we can prove the following:

Theorem 2.2 *Consider the following symmetric positive definite system:*

$$\begin{pmatrix} A & B \\ B^T & C \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix} \quad ,$$

where A and C are square SPD matrices. Suppose that I_1 iterations of PCG are applied to the reduced system

$$(A - BC^{-1}B^T)x = f - BC^{-1}g$$

with preconditioner A and initial residual $r_0^{(1)}$. Then there exists an initial residual $r_0^{(2)}$ such that I_2 PCG iterations are required to solve the reduced system

$$(C - B^TA^{-1}B)y = g - B^TA^{-1}f$$

with preconditioner C and residual $r_{I_2}^{(2)}$ no greater than $r_{I_1}^{(1)}$, with $|I_1 - I_2| \leq 1$.

Proof: Since the preconditioned Schur complements $A^{-1/2}(A-BC^{-1}B^T)A^{-1/2}$ and $C^{-1/2}(C-B^TA^{-1}B)C^{-1/2}$ are SPD and they are of the form $I - XX^T$ and $I - X^TX$, all their eigenvalues are bounded by one. If $n = m$, the eigenvalues of X^TX and XX^T are the same. Otherwise assume, for example, that $n > m$. Then the eigenvalues of X^TX and XX^T are the same, except that $\lambda = 0$ may be an eigenvalue of one of them, but not of the other. Then, by Lemma 2.5 the residuals satisfy either (2.51) or (2.52). This means that the number of iterations cannot differ by more than 1. ∎

2.7 Notation for Arithmetic Complexity

Let C_tn represent the number of floating point operations (flops) for solving a tridiagonal system of dimension n and $C_{t1}n$, the number of floating point operations for solving a tridiagonal system with ones on the off-diagonal. Let $C_s(n)$ be the arithmetic complexity for applying the fast sine transform to a vector of length $n-1$ (asymptotically, $C_S(n) = \mathcal{O}(n \log n)$).

Chapter 3

Preconditioners for the Interface System: Two Subdomain Case

3.1 Introduction

We mentioned in Chapter 1 that the interface system

$$Cu_\Gamma = g \tag{3.1}$$

is solved by iterative methods such as preconditioned conjugate gradients. In this chapter we will describe several preconditioners for the capacitance matrix C. Many have the desirable property of being spectrally equivalent to C i.e., positive constants γ_1 and γ_2 exist such that for all x:

$$\gamma_1 x^T M x \leq x^T C x \leq \gamma_2 x^T M x$$

where γ_1 and γ_2 are independent of the mesh width h. In other words, the spectral condition number of $M^{-1}C$ is bounded independent of h, which means that the rate of convergence will not be affected as the mesh is refined.

Some of the preconditioners have a decomposition in terms of sine Fourier modes of the form:

$$M = W \Lambda W^T \quad , \tag{3.2}$$

where Λ is a diagonal matrix. A linear system of the form $Mu = v$ can be easily solved by fast Fourier transforms (FFT) with $\mathcal{O}(n \log n)$ operations. Chan [12] derives an exact decomposition for C of the form (3.2) for the case of the Poisson equation when the domain Ω is a rectangle divided into two strips. The capacitance matrix can then be solved directly by using FFT's instead of by iterative methods. Since fast direct solvers exist to solve the Poisson equation on a rectangle, one would not find it necessary to apply domain decomposition in this simple case. Nevertheless, this decomposition is important because it can be used as preconditioner for more general operators or domains and because it gives a framework in which other preconditioners can be analyzed.

In what follows, we will describe preconditioners for

$$C = A_{33} - A_{13}^T A_{11}^{-1} A_{13} - A_{23}^T A_{22}^{-1} A_{23} \quad . \tag{3.3}$$

The system (3.1) of dimension n, represents the equations for the gridpoints on the interface Γ, which separates the subdomains Ω_1 and Ω_2 as in Fig. 1.1.

3.2 Dryja's Preconditioner

Dryja [18] proposed

$$M_D = \sqrt{K} \quad ,$$

where K is the one-dimensional discrete Laplacian operator (2.7), and proved that there exist constants β_0 and β_1, independent of the mesh width h, such that, for all $y \in R^n$,

$$\beta_0 y^T M_D y \leq y^T C y \leq \beta_1 y^T M_D y \quad .$$

In other words, $\kappa(M_D^{-1}C)$ — the spectral condition number of the preconditioned system — is bounded independently of the mesh size h. M_D has the factorization:

$$M_D = W \; diag(\lambda_1^D, \lambda_2^D, \cdots, \lambda_n^D) \; W^T \quad ,$$

where

$$\lambda_j^D = \sqrt{\sigma_j} \tag{3.4}$$

and σ_j is given by (2.23).

3.3 Golub and Mayers' Preconditioner

Golub and Mayers' preconditioner [26] is a modification of Dryja's:

$$M_{GM} = \sqrt{K + \frac{K^2}{4}} \quad , \tag{3.5}$$

which also has a decomposition of the form (3.2) with eigenvalues given by

$$\lambda_j^{GM} = \sqrt{\sigma_j + \frac{\sigma_j^2}{4}} \quad .$$

For the case of the Poisson equation on a rectangular domain, M_{GM} approximates the capacitance matrix when the boundaries which are parallel to the interface become infinitely apart and the Schur complement C approaches a Toeplitz matrix. Experimental results show that M_{GM} performs better than M_D.

3.4 Björstad and Widlund's Preconditioner

Another preconditioner was given by Björstad and Widlund [4], based on an idea of Dryja's. It takes one of the following forms:

$$M_{BW}^{(1)} = A_{33}^{(1)} - A_{13}^T A_{11}^{-1} A_{13}$$

or

$$M_{BW}^{(2)} = A_{33}^{(2)} - A_{13}^T A_{11}^{-1} A_{13}$$

where the matrices $A_{33}^{(1)}$ and $A_{33}^{(2)}$ represent the differential operator's contributions to A_{33} from Ω_1 and Ω_2 respectively so that

$$A_{33} = A_{33}^{(1)} + A_{33}^{(2)} \quad .$$

When Γ separates the domain Ω in two identical pieces Ω_1 and Ω_2 and the coefficients of the differential operator are symmetric with respect to Γ, it is easy to see that

$$M_{BW}^{(1)} = M_{BW}^{(2)} = \frac{1}{2} C \quad .$$

In other words, M_{BW} is exact. When L is the Laplacian operator and one of the subdomains — say Ω_i — is a rectangle, it can be shown that $M_{BW}^{(i)}$ also has a decomposition of the form (3.2). The eigenvalues are given by

$$\lambda_j^{BW} = \left(\frac{1+\gamma_j^{m_i+1}}{1-\gamma_j^{m_i+1}}\right)\sqrt{\sigma_j + \frac{\sigma_j^2}{4}}$$

where

$$\gamma_j = \left(1 + \frac{\sigma_j}{2} - \sqrt{\sigma_j + \frac{\sigma_j^2}{4}}\right)^2 \tag{3.6}$$

and m_i is the number of rows of interior grid points in the y-direction in subdomain Ω_i.

To implement the method, Björstad and Widlund solve a mixed Neumann-Dirichlet problem in one of the subdomains and a Dirichlet problem in the other. Their method has the advantage that it can be applied to more general operators, domain shapes and discretizations, but in the particular case of the five-point discretization of the Laplacian operator on a union of rectangles, such procedure is less efficient than applying a single FFT computation on the interface grid points.

3.5 Chan's Preconditioner

Chan [12] gives an exact eigenvalue decomposition of C of the form (3.2) for the case when L is the Laplacian operator and Ω is a rectangle divided into two strips, namely

$$W diag(\lambda_1^C, \lambda_2^C, \cdots, \lambda_n^C) W^T \quad , \tag{3.7}$$

where

$$\lambda_j^C = \left(\frac{1+\gamma_j^{m_1+1}}{1-\gamma_j^{m_1+1}} + \frac{1+\gamma_j^{m_2+1}}{1-\gamma_j^{m_2+1}}\right)\sqrt{\sigma_j + \frac{\sigma_j^2}{4}} \quad .$$

This expression for λ_j can be proved by applying lemma 2.2.

Although M_D, M_{GM} and M_{BW} were derived independently of the factorization (3.7), they can be viewed as progressively better approximations to the capacitance matrix C. For the case of the Poisson equation on a rectangular Ω, the factorization (3.7) is exact, while M_D and M_{GM} are not. It can be easily observed that (3.4) is a first order approximation to (3.8), while (3.6) is a second order approximation. M_{BW}, on the other hand, is exact only for the case of a rectangular domain divided symmetrically into two strips. Anderson [1] gives an interpretation of the various discrete preconditioners as approximations to a continuous operator on the interface.

Chan's analysis in [12] shows that some preconditioners are quite sensitive to the shape of the domains. In particular, their performance depends on the *aspect ratios* of the subdomains Ω_1 and Ω_2, defined by

$$\alpha_1 = \frac{m_1+1}{n+1} \quad \text{and} \quad \alpha_2 = \frac{m_2+1}{n+1} \quad .$$

By using the decomposition (3.7) for the case of the Poisson equation on a rectangular domain divided into two strips, Chan shows that, as the mesh width h tends to zero, the condition number of the preconditioned capacitance matrix with preconditioner M_{GM} is:

$$\lim_{h\to 0} \kappa(M_{GM}^{-1} C) = \frac{1}{2}\left(\frac{1+e^{-2\pi\alpha_1}}{1-e^{-2\pi\alpha_1}} + \frac{1+e^{-2\pi\alpha_2}}{1-e^{-2\pi\alpha_2}}\right) \quad , \tag{3.8}$$

The expression (3.8) suggests that M_{GM} deteriorates as the aspect ratios of the subdomains decrease. He also shows that

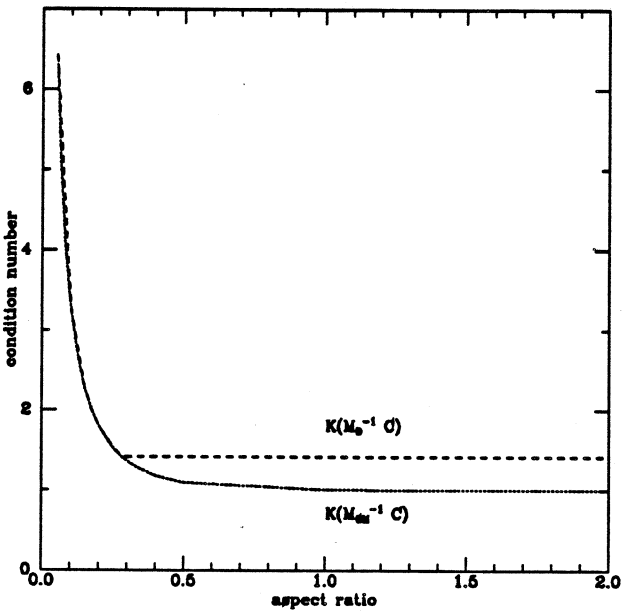

Figure 3.1: Condition number of $M_D^{-1}C$ and $M_{GM}^{-1}C$ vs. aspect ratio.

$$\frac{1}{\sqrt{2}} \leq \frac{\kappa(M_D^{-1}C)}{\kappa(M_{GM}^{-1}C)} \leq \sqrt{2} \tag{3.9}$$

and

$$\lim_{\substack{\alpha_1 \to \infty \\ \alpha_2 \to \infty \\ h \to 0}} \kappa(M_D^{-1}C) = \sqrt{2} \quad . \tag{3.10}$$

By (3.9) we can see that M_D also deteriorates as the aspect ratios of the subdomains decrease. On the other hand, by comparing (3.8) and (3.10), we can see that for large values of α_1 and α_2, M_{GM} becomes exact, while M_D does not.

In Fig. 3.1 we plot the condition numbers $\kappa(M_D^{-1}C)$ and $\kappa(M_{GM}^{-1}C)$ computed for large values of n, as a function of the aspect ratios. The domain Ω is a rectangle divided symmetrically into two strips (i.e. $m_1 = m_2$ and $\alpha_1 = \alpha_2 = \alpha$). For small values of α, both condition numbers practically coincide with the function $\phi(\alpha) = \frac{1+e^{-2\pi\alpha}}{1-e^{-2\pi\alpha}}$. At $\alpha \approx 0.28$, the curve for $\kappa(M_D^{-1}C)$ switches to the constant value $\sqrt{2}$, while $\kappa(M_{GM}^{-1}C)$ continues to approach one as $\phi(\alpha)$. The abrupt change in slope can be explained by observing the eigenvalues of $M_D^{-1}C$: λ_1 approaches $\phi(\alpha)$ for large n, λ_n approaches $\sqrt{2}$ and $\lambda_{min} \approx 1$. When $\alpha \approx 0.28$, $\phi(\alpha) = \sqrt{2}$ and at that point, the expression for $\kappa(M_D^{-1}C)$ switches from $\frac{\lambda_1}{\lambda_{min}}$ to $\frac{\lambda_n}{\lambda_{min}} \approx \sqrt{2}$. Numerical experiments also show that the eigenvalues of $M_{GM}^{-1}C$ are more clustered than those of $M_D^{-1}C$.

For the regular case of Poisson's equation on a rectangle divided into two strips, the decomposition (3.7) is always exact. This suggests that it can become a good preconditioner for more general operators and regions. We will call this preconditioner M_C. A formal analysis of M_C when applied to L-shaped and C-shaped domains is given in Chapter 5.

Chapter 4

Some Exact Decompositions of the Schur Complement

4.1 Introduction

The purpose of this chapter is to describe the Schur complement

$$C = A_\Gamma - P^T A_\Omega^{-1} P \quad ,$$

also called the interface matrix or capacitance matrix, for some regular problems for which fast solvers can be applied to the solution of the interface system

$$C u_\Gamma = g \quad .$$

In Chapter 3, an exact eigenvalue decomposition of C is given for the case of the Poisson equation on a two-strip decomposition of a rectangle. The given representation of C is exact only for this case, but it can be efficiently applied as a preconditioner for the interface matrix on other irregular domains, such as L-shaped or T-shaped regions. It can also be used as preconditioner for the interface matrix when the differential operator can be approximated by the Laplacian operator. However, the differential operator can sometimes be better approximated by other constant coefficient operators. In this chapter we derive eigenvalue decompositions for the interface matrix for more general constant and piecewise-constant coefficient problems.

We also derive expressions for the interface matrix corresponding to a multistrip decomposition of a rectangular domain. This decomposition will have an important application in the implementation of a parallel preconditioner for variable coefficient problems in Chapter 7.

4.2 Two-strip Decompositions

In this section we consider a rectangular domain Ω and describe the interface matrix for the problem

$$\begin{aligned} Lu &= f \quad \text{in} \quad \Omega \\ u &= u_b \quad \text{on} \quad \partial\Omega \end{aligned}$$

for a two-strip decomposition of Ω as shown in Fig. 4.1. As usual, we assume that an n by m regular mesh is imposed over Ω, with n gridpoints located on the interface Γ. Let m_i be the number of interior gridpoints in the y-direction in Ω_i, with $m_1 + m_2 + 1 = m$.

The discrete equation $Au = f$ is written in block form as

$$\begin{pmatrix} A_{11} & 0 & A_{13} \\ 0 & A_{22} & A_{23} \\ A_{13}^T & A_{23}^T & A_{33} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} \quad .$$

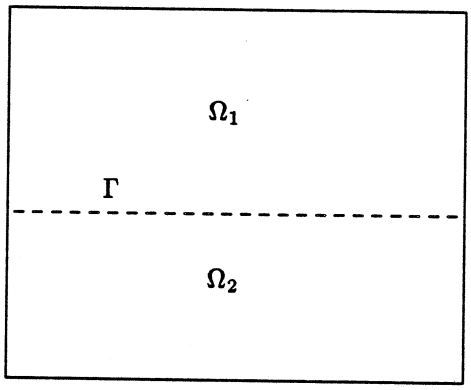

Figure 4.1: Rectangular domain divided into two strips

The case of the Poisson equation, i.e., when $L = -\Delta$, was treated in Chapter 3. The Schur complement of A_{33} in A:

$$C = A_{33} - A_{13}^T A_{11}^{-1} A_{13} - A_{23}^T A_{22}^{-1} A_{23} \quad ,$$

has an eigenvalue decomposition in terms of Fourier modes: $C = W\Lambda W$, with eigenvalues:

$$\lambda_j = \left(\frac{1+\gamma_j^{m_1+1}}{1-\gamma_j^{m_1+1}} + \frac{1+\gamma_j^{m_2+1}}{1-\gamma_j^{m_2+1}} \right) \sqrt{\sigma_j + \frac{\sigma_j^2}{4}} \tag{4.1}$$

where $\gamma_j = \left(1 + \frac{\sigma_j}{2} - \sqrt{\sigma_j + \frac{\sigma_j^2}{4}} \right)^2$.

For more general constant coefficient operators, the interface matrix C can be described in a similar way.

Suppose that the matrix A has the form

$$A = \begin{pmatrix} T_1 & Q_1 & & & & & & \\ Q_1 & T_1 & \ddots & & & & & \\ & \ddots & \ddots & Q_1 & & & & \\ & & Q_1 & T_1 & T_2 & Q_2 & & Q_1 \\ & & & & Q_2 & T_2 & \ddots & Q_2 \\ & & & & & \ddots & \ddots & Q_2 \\ & & & & & & Q_2 & T_2 \\ & & & Q_1^T & Q_2^T & & & T_3 \end{pmatrix} \quad , \tag{4.2}$$

where T_i are diagonalizable by Fourier modes with eigenvalues t_{ij} and $Q_i = -d_i I$. Then the Schur complement C is also diagonalizable by Fourier modes.

Lemma 4.1 *If A is given by (4.2), the eigenvalues of the Schur complement C are given by*

$$\lambda_j = t_{3j} - d_1 \sqrt{\gamma_{1j}} \frac{1+\gamma_{1j}^{m_1}}{1-\gamma_{1j}^{m_1+1}} - d_2 \sqrt{\gamma_{2j}} \frac{1+\gamma_{2j}^{m_2}}{1-\gamma_{2j}^{m_2+1}} \quad , \tag{4.3}$$

where $\gamma_{ij} = \gamma(\mu_{ij})$ and $\mu_{ij} = -2 + \frac{t_{ij}}{d_i}$ ($\gamma(\mu)$ is defined in page 11). If we further assume that $2T_3 = T_1 + T_2$, then we have

$$\lambda_j = d_1 \sqrt{\frac{\mu_{1j}^2}{4} + \mu_{1j}} \; \frac{1+\gamma_{1j}^{m_1+1}}{1-\gamma_{1j}^{m_1+1}} \; + \; d_2 \sqrt{\frac{\mu_{2j}^2}{4} + \mu_{2j}} \; \frac{1+\gamma_{2j}^{m_2+1}}{1-\gamma_{2j}^{m_2+1}} \quad . \tag{4.4}$$

Proof: The operator $A_{13}^T A_{11}^{-1} A_{13}$ corresponds to Q_{SS} of Lemma 2.2 (page 13) and $A_{23}^T A_{22}^{-1} A_{23}$ corresponds to Q_{NN}. By replacing $\frac{d_S^2}{d} = d_1$ and $\frac{d_N^2}{d} = d_2$, we get (4.3).

If $2T_3 = T_1 + T_2$, then (4.3) becomes

$$\lambda_j = -d_1 \left(-\frac{t_{1j}}{2d_1} + \sqrt{\gamma_{1j}} \frac{1-\gamma_{1j}^{m_1}}{1-\gamma_{1j}^{m_1+1}} \right) - d_2 \left(-\frac{t_{2j}}{2d_2} + \sqrt{\gamma_{2j}} \frac{1-\gamma_{2j}^{m_2}}{1-\gamma_{2j}^{m_2+1}} \right) .$$

By (2.18) and (2.19) we have $\sqrt{\gamma_{ij}} + \frac{1}{\sqrt{\gamma_{ij}}} = \frac{t_{ij}}{d_i}$ and $\sqrt{\gamma_{ij}} - \frac{1}{\sqrt{\gamma_{ij}}} = -2\sqrt{\frac{\mu_{ij}^2}{4} + \mu_{ij}}$. Then we can see that

$$\begin{aligned} -\frac{t_{ij}}{2d_i} + \sqrt{\gamma_{ij}} \frac{1-\gamma_{ij}^{m_i}}{1-\gamma_{ij}^{m_i+1}} &= \frac{1}{2} \left(\sqrt{\gamma_{ij}} - \frac{1}{\sqrt{\gamma_{ij}}} \right) \frac{1+\gamma_{ij}^{m_i+1}}{1-\gamma_{ij}^{m_i+1}} \\ &= -\sqrt{\frac{\mu_{ij}^2}{4} + \mu_{ij}} \frac{1+\gamma_{1j}^{m_1+1}}{1-\gamma_{1j}^{m_1+1}} \end{aligned}$$

and therefore

$$\lambda_j = d_1 \sqrt{\frac{\mu_{1j}^2}{4} + \mu_{1j}} \frac{1+\gamma_{1j}^{m_1+1}}{1-\gamma_{1j}^{m_1+1}} + d_2 \sqrt{\frac{\mu_{2j}^2}{4} + \mu_{2j}} \frac{1+\gamma_{2j}^{m_2+1}}{1-\gamma_{2j}^{m_2+1}}$$

■

In particular, consider the operator:

$$Lu \equiv -au_{xx} - bu_{yy} + cu \quad , \tag{4.5}$$

where the coefficients a, b and c take constant values a_1, b_1 and c_1 on subdomain Ω_1 and a_2, b_2 and c_2 on subdomain Ω_2. A five-point approximation of this problem was described in Chapter 2. The matrix A is given by (2.13). The interface matrix has a decomposition of the form $C = W\Lambda W$, with eigenvalues given by

$$\lambda_j = b_1 \sqrt{\frac{\mu_{1j}^2}{4} + \mu_{1j}} \frac{1+\gamma_{1j}^{m_1+1}}{1-\gamma_{1j}^{m_1+1}} + b_2 \sqrt{\frac{\mu_{2j}^2}{4} + \mu_{2j}} \frac{1+\gamma_{2j}^{m_2+1}}{1-\gamma_{2j}^{m_2+1}} \tag{4.6}$$

for $\gamma_{ij} = \gamma(\mu_{ij})$ and $\mu_{ij} = \frac{1}{b_i}(a_i\sigma_j + c_i h^2)$.

When the coefficients are constant through the whole domain, we have

$$\lambda_j = -b \sqrt{\frac{\mu_j^2}{4} + \mu_j} \left(\frac{1+\gamma_j^{m_1+1}}{1-\gamma_j^{m_1+1}} + \frac{1+\gamma_j^{m_2+1}}{1-\gamma_j^{m_2+1}} \right) \tag{4.7}$$

for $\mu_j = \frac{1}{b}(a\sigma_j + ch^2)$. In particular, when $a = b = 1$ and $c = 0$, L is the Laplacian operator. It can be easily verified that for this particular case, (4.7) reduces to (4.1).

For the case of the Helmholtz operator:

$$u_{xx} + u_{yy} + \alpha u$$

we have

$$\lambda_j = -\sqrt{\frac{\mu_j^2}{4} + \mu_j} \left(\frac{1+\gamma_j^{m_1+1}}{1-\gamma_j^{m_1+1}} + \frac{1+\gamma_j^{m_2+1}}{1-\gamma_j^{m_2+1}} \right) \tag{4.8}$$

for $\mu_j = -\sigma_j + \alpha h^2$.

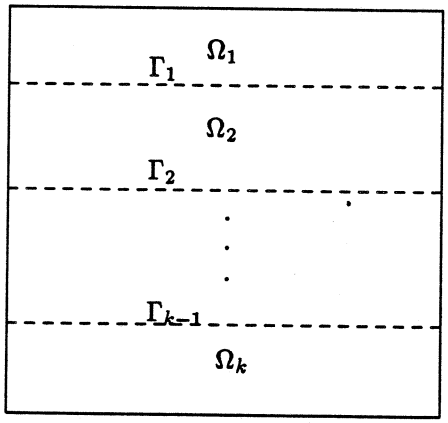

Figure 4.2: Rectangular domain divided into strips

4.3 Multi-Strip Decompositions

When considering parallel implementations of elliptic solvers, domain decomposition is a natural choice, since the subdomain problems can be solved in parallel and the communication is reduced to the boundaries. When the domain is divided into k strips as in Fig. 4.2, the interface matrix for the constant-coefficient case can be described in terms of Fourier modes.

We assume that the size of the original grid on Ω is n by m and after the decomposition into strips, we have n by m_i grids inside each subdomain.

After reordering the unknowns, the discretized equations have the form (2.13), where A_Γ is a block diagonal matrix, because there is no coupling between points on two different interfaces. The submatrix P has the following bidiagonal form:

$$P = \begin{pmatrix} P_{1,1} & & & & \\ P_{2,1} & P_{2,2} & & & \\ & \ddots & \ddots & & \\ & & P_{k-1,k-2} & P_{k-1,k-1} \\ & & & P_{k,k-1} \end{pmatrix},$$

where P_{ij} corresponds to the coupling between the unknowns in the subdomain Ω_i with those on the interface Γ_j. The Schur complement (2.14) has the following block-tridiagonal form:

$$C = \begin{pmatrix} H_1 & B_2 & & \\ B_2 & H_2 & \ddots & \\ & \ddots & \ddots & B_{k-1} \\ & & B_{k-1} & H_{k-1} \end{pmatrix}, \tag{4.9}$$

where

$$H_i = A_{\Gamma_i} - P^T_{i+1,i} A^{-1}_{\Omega_{i+1}} P_{i+1,i} - P^T_{i,i} A^{-1}_{\Omega_i} P_{i,i} \tag{4.10}$$

and

$$B_i = -P^T_{i,i-1} A^{-1}_{\Omega_i} P_{i,i} \quad . \tag{4.11}$$

Similarly to the two-strip case, we can show that for piece-wise constant coefficient problems, the blocks H_i and B_i are diagonalizable by Fourier modes.

Lemma 4.2 *Assume that each block A_{Ω_i} has the block tridiagonal form:*

$$A_{\Omega_i} = \begin{pmatrix} T_i & -d_i I_n & & \\ -d_i I_n & T_i & \ddots & \\ & \ddots & \ddots & -d_i I_n \\ & & -d_i I_n & T_i \end{pmatrix}$$

with m_i blocks, that T_i as well as all diagonal blocks in A_Γ are diagonalizable by Fourier modes, i.e.

$$T_i = W\ diag(t_{i1}, \ldots, t_{in}) W^T$$

and

$$A_{\Gamma_i} = W\ diag(\beta_{i1}, \ldots, \beta_{in})\ W^T \quad ,$$

that $P_{ii} = -d_i\ e^{(i)}_{m_i} \otimes I_n$ and $P_{i,i-1} = -d_i\ e^{(i)}_1 \otimes I_n$, where I_n is the identity matrix of dimension n and $e^{(i)}_j$, the j-th column of I_{m_i}. In other words, P_{ii} corresponds to P_S of (2.26) and $P_{i,i-1}$ corresponds to P_N of (2.25). Then,

$$W^T H_i W = \Lambda_i = \ diag(\lambda_{i1}, \ldots, \lambda_{in}) \tag{4.12}$$

and

$$W^T B_i W = D_i = \ diag(\delta_{i1}, \ldots, \delta_{in}) \quad . \tag{4.13}$$

The eigenvalues λ_{ij} are given by an expression which is equivalent to (4.3) for the case of two strips:

$$\lambda_{ij} = \beta_{ij} - d_i\sqrt{\gamma_{ij}}\frac{1-\gamma_{ij}^{m_i}}{1-\gamma_{ij}^{m_i+1}} - d_{i+1}\sqrt{\gamma_{i+1j}}\frac{1-\gamma_{i+1j}^{m_{i+1}}}{1-\gamma_{i+1j}^{m_{i+1}+1}} \tag{4.14}$$

where $\gamma_{ij} = \gamma(\mu_{ij})$ and $\mu_{ij} = -2 + \frac{t_{ij}}{d_i}$. The eigenvalues of B_i are given by

$$\delta_{ij} = -d_i\sqrt{\gamma_{ij}^{m_i}}\left(\frac{1-\gamma_{ij}}{1-\gamma_{ij}^{m_i+1}}\right) \quad . \tag{4.15}$$

and they can also be written as

$$\delta_{ij} = -2d_i\left(\frac{(\sqrt{\gamma_{ij}})^{m_i+1}}{1-\gamma_{ij}^{m_i+1}}\right)\sqrt{\frac{\mu_{ij}^2}{4} + \mu_{ij}} \quad . \tag{4.16}$$

If we further assume that $2A_{\Gamma_i} = T_i + T_{i+1}$, then we have

$$\lambda_{ij} = d_i\sqrt{\frac{\mu_{ij}^2}{4} + \mu_{ij}}\frac{1+\gamma_{ij}^{m_i+1}}{1-\gamma_{ij}^{m_i+1}} + d_{i+1}\sqrt{\frac{\mu_{i+1,j}^2}{4} + \mu_{i+1,j}}\frac{1+\gamma_{i+1,j}^{m_{i+1}+1}}{1-\gamma_{i+1,j}^{m_{i+1}+1}} \tag{4.17}$$

Proof: Equations (4.14) and (4.17) can be proved analogously to Lemma 4.1. In order to prove (4.16) we apply Lemma 2.2, with $B_i = -Q_{NS}$. Then we have

$$\delta_{ij} = -d_i\sqrt{\gamma_{ij}^{m_i}}\left(\frac{1-\gamma_{ij}}{1-\gamma_{ij}^{m_i+1}}\right) \quad .$$

Moreover, by (2.19), we have

$$\frac{1-\gamma_{ij}}{\sqrt{\gamma_{ij}}} = 2\sqrt{\frac{\mu_{ij}^2}{4} + \mu_{ij}} \quad .$$

Therefore

$$\delta_{ij} = -2d_i\left(\frac{(\sqrt{\gamma_{ij}})^{m_i+1}}{1-\gamma_{ij}^{m_i+1}}\right)\sqrt{\frac{\mu_{ij}^2}{4} + \mu_{ij}} \quad .$$

∎

Using the expressions (4.9), (4.12) and (4.13), the interface system can be solved by *matrix decomposition* [10] by applying fast sine transforms and solving n decoupled tridiagonal systems of dimension $k-1$.

The matrix C is described by Bank and Rose [2] for the particular case of the Poisson equation in terms of Chebyshev polynomials, in the context of generalized marching algorithms. For this particular case, λ_{ij} and δ_{ij} are given by

$$\lambda_{ij} = \left(\frac{1+\gamma_j^{m_i+1}}{1-\gamma_j^{m_i+1}} + \frac{1+\gamma_j^{m_{i+1}+1}}{1-\gamma_j^{m_{i+1}+1}} \right) \sqrt{\sigma_j + \frac{\sigma_j^2}{4}}$$

and

$$\delta_{ij} = -2 \frac{\gamma_j^{\frac{m_i+1}{2}}}{1-\gamma_j^{m_i+1}} \sqrt{\sigma_j + \frac{\sigma_j^2}{4}} \quad .$$

Consider the operator (4.5), where the coefficients a, b and c take constant values a_i, b_i and c_i on each subdomain Ω_i, for $i = 1, \ldots, k$. The matrix A satisfies the hypothesis of Lemma 4.2, with

$$\begin{aligned} t_{ij} &= 2b_i + h^2 c_i + a_i \sigma_j(n) \quad , \\ \beta_{ij} &= b_i + b_{i+1} + \frac{c_i + c_{i+1}}{2} h^2 + \frac{a_i + a_{i+1}}{2} \sigma_j(n) \end{aligned}$$

and $d_i = b_i$. The interface matrix can be described by expressions (4.9), (4.12) and (4.13), with eigenvalues given by

$$\lambda_{ij} = b_i \left(\frac{1+\gamma_{ij}^{m_i+1}}{1-\gamma_{ij}^{m_i+1}} \right) \sqrt{\frac{\mu_{ij}^2}{4} + \mu_{ij}} + b_{i+1} \left(\frac{1+\gamma_{i+1,j}^{m_{i+1}+1}}{1-\gamma_{i+1,j}^{m_{i+1}+1}} \right) \sqrt{\frac{\mu_{i+1j}^2}{4} + \mu_{i+1j}} \tag{4.18}$$

and

$$\delta_{ij} = -2b_i \left(\frac{(\sqrt{\gamma_{ij}})^{m_i+1}}{1-\gamma_{ij}^{m_i+1}} \right) \sqrt{\frac{\mu_{ij}^2}{4} + \mu_{ij}} \tag{4.19}$$

for $\gamma_{ij} = \gamma(\mu_{ij})$ and

$$\mu_{ij} = \frac{a_i \sigma_j + c_i h^2}{b_i} \quad .$$

The operator (4.5) is analyzed in Chapter 7 as a preconditioner for more general variable coefficient problems.

4.4 A Special Case of Irregular Mesh

In all the cases mentioned above, we assumed that a regular mesh of size h was imposed over the whole domain Ω. Similar decompositions of the interface matrix can be also found when the mesh width in the horizontal direction is different from the mesh width in the vertical direction. Moreover, consider the region Ω of Fig. 4.2 decomposed into k strips Ω_i and suppose that an n by m_i grid is imposed on each strip, where the mesh widths $h_i = 1/(m_i+1)$ are not necessarily the same. Such discretization is useful, for example, when the solution to the differential equation is known to behave in a wavefront fashion and a fine grid is needed only in a narrow subregion. Let h be the mesh width in the horizontal direction, $h = 1/(n+1)$ and define $\zeta_i = h_i/h$.

The matrix A for the operator (4.5) with the discretization just described satisfies the hypothesis of Lemma 4.2, with

$$\begin{aligned} t_{ij} &= \zeta_i a_i \sigma_j(n) + 2\frac{b_i}{\zeta_i} + h^2 \zeta_i c_i \quad , \\ \beta_{ij} &= \frac{(\zeta_i + \zeta_{i+1})}{2} \frac{(a_i + a_{i+1})}{2} \sigma_j(n) + \frac{b_i}{\zeta_i} + \frac{b_{i+1}}{\zeta_{i+1}} + h^2 \frac{(\zeta_i + \zeta_{i+1})}{2} \frac{(c_i + c_{i+1})}{2} \end{aligned}$$

and $d_i = \frac{b_i}{\zeta_i}$. The interface matrix can then be described by expressions (4.9), (4.12) and (4.13), with eigenvalues given by (4.14) and (4.15), for $\gamma_{ij} = \gamma(\mu_{ij})$ and

$$\mu_{ij} = \zeta_i^2 \frac{a_i\sigma_j + c_i h^2}{b_i} \quad .$$

For general piece-wise constant coefficient cases, $2A_{\Gamma_i} \neq T_i + T_{i+1}$, but we can prove that

$$2\beta_{ij} = t_{ij} + t_{i+1,j} - \frac{(\zeta_{i+1} - \zeta_i)}{2}\Big((a_{i+1} - a_i)\sigma_j(n) + h^2(c_{i+1} - c_i)\Big) \quad ,$$

then, we have

$$\begin{aligned}\lambda_{ij} \;=\; & \frac{b_i}{\zeta_i}\sqrt{\frac{\mu_{ij}^2}{4} + \mu_{ij}}\;\frac{1+\gamma_{ij}^{m_i+1}}{1-\gamma_{ij}^{m_i+1}} \;+\; \frac{b_{i+1}}{\zeta_{i+1}}\sqrt{\frac{\mu_{i+1,j}^2}{4} + \mu_{i+1,j}}\;\frac{1+\gamma_{i+1,j}^{m_{i+1}+1}}{1-\gamma_{i+1,j}^{m_{i+1}+1}} \\ & -\frac{1}{4}(\zeta_{i+1} - \zeta_i)\Big((a_{i+1} - a_i)\sigma_j(n) + h^2(c_{i+1} - c_i)\Big) \quad . \end{aligned} \tag{4.20}$$

In particular, if $a_i \equiv a$ and $c_i \equiv c$, we have

$$\lambda_{ij} = \frac{b_i}{\zeta_i}\sqrt{\frac{\mu_{ij}^2}{4} + \mu_{ij}}\frac{1+\gamma_{ij}^{m_i+1}}{1-\gamma_{ij}^{m_i+1}} + \frac{b_{i+1}}{\zeta_{i+1}}\sqrt{\frac{\mu_{i+1,j}^2}{4} + \mu_{i+1,j}}\frac{1+\gamma_{i+1,j}^{m_{i+1}+1}}{1-\gamma_{i+1,j}^{m_{i+1}+1}} \quad .$$

Also, we can easily verify that when $h_i \equiv h$, (4.20) becomes (4.18)

4.5 Summary

In this chapter, we derived decompositions of the interface matrix in terms of Fourier modes for several operators on rectangular domains subdivided into two or more strips. For two-strip decompositions, the interface matrix has an eigenvalue decomposition in terms of Fourier modes, of the form:

$$C = W\Lambda W \quad .$$

Table 4.1 shows the eigenvalues of C for two-strip decompositions, for the various operators described in this chapter.

When a rectangular domain is decomposed into k strips as in Fig. 4.2, the interface matrix takes the form

$$C = \begin{pmatrix} H_1 & B_2 & & \\ B_2 & H_2 & \ddots & \\ & \ddots & \ddots & B_{k-1} \\ & & B_{k-1} & H_{k-1} \end{pmatrix} \quad ,$$

where $W^T H_i W = \Lambda_i$ and $W^T B_i W = D_i$. The elements of the diagonal matrices Λ_i and D_i for the various operators given in this chapter are given in Table 4.2. All operators in the table represent particular cases of the general operator of Lemma 4.2.

Table 4.1: Two-Strip Decompositions

The Schur complement has the form $C = W^T \text{ diag } (\lambda_j) W$.

Operator	λ_j
Equation (4.2)	$t_{3j} - d_1\sqrt{\gamma(\mu_{1j})}\dfrac{1+\gamma(\mu_{1j})^{m_1}}{1-\gamma(\mu_{1j})^{m_1+1}} - d_2\sqrt{\gamma(\mu_{2j})}\dfrac{1+\gamma(\mu_{2j})^{m_2}}{1-\gamma(\mu_{2j})^{m_2+1}}$ $\gamma(\mu) = \left(1 + \frac{\mu}{2} - \sqrt{\frac{\mu^2}{4}+\mu}\right)^2 \ , \ \mu_{ij} = -2 + t_{ij}/d_i$
Piecewise-constant coefficients	$b_1\sqrt{\dfrac{\mu_{1j}^2}{4}+\mu_{1j}}\dfrac{1+\gamma(\mu_{1j})^{m_1+1}}{1-\gamma(\mu_{1j})^{m_1+1}} + b_2\sqrt{\dfrac{\mu_{2j}^2}{4}+\mu_{2j}}\dfrac{1+\gamma(\mu_{2j})^{m_2+1}}{1-\gamma(\mu_{2j})^{m_2+1}}$ $\mu_{ij} = \dfrac{1}{b_i}(a_i\sigma_j + c_i h^2)$
Laplacian	$\left(\dfrac{1+\gamma(\sigma_j)^{m_1+1}}{1-\gamma(\sigma_j)^{m_1+1}} + \dfrac{1+\gamma(\sigma_j)^{m_2+1}}{1-\gamma(\sigma_j)^{m_2+1}}\right)\sqrt{\sigma_j + \dfrac{\sigma_j^2}{4}}$ $\sigma_j = 4\sin^2\frac{j\pi}{2(n+1)}$
Helmholtz	$-\left(\dfrac{1+\gamma_j^{m_1+1}}{1-\gamma_j^{m_1+1}} + \dfrac{1+\gamma_j^{m_2+1}}{1-\gamma_j^{m_2+1}}\right)\sqrt{\dfrac{\mu_j^2}{4}+\mu_j}$ $\mu_j = -\sigma_j + \alpha h^2$

Table 4.2: Multi-Strip Decompositions

Operator	*Eigenvalues of H_i and B_i*
Lemma 4.2	λ_{ij} (4.14) δ_{ij} (4.15) or (4.16)
Piecewise-constant coefficients	$\lambda_{ij} = \sum_{s=i,i+1} b_s \left(\frac{1+\gamma(\mu_{sj})^{m_s+1}}{1-\gamma(\mu_{sj})^{m_s+1}}\right) \sqrt{\frac{\mu_{sj}^2}{4} + \mu_{sj}}$ $\delta_{ij} = -2b_i \left(\frac{(\sqrt{\gamma(\mu_{ij})})^{m_i+1}}{1-\gamma(\mu_{ij})^{m_i+1}}\right) \sqrt{\frac{\mu_{ij}^2}{4} + \mu_{ij}}$ $\mu_{ij} = \frac{1}{b_i}(a_i\sigma_j + c_i h^2)$
Poisson	$\lambda_{ij} = \left(\frac{1+\gamma_j^{m_i+1}}{1-\gamma_j^{m_i+1}} + \frac{1+\gamma_j^{m_{i+1}+1}}{1-\gamma_j^{m_{i+1}+1}}\right) \sqrt{\sigma_j + \frac{\sigma_j^2}{4}}$ $\delta_{ij} = -2\frac{\gamma_j^{\frac{m_i+1}{2}}}{1-\gamma_j^{m_i+1}} \sqrt{\sigma_j + \frac{\sigma_j^2}{4}}$
Helmholtz	$\lambda_{ij} = -\left(\frac{1+\gamma(\mu_{ij})^{m_i+1}}{1-\gamma(\mu_{ij})^{m_i+1}}\right) \sqrt{\frac{\mu_j^2}{4} + \mu_j}$ $\delta_{ij} = 2\left(\frac{(\sqrt{\gamma(\mu_j)})^{m_i+1}}{1-\gamma(\mu_j)^{m_i+1}}\right) \sqrt{\frac{\mu_j^2}{4} + \mu_j}$ $\mu_j = \sigma_j + \alpha h^2$
Piece-wise constant coefficients on irregular mesh	$\lambda_{ij} = \sum_{s=i,i+1} \frac{b_s}{\zeta_s} \frac{1+\gamma(\mu_{sj})^{m_s+1}}{1-\gamma(\mu_{sj})^{m_s+1}} \sqrt{\frac{\mu_{sj}^2}{4} + \mu_{sj}}$ $-\frac{1}{4}(\zeta_{i+1} - \zeta_i)\left((a_{i+1} - a_i)\sigma_j(n) + h^2(c_{i+1} - c_i)\right)$ $\delta_{ij} = -2\frac{b_i}{\zeta_i} \left(\frac{\sqrt{\gamma(\mu_{ij})}^{m_i+1}}{1-\gamma(\mu_{ij})^{m_i+1}}\right) \sqrt{\frac{\mu_{ij}^2}{4} + \mu_{ij}}$ $\mu_{ij} = \frac{\zeta_i^2}{b_i}(a_i\sigma_j + c_i h^2)$

Chapter 5

Preconditioning the Interface System on Irregular Domains: L-shaped and C-shaped Domains

5.1 Introduction

In this chapter we analyze the preconditioner M_C for the interface matrix on irregular domains. In particular, we want to study the dependence of the convergence rate on the meshsize and the shape of the domain. Many preconditioners, when applied to an L-shaped region, have convergence rates that are bounded independently of the meshsize. The bound, however, depends on the relative aspect ratios of the subdomains.

Consider for example, a five-point discretization of the Poisson equation on the region Ω of Fig. 5.1. In Fig. 5.2 we show how the condition number of $\hat{C} = M^{-1/2}CM^{-1/2}$ grows as h approaches zero, for each of the preconditioners described in Chapter 3, when the aspect ratios are fixed. For this example we chose $\alpha_1 = \alpha_2 = \alpha_3 = 1$. As we can see, the condition number of C (no preconditioner) grows linearly with $\frac{1}{h}$. For all the other preconditioners, including M_C, the condition number of $\hat{C}$ is bounded independently of the meshsize.

In Fig. 5.3 we show the condition number as a function of α_1, the aspect ratio of subdomain Ω_1, fixing $\alpha_2 = 1$ and $\alpha_3 = 1$. In Fig. 5.4 we show the dependency of the condition number on α_2 when $\alpha_1 = \alpha_3 = 1$. For these examples we used a mesh width $h = 0.02$. We can see that for all preconditioners except for M_C, the condition number deteriorates as the aspect ratios decrease. Moreover, in Fig. 5.5 we can see, for example, how the distribution of the eigenvalues of $\hat{C}$ changes as α_2 decreases. We show the case where $\alpha_1 = \alpha_3 = 1$ for $\alpha_2 = 1$, $\alpha_2 = 0.5$ and $\alpha_2 = 0.05$. The eigenvalues of $M_C^{-1}C$ remain clustered around one for all three values of α_2, while for the other preconditioners, the eigenvalues begin to spread appart as α_2 decreases.

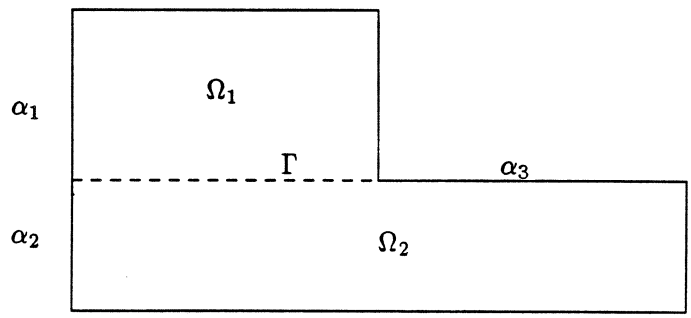

Figure 5.1: L-shaped domain

In this chapter we prove that, in fact, when the M_C preconditioner is used, we have $\kappa(\hat{C}) \leq 2.16$ for all L-shaped regions, independently of meshsize *and aspect ratios* [1].

Another interesting property of the preconditioner M_C on L-shaped domains is also proved in this chapter: there are two ways of decomposing an L-shaped region into two rectangular subregions, but the convergence rate is not affected by the way we choose to subdivide the domain.

Although these results only apply to L-shaped regions, the same ideas can be applied to obtain similar results for other regions which are unions of rectangles. We derive some results for C-shaped regions.

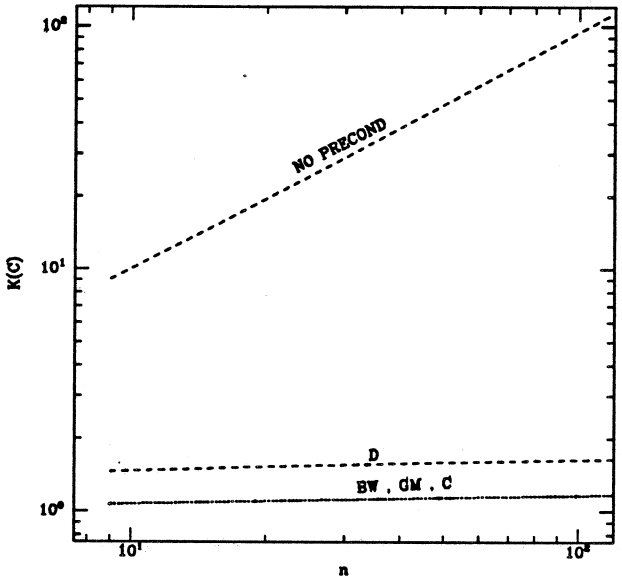

Figure 5.2: Condition number vs. $\frac{1}{h}$

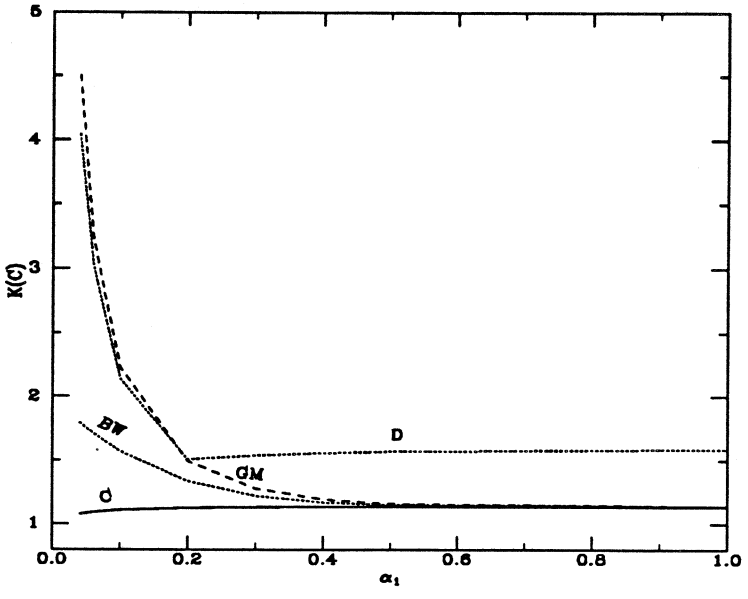

Figure 5.3: Condition number vs. aspect ratio of Ω_1

[1] This bound has been recently improved. Chan, Hou and Lions [14] proved that in fact, $\kappa \leq 2$.

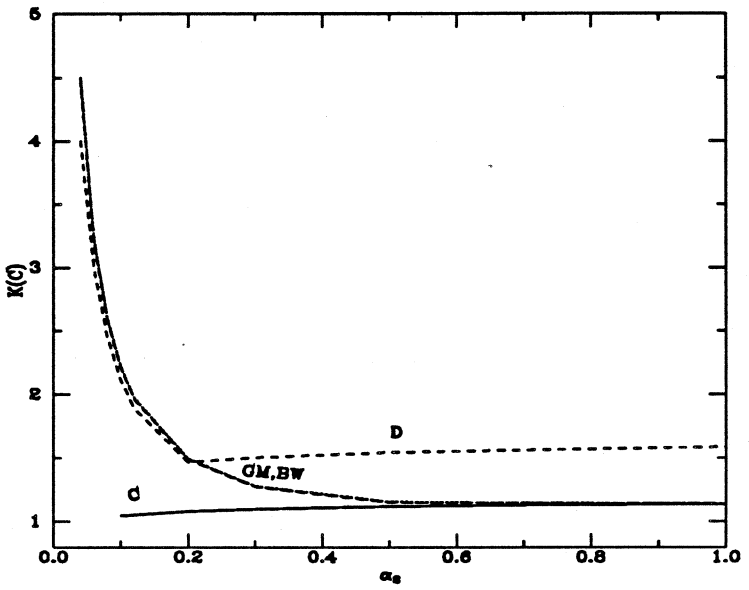

Figure 5.4: Condition number vs. aspect ratio of Ω_2

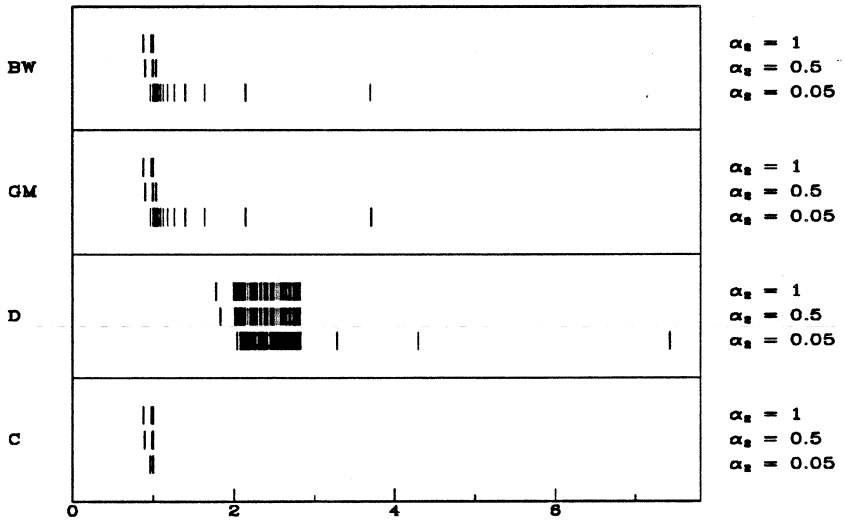

Figure 5.5: Eigenvalue clustering vs. aspect ratio of Ω_2

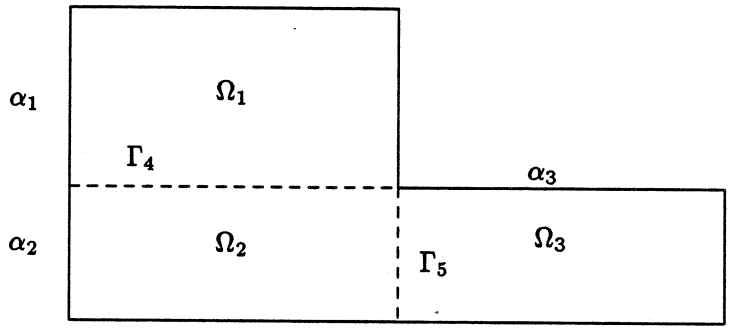

Figure 5.6: Decomposition of an L-shaped domain

5.2 L-shaped Regions

In this section, we describe the interface operator and its preconditioners for an L-shaped domain, the simplest irregular shape that can be decomposed in rectangular subregions. Consider the Poisson equation on the region Ω of Fig. 5.6. It is clear that either interface, Γ_4 or Γ_5, will divide the domain into two rectangles. We might ask ourselves two questions:

- Is one decomposition better than the other?
- How does the convergence rate depend on the mesh size and the aspect ratios of the subdomains?

We will show that, when preconditioner M_C is used, the rate of convergence is the same for the two decompositions. We also give a bound for the condition number that holds for all meshsizes and all L-shaped regions.

Let the linear system

$$Au = f \tag{5.1}$$

represent a standard five-point discretization of the differential equation on a regular grid of size $h = 1/(n+1)$ imposed on the domain Ω. Assume that the size of the grid is n by m_1 in Ω_1, n by m_2 in Ω_2 and n_3 by m_2 in Ω_3. If we consider the domain Ω as the union of two rectangles divided by Γ_4, then an interface system of the form

$$C_4 u_4 = g_4 \tag{5.2}$$

can be derived for the variables on Γ_4 by the process of block elimination. Similarly, if we consider the domain Ω as the union of two rectangles divided by Γ_5, then an interface system of the form

$$C_5 u_5 = g_5 \tag{5.3}$$

can be derived for the variables on Γ_5.

On the other hand, by reordering the gridpoints on Ω_1, Ω_2 and Ω_3 first and then those on the interfaces Γ_4 and Γ_5, (5.1) can be written in block form as:

$$\begin{pmatrix} A_\Omega & P \\ P^T & A_\Gamma \end{pmatrix} \begin{pmatrix} u_\Omega \\ u_\Gamma \end{pmatrix} = \begin{pmatrix} f_\Omega \\ f_\Gamma \end{pmatrix} \quad , \tag{5.4}$$

where

$$A_\Omega = \begin{pmatrix} A_{11} & & \\ & A_{22} & \\ & & A_{33} \end{pmatrix} \quad , \quad A_\Gamma = \begin{pmatrix} A_{44} & \\ & A_{55} \end{pmatrix}$$

and

$$P = \begin{pmatrix} A_{14} & 0 \\ A_{24} & A_{25} \\ 0 & A_{35} \end{pmatrix} \quad .$$

The matrix A of (5.4) can be decomposed as follows:

$$A = \begin{pmatrix} A_\Omega & \\ P^T & C_{45} \end{pmatrix} \begin{pmatrix} I & A_\Omega^{-1} P \\ & I \end{pmatrix} \quad ,$$

where C_{45} is the Schur complement of A_Γ in A, i.e.,

$$C_{45} \equiv A_\Gamma - P^T A_\Omega^{-1} P = \begin{pmatrix} M_4 & -A_{24}^T A_{22}^{-1} A_{25} \\ -A_{25}^T A_{22}^{-1} A_{24} & M_5 \end{pmatrix} \quad , \tag{5.5}$$

with

$$M_4 = A_{44} - A_{14}^T A_{11}^{-1} A_{14} - A_{24}^T A_{22}^{-1} A_{24}$$

and

$$M_5 = A_{55} - A_{25}^T A_{22}^{-1} A_{25} - A_{35}^T A_{33}^{-1} A_{35}.$$

The matrix M_4 would be the capacitance matrix for Γ_4 if the domain Ω_3 were absent. Similarly, M_5 would be the capacitance matrix for Γ_5 if the domain Ω_1 were absent. They are, in fact, nothing but the preconditioner M_C described in Chapter 3.

Let

$$\gamma(\mu) = \left(1 + \frac{\mu}{2} - \sqrt{\frac{\mu^2}{4} + \mu}\right)^2$$

and given integers n, m and p, define

$$\lambda_j(n,m,p) = \left(\frac{1+\gamma(\sigma_j(n))^{m+1}}{1-\gamma(\sigma_j(n))^{m+1}} + \frac{1+\gamma(\sigma_j(n))^{p+1}}{1-\gamma(\sigma_j(n))^{p+1}}\right) \sqrt{\sigma_j(n) + \frac{\sigma_j(n)^2}{4}}$$

for $1 \leq j \leq n$, where $\sigma_j(n) = 4 \sin^2 \frac{j\pi}{2(n+1)}$.

Both M_4 and M_5 have eigenvalue decompositions in terms of Fourier modes, namely

$$\begin{aligned} M_4 &= W_n \Lambda_4 W_n \\ M_5 &= W_{m_2} \Lambda_5 W_{m_2} \quad , \end{aligned}$$

with eigenvalues given by $\lambda_j(n, m_1, m_2)$ and $\lambda_j(m_2, n, n_3)$ respectively.

The matrix C_4 of (5.2) is the Schur complement of A_{44} in A, but it can also be written as the Schur complement of M_4 in C_{45} (see (5.5)). Similarly, C_5 is the Schur complement of A_{55} in A, but it can also be written as the Schur complement of M_5 in C_{45}. We can therefore derive the following expressions for C_4 and C_5:

Lemma 5.1 *The interface matrix for Γ_4 in Ω can be written as*

$$C_4 = M_4 - B^T M_5^{-1} B \tag{5.6}$$

where $B = A_{25}^T A_{22}^{-1} A_{24}$ and analogously, the interface matrix for Γ_5 in Ω can be written as

$$C_5 = M_5 - B M_4^{-1} B^T \quad . \tag{5.7}$$

∎

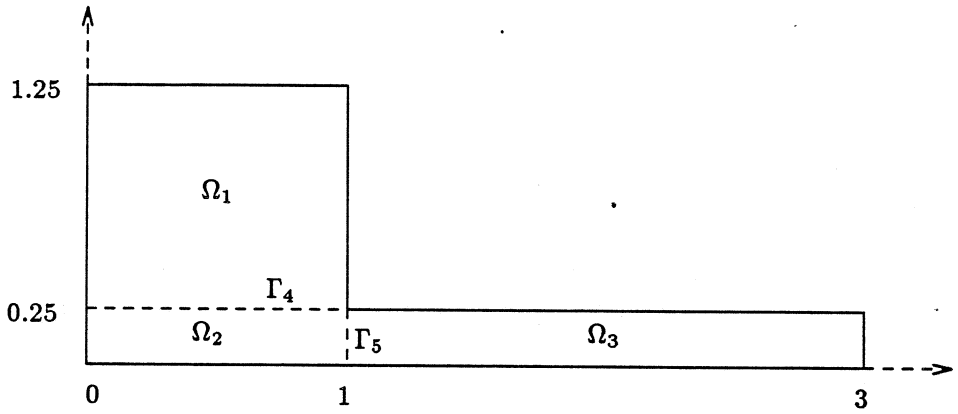

Figure 5.7: Example: an L shaped domain

If we choose Γ_4 as the interface, subdomain problems will be solved on Ω_1 and $\Omega_2 \cup \Gamma_5 \cup \Omega_3$ at each iteration. Similarly, if we choose Γ_5 as the interface, subdomain problems will be solved on $\Omega_1 \cup \Gamma_4 \cup \Omega_2$ and Ω_3 at each iteration. The work per iteration is therefore comparable. We will next show that, by solving (5.2) with preconditioner M_4 and (5.3) with preconditioner M_5, both systems are also equivalent from the convergence point of view. Therefore, there is no *a priori* reason to prefer one way of decomposing the domain over the other.

Since M_C is positive definite, we can choose $\sqrt{M_C}$ as a symmetric preconditioner and define the preconditioned matrices

$$\hat{C}_4 = M_4^{-1/2} C_4 M_4^{-1/2} \quad \text{and} \quad \hat{C}_5 = M_5^{-1/2} C_5 M_5^{-1/2} \quad .$$

By (5.6), we have

$$\hat{C}_4 = I_n - \hat{B}^T \hat{B} \quad \text{and} \quad \hat{C}_5 = I_{m_2} - \hat{B} \hat{B}^T \quad , \tag{5.8}$$

where $\hat{B} = M_5^{-1/2} A_{25}^T A_{22}^{-1} A_{24} M_4^{-1/2}$.

If $n = m_2$, then both interface systems have the same order and it is easy to see that $\hat{C}_4 = \hat{C}_5$. It is therefore not surprising that both systems have equivalent convergence properties. It is not obvious, however, whether one way of decomposing the domain should be prefered when $n \neq m_2$. As it turns out, even in this case the asymptotic convergence rate is the same for both systems, because $\hat{C}_4$ and $\hat{C}_5$ satisfy the hypotheses of Theorem 2.2.

When a preconditioned variational iterative method such as preconditioned conjugate gradients (PCG) is used, the rate of convergence depends on the distribution of the eigenvalues of the corresponding preconditioned matrix. In particular, the bound (2.2) on the convergence rate depends on the condition number of $\hat{C}$. By applying Theorem 2.2 to C_{45} and by using (5.8), we can conclude the following:

Theorem 5.1 *Solving systems (5.2) and (5.3) with preconditioners of the form M_C produce equivalent asymptotic convergence rates. Moreover,*

$$\kappa(\hat{C}_4) \leq \frac{1}{1 - \|\hat{B}\|_2^2} \quad \textit{and} \quad \kappa(\hat{C}_5) \leq \frac{1}{1 - \|\hat{B}\|_2^2} \quad . \tag{5.9}$$

∎

By (5.8), the eigenvalues of $\hat{C}_4$ and $\hat{C}_5$ are given by $1 - \beta_i^2$, where β_i are the singular values of $\hat{B}$. The value 1 is also an eigenvalue of the matrix with the higher dimension. Numerical computations show that the β_i's decrease very quickly with the index i. Therefore, in practice, only a few eigenvalues of $\hat{C}_4$ and

Table 5.1: Singular Values and Eigenvalues of the Preconditioned Interface System for an L-shaped region.

$h = 2^{-5}$		$h = 2^{-6}$	
β_j	λ_j	β_j	λ_j
0.18204	0.96686	2.165E-01	0.95312
0.03868	0.99850	6.816E-02	0.99535
0.00514	0.99997	1.578E-02	0.99975
0.00045	0.99999	2.971E-03	0.99999
0.00002	1.00000	4.607E-04	0.99999
0.00000	1.00000	5.863E-05	1.00000
0.00000	1.00000	6.082E-06	1.00000
		5.093E-07	1.00000
		.	.
		.	.
		.	.

$\hat{C}_5$ are different from 1, which leads to rapid convergence of the iterative method when applied to either matrix. For example, for the L-shaped region of Fig. 5.7 we computed the singular values of $\hat{B}$ and the eigenvalues of $\hat{C}_5$ (and $\hat{C}_4$) in single precision. In the first column of Table 5.1 we show the values of β_j and $\lambda_j^{(5)}$ when $h = 2^{-5}$ and $m_2 = 7$. In the second column, $h = 2^{-6}$ and $m_2 = 15$. Since all but a few of the singular values of $\hat{B}$ are very small, only a few eigenvalues of the preconditioned system are different from one (to working precision). Therefore, the algorithm PCG will converge rapidly.

It is interesting to note the connection between this method and the Schwarz alternating procedure with overlapping region Ω_2. Chan and Goovaerts [13] proved that the Schwarz procedure corresponds to applying the block Gauss-Seidel iteration to the reduced system obtained by eliminating u_Ω in (5.4). If the PCG method is then applied in order to accelerate the convergence of the Schwarz method (two Schwarz steps, in order to make the preconditioner symmetric), the computed values on Γ at each iteration will be the same values computed by our non-overlapping domain decomposition method (see also [5] for further discussion).

5.3 Bound on the Condition Number for L-shaped Regions

Our conclusion from Theorem 5.1 is that, when preconditioner M_C is used, either way of decomposing an L-shaped region into two rectangles produces the same convergence rate. Moreover, we will be able to give an analytical bound on the condition number of the preconditioned capacitance matrix. This bound is derived from a bound on the norm of the operator $\hat{B}^T\hat{B}$.

But first, the following lemma will give us a useful expression for the elements of a unitary transformation of $\hat{B}$.

Lemma 5.2 *Let $V = W_{m_2}\hat{B}W_n$. Then, $\|V\|_2 = \|\hat{B}\|_2$. The elements of the matrix V are given by*

$$v_{ij} = \frac{2}{\sqrt{(n+1)(m_2+1)}} \frac{\sin\frac{i\pi}{m_2+1} \quad \sin\frac{jn\pi}{n+1}}{s_j^{(4)} s_i^{(5)} \left(\sigma_j^{(n)} + \sigma_i^{(m_2)}\right)} \tag{5.10}$$

with $s_j^{(4)} = \sqrt{|\lambda_j(n, m_1, m_2)|}$ *and* $s_i^{(5)} = \sqrt{|\lambda_i(m_2, n, n_3)|}$.

Proof: Since $A_{25}^T A_{22}^{-1} A_{24}$ corresponds to the operator Q_{NE} of Lemma 2.3, we have

$$V = \Lambda_4^{-1/2}(W_{m_2} Q_{NE} W_n)\Lambda_4^{-1/2} \quad .$$

Then,

$$v_{ij} = \frac{2}{s_i^{(4)} s_j^{(5)}} \frac{1}{\sqrt{(n+1)(m_2+1)}} \frac{\sin\frac{ni\pi}{n+1} \quad \sin\frac{j\pi}{m_2+1}}{(\sigma_i(n) + \sigma_j(m_2))} \quad .$$

■

By (5.9) we can see that by finding a bound for $\|\hat{B}\|$ (or equivalently for $\|V\|$), we can bound the condition number of the preconditioned interface system. Since we have an expression for the elements of V, we can bound $\|V\|_1$ and $\|V\|_\infty$ and then use the property:

$$\|V\|_2 \leq \sqrt{\|V\|_1 \|V\|_\infty} \quad .$$

The results are summarized in the next theorem.

Theorem 5.2 *Define the aspect ratio for the domain* Ω_2 *in Fig. 5.6 as* $\alpha = \frac{m_2+1}{n+1}$. *Then*

a) $\|V\|_1 \leq \sqrt{\alpha}\ 0.733$ *and* $\|V\|_\infty \leq \frac{1}{\sqrt{\alpha}}\ 0.733$.

b) $\|\hat{B}^T \hat{B}\|_2 \leq \|\hat{B}\|_2^2 = \|V\|_2^2 \leq \|V\|_1 \|V\|_\infty \leq 0.537$.

c) *For all meshsizes and all L-shaped regions,*

$$\kappa(\hat{C}_4) \leq 2.16 \quad \textit{and} \quad \kappa(\hat{C}_5) \leq 2.16 \quad .$$

Proof: (b) follows from (a). (c) follows from (b) and from (5.9). In order to prove (a), we prove bounds for the column and row sums of the absolute values of (5.10), by applying two lemmas that are proved at the end of this section.

The eigenvalues of M_4 and M_5 can be bounded by

$$\lambda_j(n, m_1, m_2) \geq 4 \sin\frac{j\pi}{2(n+1)}$$

and

$$\lambda_i(m_2, n, n_3) \geq 4 \sin\frac{i\pi}{2(m_2+1)}$$

respectively. It is easy to show that

$$|v_{ij}| \leq \frac{1}{2\sqrt{(n+1)(m_2+1)}} f(x_i, y_j) = \frac{1}{2\sqrt{\alpha}} \frac{f(x_i, y_j)}{n+1} = \frac{\sqrt{\alpha}}{2} \frac{f(x_i, y_j)}{m_2+1} \quad , \tag{5.11}$$

where $x_i = \frac{i}{m_2+1}$ and $y_j = \frac{j}{n+1}$ and the function f is defined by

$$f(x, y) = \frac{\sqrt{\sin x\frac{\pi}{2}}\ \cos x\frac{\pi}{2}\ \sqrt{\sin y\frac{\pi}{2}}\ \cos y\frac{\pi}{2}}{\sin^2 x\frac{\pi}{2} + \sin^2 y\frac{\pi}{2}} \quad . \tag{5.12}$$

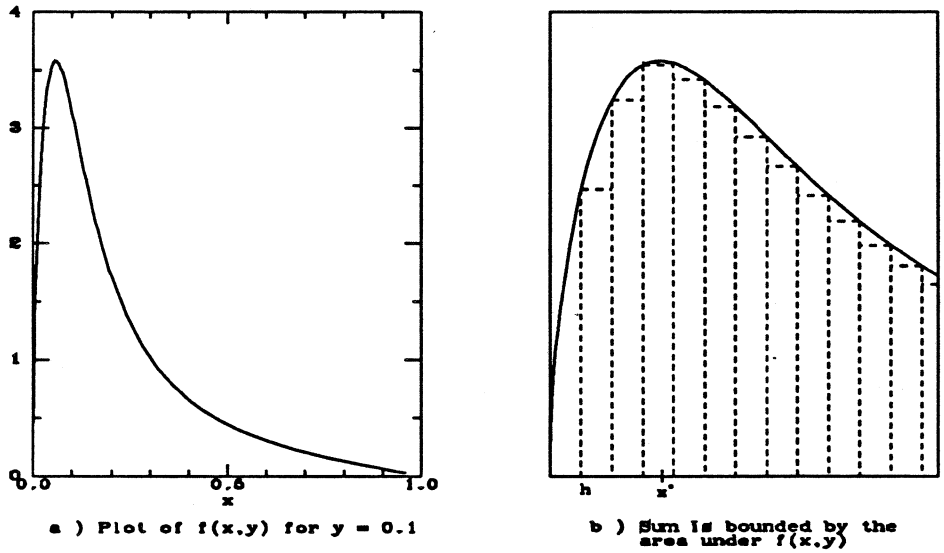

Figure 5.8:

The column sums and the row sums of V can then be bounded by expressions that involve the integrals of f with respect to x or with respect to y. In Lemma 5.3 (appendix at the end of this section), we prove the following expression for the integral of f with respect to x, for a fixed y:

$$\int_a^b f(x,y)\,dx = \frac{2}{\sqrt{2\pi}} \cos y\frac{\pi}{2} \left(\pi - g(\sin b\frac{\pi}{2}, \sin y\frac{\pi}{2}) + g(\sin a\frac{\pi}{2}, \sin y\frac{\pi}{2})\right) \tag{5.13}$$

where

$$g(z,w) = \frac{1}{2}\log\frac{z+\sqrt{2zw}+w}{z-\sqrt{2zw}+w} + \arctan\frac{\sqrt{2zw}}{z-w} \quad .$$

Since $f(x,y) = f(y,x)$, an analogous result holds for the integral of f with respect to y, for a fixed x.

We also need to describe the behavior of $f(x,y)$ for a fixed y, in the interval $x \in (0,1)$. We can easily see that $f(x,y) > 0$ for all $x, y \in (0,1)$. In Lemma 5.4 we prove that, given $y \in (0,1)$, there exists a unique $x^*(y) \in (0,1)$ such that $\max_{0<x<1} f(x,y) = f(x^*,y)$, that $f(.,y)$ is monotonically increasing on the interval $(0,x^*)$, and monotonically decreasing on $(x^*,1)$. Moreover, we prove that $f(x^*,y)$ is bounded by

$$f(x^*,y) \leq \frac{3}{4\sqrt[4]{3}} \cot y\frac{\pi}{2} \quad . \tag{5.14}$$

We can now prove (a) in theorem 5.2. We will only prove the inequality $\|V\|_1 \leq \sqrt{\alpha}\ 0.733$. The proof of $\|V\|_\infty \leq \sqrt{\frac{1}{\alpha}}\ 0.733$ is completely analogous.

By (5.11), we have

$$\|V\|_1 \equiv \max_{1\leq j\leq n} \sum_{i=1}^{m_2} |v_{ij}| \leq \frac{\sqrt{\alpha}}{2}\frac{1}{m_2+1} \max_{1\leq j\leq n} \sum_{i=1}^{m_2} f(x_i,y_j) \quad . \tag{5.15}$$

Let $h = \frac{1}{m_2+1} = x_1$. We know that $f(x,y) > 0$ for $x, y \in (0,1)$ and by Lemma 5.4 f is monotonic in the intervals $(0, x^*(y_j))$ and $(x^*(y_j), 1)$. In Fig. 5.8a we show a plot of $f(x,y)$ for $y = 0.1$. The $\sum hf(x_i,y_j)$ can be bounded in terms of the area under $f(x,y)$ (as illustrated in Fig. 5.8b). Suppose that $h \leq x^*(y_j)$ and let $1 \leq k < m_2$ be such that $kh < x^*(y_j) \leq (k+1)h$. Then

$$h\sum_{i=1}^{k} f(x_i, y_j) \leq \int_{h}^{x_{k+1}} f(x, y_j)dx$$

and

$$h\sum_{i=k+2}^{m_2} f(x_i, y_j) \leq \int_{x_{k+1}}^{1} f(x, y_j)dx$$

Therefore,

$$h\sum_{i=1}^{m_2} f(x_i, y_j) \leq \int_{h}^{1} f(x, y_j)dx + hf(x_{k+1}, y_j) \leq \int_{h}^{1} f(x, y_j)dx + hf(x^*(y_j), y_j) \quad . \tag{5.16}$$

On the other hand, when $h > x^*(y_j)$, all x_i fall on the interval where f is monotonically decreasing. Then we have

$$h\sum_{i=1}^{m_2} f(x_i, y_j) = hf(h, y_j) + h\sum_{i=2}^{m_2} f(x_i, y_j) \leq \int_{h}^{1} f(x, y_j)dx + hf(h, y_j) \quad . \tag{5.17}$$

Let us first assume that $h \leq x^*(y_j)$. By (5.29) and (5.30), we can prove that $x^*(y) \leq y$ for all $y \in (0, 1)$. Then, by (5.13), we have:

$$\int_{h}^{1} f(x, y_j)\, dx \leq \frac{\sqrt{2}}{\pi}\left(\pi + g(\sin h\frac{\pi}{2}, \sin y_j\frac{\pi}{2})\right) \tag{5.18}$$

because $g(z, w) \geq 0$ for $w < z$.

Define the function $G(h, \beta) = g(\sin h\frac{\pi}{2}, \sin \beta h\frac{\pi}{2})$. Then, the right hand side of 5.18 can be written as $\frac{2}{\sqrt{2}\pi}(\pi + G(h, \beta))$ with $\beta = \frac{y_j}{h} \geq 1$. By differentiating G with respect to β we can see that $\frac{\partial G}{\partial \beta} < 0$ for all $h \in (0, 1)$ and $\beta \in [1, +\infty)$. Therefore, G decreases with β and then

$$G(h, \beta) \leq G(h, 1) = \lim_{w \to z^+} g(z, w) = \frac{1}{2}\log\frac{2+\sqrt{2}}{2-\sqrt{2}} - \frac{\pi}{2} \quad . \tag{5.19}$$

for all $h \in (0, 1)$ and $\beta \in [1, +\infty)$. We can then bound (5.18) by

$$\int_{h}^{1} f(x, y_j)\, dx \leq \frac{1}{\sqrt{2}\pi}\left(\pi + \log\frac{2+\sqrt{2}}{2-\sqrt{2}}\right) \quad . \tag{5.20}$$

On the other hand, by (5.14), we have

$$hf(x^*(y_j), y_j) \leq \frac{3}{4\sqrt[4]{3}} h\cot\beta h\frac{\pi}{2} \quad . \tag{5.21}$$

We can prove, moreover, that the right hand side of (5.21) decreases with β and h and therefore we have

$$hf(x^*(y_j), y_j) \leq \frac{3}{4\sqrt[4]{3}} h\cot h\frac{\pi}{2} \leq \frac{3}{2\sqrt[4]{3}\pi} \quad . \tag{5.22}$$

By replacing (5.20) and (5.22) in (5.16), we have

$$h\sum_{i=1}^{m_2} f(x_i, y_j) \leq \frac{1}{\sqrt{2}\pi}\left(\pi + \log\frac{2+\sqrt{2}}{2-\sqrt{2}}\right) + \frac{3}{2\sqrt[4]{3}\pi} = 1.4666 \tag{5.23}$$

and therefore, by (5.15), we have $\|V\|_1 \leq \sqrt{\alpha}\ 0.7333$ when $h \leq x^*(y_j)$.

On the other hand, when $h > x^*(y_j)$, by (5.17) we have

$$h\sum_{i=1}^{m_2} f(x_i, y_j) \leq \int_{x^*(y_j)}^{1} f(x, y_j)dx + hf(h, y_j) \quad . \tag{5.24}$$

By (5.13),

$$\int_{x^*(y_j)}^{1} f(x, y_j)dx \leq \frac{2}{\sqrt{2}\pi} \cos y_j \frac{\pi}{2} \left(\pi + g(\sin x^* \frac{\pi}{2}, \sin y_j \frac{\pi}{2})\right) \leq \frac{2}{\sqrt{2}\pi} \left(\pi + G(x^*, \frac{y_j}{x^*})\right)$$

and by (5.19) we have

$$\int_{x^*(y_j)}^{1} f(x, y_j)dx \leq \frac{1}{\sqrt{2}\pi} \left(\pi + \log \frac{2+\sqrt{2}}{2-\sqrt{2}}\right) \quad . \tag{5.25}$$

Since $f(h, y_j) \leq f(x^*(y_j), y_j)$ and by 5.22 we have

$$hf(h, y_j) \leq \frac{3}{2\sqrt[4]{3}\pi} \tag{5.26}$$

By replacing (5.25) and (5.26 in (5.24), we have

$$h\sum_{i=1}^{m_2} f(x_i, y_j) \leq 1.4666$$

and then by (5.15), we have $\|V\|_1 \leq \sqrt{\alpha}0.7333$ also when $h > x^*(y_j)$. Therefore, $\|V\|_1 \leq \sqrt{\alpha}0.7333$ holds for all $h < 1$. ∎

The result of Theorem 5.2 is important, first because it shows that the preconditioner M_C does not deteriorate when one or both subdomains become flat, second because it gives a small, numerical bound which guarantees fast convergence.

In experiments on L-shaped domains with many different aspect ratios, condition numbers larger than 1.2 were not observed, which indicates that the 2-norm of $\hat{B}^T\hat{B}$ could perhaps be bounded by a number smaller than 0.537. However, this bound is fairly tight for $\|V\|_1\|V\|_\infty$, since numerical experiments with large values of n and m_2 show that $\|V\|_1\|V\|_\infty$ approaches the value 0.5 (we can show, for example, that for $n+1 = m_1+1 = m_2+1 = 10^6$, we have $\|V\|_1\|V\|_\infty = 0.495$). Therefore, if a tighter bound is desired for the condition number, one would need to bound the 2-norm of $\hat{B}^T\hat{B}$ directly.

We next briefly discuss how the parameter n_3 (or, respectively, m_1) affects the performance of preconditioner M_4 (respectively M_5). Clearly, as n_3 tends to zero for large m_2, the domain Ω approaches the shape of a perfect rectangle. The preconditioner M_4 should reflect this by becoming the exact boundary operator. In other words, $\kappa(\hat{C}_4)$ should approach one. We can verify that this is the case as follows: v_{ij} in (5.10) depends on n_3 only through $\lambda_i(m_2, n, n_3)$ (defined in (3.8)). When the aspect ratio $\frac{n_3+1}{m_2+1}$ tends to zero (i.e. Ω_3 becomes thinner), $\lambda_i(m_2, n, n_3)$ tends to infinity and therefore v_{ij} tends to zero. However, we can see that this dependency is very weak, because $\lambda_j(m_2, n, n_3)$ tends rapidly to an asymptotic value independent of n_3 as such aspect ratio grows. Only the fact that

$$\lambda_j(m_2, n, n_3) \geq 2\sqrt{\sigma_j} \tag{5.27}$$

is used in the proof of Theorem 5.2, which is true for all n_3. The discussion above implies that the performance of M_4 as a preconditioner for C_4 is fairly independent of how irregular the region is.

Incidentally, whereas only (5.27) was used in the proof of Theorem 5.2, for other preconditioners such as the ones given in [18, 4] and [26] the preconditioned capacitance matrix always has the form $X + \tilde{B}^T\tilde{B}$, for some operator $\tilde{B}$ to which the bounds (a) and (b) of Theorem 5.2 can also be applied, as long as (5.27) holds. The bound given in (c), however, does not hold for other preconditioners, for which the norm of X may grow when the aspect ratio α of the domain Ω_2 decreases.

Appendix

Lemma 5.3 *Let*

$$f(x,y) = \frac{\sqrt{\sin x\frac{\pi}{2}}\ \cos x\frac{\pi}{2}\ \sqrt{\sin y\frac{\pi}{2}}\ \cos y\frac{\pi}{2}}{\sin^2 x\frac{\pi}{2} + \sin^2 y\frac{\pi}{2}} \quad .$$

Given $a, b \in (0,1)$, let $y \in (0,1)$ such that $a \le y < b$. Then,

$$\int_a^b f(x,y)\ dx = \frac{2}{\sqrt{2\pi}} \cos y\frac{\pi}{2} \left(\pi - g(\sin b\frac{\pi}{2}, \sin y\frac{\pi}{2}) + g(\sin a\frac{\pi}{2}, \sin y\frac{\pi}{2})\right)$$

where

$$g(z,w) = \frac{1}{2}\log\frac{z+\sqrt{2zw}+w}{z-\sqrt{2zw}+w} + arctan\frac{\sqrt{2zw}}{z-w} \quad .$$

Proof: By replacing $z = \sin x\frac{\pi}{2}$ and $w = \sin y\frac{\pi}{2}$ in $f(x,y)$ and defining

$$F(z,w) = \frac{\sqrt{zw}}{z^2+w^2} \quad , \tag{5.28}$$

we get

$$\begin{aligned}
\int_a^b f(x,y) \qquad dx &= \frac{2}{\pi}\cos y\frac{\pi}{2} \int_{\sin a\frac{\pi}{2}}^{\sin b\frac{\pi}{2}} F(z,w)dz \\
&= \frac{\sqrt{2}}{\pi}\cos y\frac{\pi}{2}\left(-\frac{1}{2}\log\frac{z+\sqrt{2zw}+w}{z-\sqrt{2zw}+w} + \arctan\frac{\sqrt{2zw}}{w-z}\right)\Bigg|_{\sin a\frac{\pi}{2}}^{\sin b\frac{\pi}{2}} \\
&= \frac{2}{\sqrt{2\pi}}\cos y\frac{\pi}{2}\left(\pi - g(\sin b\frac{\pi}{2}, \sin y\frac{\pi}{2}) + g(\sin a\frac{\pi}{2}, \sin y\frac{\pi}{2})\right) \quad .
\end{aligned}$$

∎

Lemma 5.4 *Given $y \in (0,1)$, there exists a unique $x^*(y) \in (0,1)$ such that:*

$$\max_{0<x<1} f(x,y) = f(x^*,y) \quad ;$$

$f(.,y)$ is monotonically increasing on the interval $(0,x^*)$ and $f(.,y)$ is monotonically decreasing on $(x^*,1)$. Moreover, $f(x^*,y)$ is bounded by

$$f(x^*,y) \le \frac{3}{4\sqrt[4]{3}}\cot y\frac{\pi}{2} \quad .$$

Proof: The partial derivative of f with respect to x is given by:

$$\frac{\partial f}{\partial x} = \xi(x,y)(\sin^2 x\frac{\pi}{2} - z_-)(\sin^2 x\frac{\pi}{2} - z_+) \quad ,$$

where $\xi(x,y) > 0$ for all $x, y \in (0,1)$ and

$$z_\pm = \frac{3}{2}(1+\sin^2 y\frac{\pi}{2}) \pm \sqrt{\frac{9}{4}(1+\sin^2 y\frac{\pi}{2})^2 - \sin^2 y\frac{\pi}{2}} \quad .$$

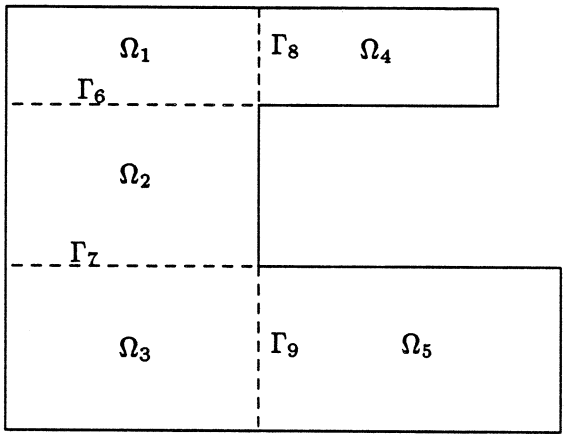

Figure 5.9: C-shaped domain

It can be shown that

$$0 < z_- \leq \frac{\sin^2 y\frac{\pi}{2}}{3} < 1 \tag{5.29}$$

and $z_+ > 1$. Therefore, $\frac{\partial f}{\partial x} > 0$ for $x < x^*$ and $\frac{\partial f}{\partial x} < 0$ for $x > x^*$, where x^* is the unique solution in $(0,1)$ to

$$\sin^2 x^* \frac{\pi}{2} = z_- \quad . \tag{5.30}$$

Therefore, f has a unique maximum in $(0,1)$ at x^*. Moreover, since for all $x \in (0,1)$, we have

$$f(x,y) \leq F(\sin x\frac{\pi}{2}, \sin y\frac{\pi}{2})$$

where F is defined by (5.28), then

$$f(x^*,y) \leq \max_{0<z<1} F(z, \sin y\frac{\pi}{2}) = \frac{3}{4\sqrt[4]{3}} \cot y\frac{\pi}{2} \quad .$$

∎

5.4 C-shaped Regions

Some of the expressions and results of the previous section are more general than they appear and can be used as basic components for more complicated regions that are unions of rectangles. For example, a C-shaped region can be subdivided as indicated in Fig. 5.9.

Similar to L-shaped domains, this region can be separated in three rectangles by either Γ_6 and Γ_7, or Γ_8 and Γ_9. By ordering the variables in Ω_i before those on Γ_j, the matrix A that represents the discrete differential operator on Ω can be written in block form as follows:

$$\begin{pmatrix} A_\Omega & P \\ P^T & A_\Gamma \end{pmatrix} \begin{pmatrix} u_\Omega \\ u_\Gamma \end{pmatrix} = \begin{pmatrix} f_\Omega \\ f_\Gamma \end{pmatrix} \quad , \tag{5.31}$$

where

$$A_\Omega = \begin{pmatrix} A_{11} & & & & & \\ & A_{22} & & & & \\ & & A_{33} & & & \\ & & & A_{44} & & \\ & & & & A_{55} & \end{pmatrix} \quad , \quad A_\Gamma = \begin{pmatrix} A_{66} & & & & \\ & A_{77} & & & \\ & & A_{88} & & \\ & & & A_{99} & \end{pmatrix} \quad ,$$

and

$$P = \begin{pmatrix} A_{16} & 0 & A_{18} & 0 \\ A_{26} & A_{27} & 0 & 0 \\ 0 & A_{37} & 0 & A_{39} \\ 0 & 0 & A_{48} & 0 \\ 0 & 0 & 0 & A_{59} \end{pmatrix} \quad .$$

The system

$$C_{67} \begin{pmatrix} u_6 \\ u_7 \end{pmatrix} = g_{67}$$

can be derived by block elimination for the interfaces Γ_6 and Γ_7, where C_{67} is the Schur complement in A of the blocks A_{66} and A_{77}. A multistrip interface operator M_{67} is described in [15] for the problem of a rectangle divided into three strips (Ω_1, Ω_2 and Ω_3). We will analyze M_{67} as a preconditioner for C_{67}.

The operator M_{67} has the following block structure:

$$M_{67} = \begin{pmatrix} H_6 & S \\ S & H_7 \end{pmatrix} \quad ,$$

where

$$\begin{aligned} H_6 &= A_{66} - A_{16}^T A_{11}^{-1} A_{16} - A_{26}^T A_{22}^{-1} A_{26} \quad , \\ H_7 &= A_{77} - A_{37}^T A_{33}^{-1} A_{37} - A_{27}^T A_{22}^{-1} A_{27} \quad , \\ S &= -A_{26}^T A_{22}^{-1} A_{27} \quad . \end{aligned}$$

By using Lemma 2.2, we see that the blocks H_6, H_7 and S have eigenvalue decompositions of the form (3.2). The eigenvalues of H_6 and H_7 are given by $\lambda_j(n, m_1, m_2)$ and $\lambda_j(n, m_2, m_3)$ respectively and the eigenvalues of S are:

$$\delta_j(n, m_2) = -2\sqrt{\sigma_j + \frac{\sigma_j^2}{4}} \left(\frac{\gamma_j^{\frac{m_2+1}{2}}}{1 - \gamma_j^{m_2+1}} \right) \quad . \tag{5.32}$$

Similarly, a system

$$C_{89} \begin{pmatrix} u_8 \\ u_9 \end{pmatrix} = g_{89} \tag{5.33}$$

can be derived for the interfaces Γ_8 and Γ_9, where C_{89} is the Schur complement in A of the blocks A_{88} and A_{99}. The system (5.33) can be preconditioned by a block diagonal preconditioner M_{89}, with diagonal blocks M_8 and M_9. M_8 is the exact interface system for Γ_8 with respect to the subdomains Ω_1 and Ω_4, and M_9 is the exact interface system for Γ_9 with respect to the subdomains Ω_3 and Ω_5. Both M_8 and M_9 have decompositions of the form (1.6).

It can be easily shown that C_Γ, the Schur complement of the blocks A_Γ in A, can be written in block form as:

$$C_\Gamma = \begin{pmatrix} & M_{67} & Q_{SE} & 0 \\ & & 0 & Q_{NE} \\ Q_{SE}^T & 0 & M_8 & 0 \\ 0 & Q_{NE}^T & 0 & M_9 \end{pmatrix} \quad ,$$

where Q_{SE} and Q_{NE} are the operators that describe the interaction between perpendicular interfaces, defined in Chapter 2. Again, by applying Theorem 2.1, we can prove that both ways of dividing the domain are equivalent, in the sense that the asymptotic convergence rate of an iterative method is the

same when applied to C_{67} with preconditioner M_{67} as when applied to C_{89} with preconditioner M_{89}. The preconditioned interface system for Γ_6 and Γ_7 can be written in the form

$$\hat{C}_{67} \equiv M_{67}^{-1/2} C_{67} M_{67}^{-1/2} = I - \hat{B}^T \hat{B}$$

and similarly,

$$\hat{C}_{89} \equiv \begin{pmatrix} M_8 & 0 \\ 0 & M_9 \end{pmatrix}^{-1/2} C_{89} \begin{pmatrix} M_8 & 0 \\ 0 & M_9 \end{pmatrix}^{-1/2} = I - \hat{B}\hat{B}^T \quad ,$$

where $\hat{B} \in R^{(m_1+m_3)\times 2n}$ and

$$\hat{B} = \begin{pmatrix} M_8 & 0 \\ 0 & M_9 \end{pmatrix}^{-1/2} \begin{pmatrix} Q_{SE}^T & 0 \\ 0 & Q_{NE}^T \end{pmatrix} (M_{67})^{-1/2} \quad . \tag{5.34}$$

The condition numbers of $\hat{C}_{67}$ and $\hat{C}_{89}$ are bounded by

$$\kappa(\hat{C}_{67}) \leq \frac{1}{1 - \|\hat{B}^T \hat{B}\|_2} \qquad \text{and} \qquad \kappa(\hat{C}_{89}) \leq \frac{1}{1 - \|\hat{B}^T \hat{B}\|_2} \quad .$$

Define V as the following unitary tranformation of $\hat{B}$:

$$V = \begin{pmatrix} W_{m_1} & 0 \\ 0 & W_{m_3} \end{pmatrix} \hat{B} \begin{pmatrix} W_n & 0 \\ 0 & W_n \end{pmatrix} \quad .$$

Then $\|V\| = \|\hat{B}\|$. The matrix V can be written as a block two by two matrix

$$V = \begin{pmatrix} V_{66} & V_{67} \\ V_{76} & V_{77} \end{pmatrix} \quad , \tag{5.35}$$

whose block elements have expressions similar to the matrix V for L-shaped regions, namely,

$$V_{66} = W_{m_1} M_8^{-1/2} Q_{SE}^T W_n R_6 \tag{5.36}$$

$$V_{67} = W_{m_1} M_8^{-1/2} Q_{SE}^T W_n R_- \tag{5.37}$$

$$V_{76} = W_{m_3} M_9^{-1/2} Q_{NE}^T W_n R_- \tag{5.38}$$

$$V_{77} = W_{m_3} M_9^{-1/2} Q_{NE}^T W_n R_7 \tag{5.39}$$

where Q_{ij} are defined by (2.15) and R_6, R_7 and R_- are diagonal matrices such that:

$$\begin{pmatrix} R_6 & R_- \\ R_- & R_7 \end{pmatrix} = \begin{pmatrix} W_n & 0 \\ 0 & W_n \end{pmatrix} M_{67}^{-1/2} \begin{pmatrix} W_n & 0 \\ 0 & W_n \end{pmatrix} \quad .$$

For the case when $m_1 = m_3 \leq m_2$, a simple expression can be found for R_6, R_7 and R_-, namely $R_6 = R_7 = R_+$, with the diagonal elements of $R_\pm$ given by

$$r_j^\pm = \frac{1}{2} \left(\frac{1}{\sqrt{\lambda_j - |\delta_j|}} \pm \frac{1}{\sqrt{\lambda_j + |\delta_j|}} \right) \quad , \tag{5.40}$$

where λ_j is $\lambda_j(n, m_1, m_2)$, given by (3.8) and δ_j is $\delta_j(n, m_2)$, given by (5.32). Arguments similar to those in Theorem 5.2 can be applied to the following:

Theorem 5.3 *Consider a C-shaped region like Fig.5.9, where $m_1 = m_3 \leq m_2$ and α is the aspect ratio for the domain Ω_1 or Ω_3 in the picture, i.e. $\alpha = \frac{m_1+1}{n+1}$. Then,*

a) $\|V\|_1 \leq \sqrt{\alpha}\ 0.7877$ *and* $\|V\|_\infty \leq \frac{1}{\sqrt{\alpha}}\ 0.7877$.

b) $\|\hat{B}^T\hat{B}\|_2 \leq \|\hat{B}\|_2^2 = \|V\|_2^2 \leq \|V\|_1\|V\|_\infty \leq 0.62.$

c) $\kappa(\hat{C}_{67}) \leq 2.63$ *and* $\kappa(\hat{C}_{89}) \leq 2.63$ *for all meshsizes and all C-shaped regions such that* $m_1 = m_3 \leq m_2$.

Proof: Define the function

$$f(x) = \frac{1 + x - \sqrt{x}}{1 - x} \quad .$$

We can easily prove that $f(x) \geq 0.866$ for all $x \in [0, 1)$. By (5.32) and (3.8), we have

$$|\lambda_j(n, m_1, m_2)| - \delta_j(n, m_2) = \left(\frac{1 + \gamma_j^{m_1+1}}{1 - \gamma_j^{m_1+1}} + \frac{1 + \gamma_j^{m_2+1}}{1 - \gamma_j^{m_2+1}} - 2\frac{\gamma_j^{\frac{m_2+1}{2}}}{1 - \gamma_j^{m_2+1}} \right) \sqrt{\sigma_j + \frac{\sigma_j^2}{4}} \quad .$$

Since $\gamma_j < 1$ and $m_1 \leq m_2$, we have

$$\begin{aligned} |\lambda_j(n, m_1, m_2)| - \delta_j(n, m_2) &\geq 2\left(\frac{1 + \gamma_j^{m_2+1}}{1 - \gamma_j^{m_2+1}} - \frac{\gamma_j^{\frac{m_2+1}{2}}}{1 - \gamma_j^{m_2+1}} \right) \sqrt{\sigma_j + \frac{\sigma_j^2}{4}} \\ &= 2\sqrt{\sigma_j + \frac{\sigma_j^2}{4}} f(\gamma_j^{m_2+1}) \geq 1.73\sqrt{\sigma_j + \frac{\sigma_j^2}{4}} \quad . \end{aligned} \tag{5.41}$$

Lemma 2.3 gives expressions for the elements of $W_{m_3} Q_{NE}^T W_n$ and $W_{m_3} Q_{SE}^T W_n$. We can see that they have the same absolute values. Also, both M_8 and M_9 have eigenvalues that are bounded from below by $4 \sin \frac{j\pi}{2(m_1+1)}$. These facts can be used to prove a bound on the 1-norm of (5.36) to (5.39).

By (5.35), (5.40) and (5.41), we can see that $\|V\|_1$ is bounded by

$$\|V\|_1 \leq \frac{1}{\sqrt{0.866}} \left(\frac{\sqrt{\alpha}}{2} \frac{1}{m_1 + 1} \max_{1 \leq j \leq n} \sum_{i=1}^{m_1} \frac{\sqrt{\sin x_i \frac{\pi}{2}}\ \cos x_i \frac{\pi}{2}\ \sqrt{\sin y_j \frac{\pi}{2}}\ \cos y_j \frac{\pi}{2}}{\sin^2 x_i \frac{\pi}{2} + \sin^2 y_j \frac{\pi}{2}} \right),$$

where $x_i = i/(m_1 + 1)$ and $y_j = j/(n + 1)$, for $i = 1, \ldots m_1$ and $j = 1, \ldots n$. The proof of Theorem 5.2 applies now to the expression in parentheses. ∎

Chapter 6

Parallel Domain-Decomposed Fast Poisson Solvers

6.1 Introduction

In this chapter, we are concerned with the parallel solution of constant-coefficient problems on regular domains. We are primarily interested in coarse granularity. More specifically, we consider the case $p << m$, where p is the number of processors and m is the number of meshpoints in one spatial dimension.

Domain decomposition methods are a natural choice, because the domain can be divided into as many subdomains as processors and the computation corresponding to each subdomain can be assigned to a different processor. In the case of regular domains, the domain is divided into strips in the two-dimensional case and horizontal slices in the three-dimensional case. Then, block elimination is applied to the discretized equations in order to form a system for the separator variables.

In Chapter 4 we derived expressions for the capacitance matrix C for constant-coefficient problems on rectangular domains divided into strips. We showed that the interface system can be solved by fast direct methods using FFT's. Since the subdomain problems can also be solved by fast direct methods, we thus have a domain-decomposed fast direct Poisson solver which is easily parallelizable. Algorithm DDFAST is presented in Section 6.2.

Unless the strips are extremely thin — i.e. $p \approx m$ — most of the computing time is spent in the solution of the independent subdomain problems. The solution of the interface system only involves communication of boundary values.

One desirable property of this method is its black box modularity. Any sequential fast direct solver can be applied to solve the subdomain problems. The only part of the algorithm which requires specialized code for a particular parallel machine is the solution of the interface system. A second, more important property is that it provides a framework for the implementation of efficient parallel preconditioners for variable-coefficient problems for which fast direct methods are not available. We discuss these preconditioners in Chapter 7.

The arithmetic complexity of the algorithm depends mostly on the particular method chosen as a subdomain solver. A naive implementation requires solving two problems on each subdomain — one with zero boundary conditions on the interface, one with the exact solution on the interface — thus increasing the work asymptotically by a factor of 2. We show, however, that the particular structure of these two systems can be exploited for each particular choice of subdomain solver, thus saving a factor of two in the leading term of the asymptotic complexity. Efficient implementations of the algorithm corresponding to some standard choices of fast direct solvers for the subdomain problems are discussed in Sections 6.3 and 6.4. In Section 6.5 we discuss details of the implementation on hypercube architectures and in Section 6.6 we show some experimental results obtained on an Intel iPSC/2 multiprocessor system.

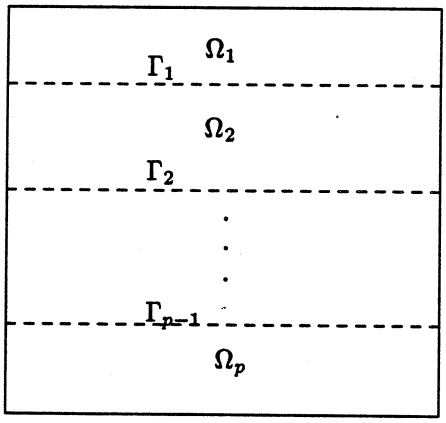

Figure 6.1: Rectangular domain divided into strips

6.2 Domain-Decomposed Fast Poisson Solvers

Consider a five-point discretization of the problem

$$Lu = f \quad \text{in } \Omega \quad \text{with} \quad u = u_b \quad \text{on } \partial\Omega \quad , \tag{6.1}$$

on an n by m grid. We divide the rectangular domain Ω into p strips as shown in Fig. 6.1 and assume that $m+1$ is a multiple of p with $\frac{m+1}{p} \geq 4$.

Let the system

$$\begin{pmatrix} A_\Omega & P \\ P^T & A_\Gamma \end{pmatrix} \begin{pmatrix} u_\Omega \\ u_\Gamma \end{pmatrix} = \begin{pmatrix} f_\Omega \\ f_\Gamma \end{pmatrix} \tag{6.2}$$

represent the discretization of (6.1) in Ω, where u_Ω corresponds to values on the interior of the strips and u_Γ corresponds to values on the interfaces and assume the natural row-wise ordering of the gridpoints inside each subdomain. When block elimination is applied to (6.2), the problem is reduced to independent problems on the subdomains and the linear system

$$Cu_\Gamma = g \tag{6.3}$$

for the interface unknowns, where C is the Schur complement

$$C = A_\Gamma - P^T A_\Omega^{-1} P \tag{6.4}$$

and

$$g = f_\Gamma - P^T A_\Omega^{-1} f_\Omega \quad . \tag{6.5}$$

In order to compute the right-hand side g, we need to solve $A_\Omega z = f_\Omega$. The matrix A_Ω is block diagonal. The p components z_i of the corresponding partition of z are the solutions of independent subdomain problems, given by the restrictions of (6.1) to Ω_i with zero Dirichlet boundary conditions at the interfaces Γ_i:

$$A_{ii} z_i = f_{\Omega_i} \quad . \tag{6.6}$$

Once the interface system is solved, the problem is decoupled and the solution u_Ω on the subdomains can be computed by solving

$$A_\Omega u_\Omega = f_\Omega - P u_\Gamma \quad . \tag{6.7}$$

Again, since A_Ω is block diagonal, this equation represents p independent subproblems of the form

$$A_{ii} u_{\Omega_i} = f_{\Omega_i} - P_{i,i-1} u_{\Gamma_{i-1}} - P_{ii} u_{\Gamma_i} \tag{6.8}$$

where P_{ij} are the blocks of the submatrix P, with

$$P = \begin{pmatrix} P_{1,1} & & & & \\ P_{2,1} & P_{2,2} & & & \\ & \ddots & \ddots & & \\ & & P_{p-1,p-2} & P_{p-1,p-1} & \\ & & & P_{p,p-1} & \end{pmatrix}$$

Each block P_{ij} corresponds to the coupling between the unknowns in the subdomain Ω_i and those on the interface Γ_j. Solving (6.8) means solving for u on each subdomain with the computed values of u_{Γ_i} as boundary conditions on the interfaces.

In Chapter 4 we showed that the Schur complement (6.4) is given by

$$C = \begin{pmatrix} H_1 & B_2 & & \\ B_2 & H_2 & \ddots & \\ & \ddots & \ddots & B_{p-1} \\ & & B_{p-1} & H_{p-1} \end{pmatrix} . \tag{6.9}$$

The eigenvalues of H_i and B_i can be derived for some particular cases such as constant or piece-wise constant-coefficients and are summarized in Tables 4.1 and 4.2.

Although the algorithm can be applied to all operators mentioned in Chapter 4 and to strips of different sizes, we will assume for simplicity in this chapter that the operator L has constant coefficients and that all subdomains are of the same size, each subdomain containing $n \times m_0$ gridpoints of the original grid, where $m + 1 = p(m_0 + 1)$. In this case, the blocks of C do not depend on i, so we have $H_i \equiv H$, $B_i \equiv B$, and

$$W^T H W = \Lambda = \text{ diag}(\lambda_1, \ldots, \lambda_n) \tag{6.10}$$

$$W^T B W = D = \text{ diag}(\delta_1, \ldots, \delta_n) \tag{6.11}$$

For example, when L is the Laplacian operator, λ_j and δ_j are given by:

$$\lambda_j = 2\left(\frac{1+\gamma_j^{m_0+1}}{1-\gamma_j^{m_0+1}}\right)\sqrt{\sigma_j + \frac{\sigma_j^2}{4}} \tag{6.12}$$

and

$$\delta_j = -2\frac{\gamma_j^{\frac{m_0+1}{2}}}{1-\gamma_j^{m_0+1}}\sqrt{\sigma_j + \frac{\sigma_j^2}{4}} \quad . \tag{6.13}$$

The interface system is solved by *matrix decomposition*, a fast direct method that combines fast sine transforms and tridiagonal solvers [10].

The resulting algorithm is a fast direct Poisson solver derived from domain decomposition.

6.2.1 Algorithm DDFAST

In order to compute g using (6.5), the systems (6.6) must be solved first. A fast direct solver can be used to solve independent problems on each rectangular subdomain.

The interface system (6.3) is solved by matrix decomposition as follows: by substituting (6.10) and (6.11) in (6.9), the system $Cu_\Gamma = g$ is transformed into:

$$\begin{pmatrix} \Lambda & D & & \\ D & \Lambda & \ddots & \\ & \ddots & \ddots & D \\ & & D & \Lambda \end{pmatrix} \hat{u}_\Gamma = \hat{g} \quad , \tag{6.14}$$

Figure 6.2: **Algorithm DDFAST**

Step 1: For $i = 1, \ldots, p$, solve
$$A_{ii} z_i = f_{\Omega_i}.$$
Step 2: Form the right-hand side for the interface system:
for $i = 1, \ldots, p-1$
$$g_{\Gamma_i} = f_{\Gamma_i} - P_{ii}^T z_i - P_{i+1,i}^T z_{i+1}.$$
Step 3: Solve the interface system:
i) Use fast sine transforms to compute
for $i = 1, \ldots, p-1$,
$\hat{g}_{\Gamma_i} = W^T g_{\Gamma_i}$.
ii) Solve the transformed system (6.14) by reordering the unknowns and solving n tridiagonal systems of size $p-1$.
iii) Use fast sine transforms to compute
for $i = 1, \ldots, p-1$,
$u_{\Gamma_i} = W \hat{u}_{\Gamma_i}$.
Step 4: For $i = 1, \ldots, p$, solve:
$$A_{ii} u_{\Omega_i} = f_{\Omega_i} - P_{i,i-1} u_{\Gamma_{i-1}} - P_{ii} u_{\Gamma_i},$$
with $u_{\Gamma_0} = u_{\Gamma_p} = 0$.

where the components of $\hat{u}_\Gamma$ and $\hat{g}$ are given by

$$\hat{u}_{\Gamma_i} = W^T u_{\Gamma_i} \quad \text{and} \quad \hat{g}_{\Gamma_i} = W^T g_{\Gamma_i} \quad .$$

Each component of the new right-hand side $\hat{g}$ can be computed with a fast Fourier sine transform.

By reordering the unknowns and rearranging the equations accordingly, (6.14) can be written as n decoupled tridiagonal systems of size $p-1$, given by the matrices

$$T_j = \begin{pmatrix} \lambda_j & \delta_j & & \\ \delta_j & \lambda_j & \ddots & \\ & \ddots & \ddots & \delta_j \\ & & \delta_j & \lambda_j \end{pmatrix} \tag{6.15}$$

The solution at the interfaces is then computed by applying inverse sine transforms to each $\hat{u}_{\Gamma_i}$, i.e., $u_{\Gamma_i} = W \hat{u}_{\Gamma_i}$. Finally, we can solve the decoupled problem (6.8) by fast direct solvers on the rectangular subdomains.

The domain-decomposed fast Poisson solver DDFAST is summarized in Fig. 6.2. The algorithm is naturally suited for parallel implementation. Steps 1 and 4, which are the most computationally intensive, are completely parallelizable because each involves the solution of p independent Poisson problems. Steps 2 and 3 have lower order computational complexity because they only involve the interface values. The fast sine transform computations in Step 3 can be done locally. Steps 2 and 4 only require communication of boundary values. Finally, the right-hand sides for the n tridiagonal systems of size $p-1$ of Step 3 are distributed among the processors. The most efficient method for solving these systems will depend on the particular parallel machine. In Section 6.5 we will discuss details of the parallel implementation of DDFAST, for the case where the processors are connected in a hypercube configuartion.

6.2.2 Complexity of DDFAST

The arithmetic complexity of algorithm DDFAST is

$$C_{DD}(m,n,p) = 2kC_P(\frac{m+1}{p}, n) + C_{interf}(n,p) + \mathcal{O}(pn) \quad , \tag{6.16}$$

where $C_P(m,n)$ represents the complexity of a Poisson solver on an m by n rectangular grid and $C_{interf}(n,p)$ denotes the complexity for solving the interface system, with

$$C_{interf}(n,p) = 2(p-1)C_s(n+1) + nC_t(p-1) \quad .$$

Since the complexity of any Poisson solver applied to the subdomain problems is at least linear in m_0 (e.g. $\mathcal{O}(m_0 n \log n)$ or $\mathcal{O}(nm_0 \log m_0)$) and the other terms in (6.16) have lower complexity, the asymptotic complexity of algorithm DDFAST is at most twice the complexity of the same subdomain solver applied to the entire domain Ω. In other words, asymptotically, for any p, $C_{DD}(m,n,p) \leq 2C_P(m,n)$. Moreover, when the m-dependency is stronger than linear, we can take advantage of this fact in a divide and conquer fashion. For example, consider a Poisson solver such that $C_P(m,n) = \mathcal{O}(mn \log m)$. Then, $C_{DD} = \mathcal{O}(mn \log \frac{m+1}{p})$, and for the optimal $p = \mathcal{O}(\frac{m}{\log m})$, C_{DD} becomes $\mathcal{O}(mn \log\log m)$.

Algorithm DDFAST also has the advantage that the Poisson problems in Steps 1 and 4 can be solved by any sequential fast direct solver in a black box fashion. The price to be paid is the factor 2 in the leading term of (6.16). That is, if we view the subdomain solver as a black box, we must independently solve two problems on each subdomain, namely Steps 1 and 4. Note, however, the following facts about these two subdomain solves: first, only the values of z_i at the gridpoints near the interfaces are needed for the computation of g in Step 2; second, the right-hand sides for the equations in Step 4 differ from the right hand sides of Step 1 only at the mentioned gridpoints. Thus, given a particular subdomain solver, the two solves can be tailored (sacrificing modularity), to save the leading factor of two. In the next two sections we will discuss efficient implementations of DDFAST based on Fourier analysis and odd-even block-cyclic reduction.

6.3 Subdomain Solvers using Fourier Analysis

We first consider the fast direct solver from FISHPACK [34, 10], which we will call MD and which is based on the technique of matrix decomposition, described earlier in this chapter. Algorithms DD1 and DD2 are particular instances of algorithm DDFAST, when the subdomain problems of Steps 1 and 4 are solved by MD with row-wise and column-wise numbering of the subdomain interior gridponts. In DD1, sine transforms are computed in the direction parallel to the strips (size n) and tridiagonal systems are solved in the other direction (size m_0). In DD2, sine transforms are computed in the direction perpendicular to the strips (size m_0) and tridiagonal systems are solved in the other direction (size n). While DD1 is a natural choice from the point of view of data storage since we assume that the gridpoints were originally numbered row-wise, the solver that performs fast sine transform computations on shorter vectors has lower complexity. Therefore DD2 is asymptotically faster than DD1. In particular, as we mentioned earlier in this chapter, DD2 can be made $\mathcal{O}(mn \log\log m)$ for a particular choice of p.

6.3.1 Algorithm DD1

In a naive version of Algorithm DD1, the subproblems of Steps 1 and 4 of DDFAST are solved by the MD algorithm, requiring $2(2m-(p-1))$ sine transforms of vectors of dimension n. However, by solving the interface system in Fourier space, we can save two intermediate Fourier transform phases and reduce this to $2m$ transforms.

We outline the algorithm in Fig. 6.3. The equations in Step 3 are reordered, so this corresponds to solving n tridiagonal systems of dimension $p-1$ with matrices (6.15), where λ_j and δ_j are given by (6.12) and (6.13).

Figure 6.3: **Algorithm DD1**

Step 1: First Solve

a) Compute the sine transforms of the right hand side:
$\hat{f} = \text{diag}\,(W^T) f$.

b) Solve $\hat{A}_{ii}\hat{z}_i = \hat{f}_{\Omega_i}$, where
$\hat{A}_{ii} = \text{diag}(W^T) A_{ii}\ \text{diag}(W)$.

Step 2: Form the transformed right-hand side for the interface system, i.e. $\hat{g}_{\Gamma_i} = \hat{f}_{\Gamma_i} - P_{i+1,i}\hat{z}_{i+1} - P_{ii}\hat{z}_i$

Step 3: Solve (6.14), the interface system *in Fourier space.*

Step 4: Second solve

a) Compute $\hat{u}_{\Omega_i}$ by solving
$\hat{A}_{ii}\hat{u}_{\Omega_i} = \hat{f}_{\Omega_i} - P_{ii}\hat{u}_{\Gamma_i} - P_{i,i-1}\hat{u}_{\Gamma_{i-1}}$.

b) Compute the solution at the interior of the subdomains and at the interfaces, by applying the inverse sine transforms to $\hat{u}_{\Omega_i}$ and $\hat{u}_{\Gamma_i}$.

Complexity of Algorithm DD1

Since $C_P = 2m_0C_s(n+1) + C_{t1}nm_0$, the arithmetic complexity of the naive sequential implementation of DD1 is

$$\begin{aligned} C_{black-box-DD1}(n,m,p) &= 4mC_s(n+1) - 2(p+1)C_s(n+1) \\ &\quad +2C_{t1}(mn) + \mathcal{O}(pn) \quad . \end{aligned}$$

The complexity for the efficient implementation can be computed as follows: Steps 1 and 4 involve $2m$ fast sine transforms of dimension n and the solution of $2pn$ tridiagonal systems of dimension m_0. In Step 3, n tridiagonal systems of dimension $p-1$ must be solved. The rest of the computation is also $\mathcal{O}(pn)$. Thus, for the more efficient implementation of DD1 we have:

$$\begin{aligned} C_{DD1}(n,m,p) &= 2mC_s(n+1) + 2pnC_{t1}\left(\frac{m+1}{p}\right) + \mathcal{O}(pn) \\ &= 2mC_s(n+1) + 2C_{t1}nm + \mathcal{O}(pn) \quad . \end{aligned}$$

Asymptotically, this version of DD1 has the same complexity as the algorithm MD applied to the whole domain, namely $C_{MD}(n,m) = 2mC_s(n+1) + C_tnm$, but for small values of n, the contribution of the second term may still be significant, so DD1 is a slower *sequential* solver than MD.

6.3.2 Algorithm DD2

Algorithm DD2 corresponds to computing fast sine transforms in the direction perpendicular to the strips (shorter vectors). Even in the sequential case, this algorithm has lower overall complexity than the subdomain solver when it is applied to the whole domain.

In order to set the notation, we first describe the black-box version of this algorithm. Let the interior gridpoints on each subdomain be numbered by columns instead of by rows. Then, each subproblem can be solved by matrix decomposition with $2n$ Fourier transforms of dimension m_0 and the solution of m_0 tridiagonal systems of dimension n. After reordering the unpnowns and applying fast sine transforms in the y-direction, the equation $\overline{A}_{ii}z_i = f_{\Omega_i}$ is transformed into $\overline{A}_{ii}\overline{z}_i = \overline{f}_{\Omega_i}$. This new equation is reordered to get m_0 tridiagonal systems of dimension n. The black-box version of DD2 is outlined in Fig. 6.4.

Figure 6.4: **Algorithm DD2**

Black-box Implementation

Step 1: First Solve

For $i = 1, \ldots, p$

a) compute $\overline{f}_{\Omega_i}$ (n sine transforms of dimension m_0)

b) solve $\overline{A}_{ii}\overline{z}_i = \overline{f}_{\Omega_i}$ (m_0 tridiagonal systems of dimension n)

c) compute z_i by applying inverse fast sine transforms to $\overline{z}_i$.

Step 2: Form the right-hand side for the interface system:

for $i = 1, \ldots, p-1$

$g_{\Gamma_i} = f_{\Gamma_i} - P_{ii}^T z_i - P_{i+1,i}^T z_{i+1}$.

Step 3: Solve the interface system:

- Use fast sine transforms to compute $\hat{g}_{\Gamma_i} = W^T g_{\Gamma_i}$.
- Solve the transformed system (6.14) by reordering the unknowns and solving n tridiagonal systems of size $p-1$.
- Use fast sine transforms to compute $u_{\Gamma_i} = W\hat{u}_{\Gamma_i}$.

Step 4: Second solve

a) Compute $\overline{h}_{\Omega_i}$, the sine transform of:
Let $h_{\Omega_i} = f_{\Omega_i} - P_{i,i-1}u_{\Gamma_{i-1}} - P_{ii}u_{\Gamma_i}$.

b) Solve the m_0 tridiagonal systems $\overline{A}_{ii}\overline{u}_{\Omega_i} = \overline{h}_{\Omega_i}$.

c) Compute u_{Ω_i} by fast sine transforms.

Efficient Implementation of DD2

The complexity of Step 1-c is $\mathcal{O}(nm_0 \log m_0)$ to compute all entries of z_i by fast sine transforms. Since only the values of z_i on the first and last rows of gridpoints in Ω_i are needed to compute $P_{ii}^T z_i$ and $P_{i,i-1}^T z_i$, they can be computed from $\overline{z}_i$ in $\mathcal{O}(nm_0)$ time by direct calculation. Denote the values of z_i on the first row of gridpoints by $(z_{11}^i, z_{21}^i, \ldots, z_{n1}^i)^T$ and the values of z_i on the last row of gridpoints by $(z_{1m_0}^i, z_{2m_0}^i, \ldots, z_{nm_0}^i)^T$. Then,

$$z_{j1}^i = \sqrt{\frac{2}{m_o+1}} \sum_{s=1}^{m_0} \sin\left(\frac{s\pi}{m_0+1}\right) \overline{z}_{js}^i \quad , \tag{6.17}$$

$$z_{jm_0}^i = \sqrt{\frac{2}{m_o+1}} \sum_{s=1}^{m_0} \sin\left(\frac{m_0 s\pi}{m_0+1}\right) \overline{z}_{js}^i \quad . \tag{6.18}$$

We can also save some operations in Step 4 by taking advantage of the fact that, since the vector $p_i = -P_{i,i-1}u_{\Gamma_{i-1}} - P_{ii}u_{\Gamma_i}$ is sparse, its sine transform $\overline{p}_i$ can be computed in $\mathcal{O}(nm_0)$ time. The elements of $\overline{p}_i$ are given by:

$$\sqrt{\frac{2}{m_o+1}} \sin\left(\frac{s\pi}{m_0+1}\right) \quad (u_{\Gamma_i,j} + (-1)^s u_{\Gamma_{i-1},j}) \quad . \tag{6.19}$$

Then $\overline{h}_{\Omega_i}$ can be computed by $\overline{h}_{\Omega_i} = \overline{f}_{\Omega_i} + \overline{p}_i$. We summarize the efficient implementation of Algorithm DD2 in Fig. 6.5.

Complexity of DD2

The efficient version of algorithm DD2 only requires $2n$ fast sine transforms of dimension m_0, as opposed to $4n$ for the naive implementation, and the complexity of Steps 1-(c) and 4-(a) is $\mathcal{O}(nm_0)$ instead of $\mathcal{O}(nm_0 \log m_0)$.

Figure 6.5: **Algorithm DD2**

Efficient Implementation

Step 1: First Solve

For $i = 1, \ldots, p$,

a) compute the sine transforms of f_{Ω_i}
(n transforms of dimension m_0)

b) solve $\overline{A}_{ii}\hat{z}_i = \hat{f}_{\Omega_i}$
(m_0 tridiagonal systems of dimension n),

c) compute (only) $P_{ii}^T z_i = (z_{1m_0}^i, \ldots, z_{nm_0}^i)$ for $i < p$
by (6.17) and $P_{i,i-1}^T z_i = (z_{11}^i, \ldots, z_{n1}^i)$ for $i > 1$ by (6.18).

Step 2: Form the right-hand side for the interface system:

$g_{\Gamma_i} = f_{\Gamma_i} - P_{ii}^T z_i - P_{i+1,i}^T z_{i+1}$, $i = 1, \ldots, p-1$.

Step 3: Solve the interface system:

- Use fast sine transforms to compute $\hat{g}_{\Gamma_i} = W^T g_{\Gamma_i}$
- Solve the transformed system (6.14) by reordering the unknowns and solving n tridiagonal systems of size $p-1$.
- Use fast sine transforms to compute $u_{\Gamma_i} = W\hat{u}_{\Gamma_i}$.

Step 4: Second solve

a) update the sine transforms of the right-hand side by computing only the sine transforms of $q_i = -P_{i,i-1}u_{\Gamma_{i-1}} - P_{ii}u_{\Gamma_i}$, which is a sparse vector (direct calculation, i.e., no fast sine transforms)

b) solve the m_0 tridiagonal systems $\overline{A}_{ii}\overline{u}_{\Omega_i} = \overline{h}_{\Omega_i}$.

c) compute u_{Ω_i} by fast sine transforms (dimension m_0).

The complexity of the black-box version of DD2 can be obtained from (6.16) by using

$$C_P(m_0, n) = 2nC_s(m_0) + C_{t1}m_0 n$$

Then

$$\begin{aligned} C_{black-box-DD2}(m,n,p) &= 4pnC_s(\frac{m+1}{p}) + 2C_{t1}mn + 2(p-1)C_s(n+1) \\ &\quad + \mathcal{O}(pn) \quad . \end{aligned}$$

For the efficient implementation of DD2, the complexity of Steps 1-a and 4-c is $pnC_s(m_0)$, and the complexity of Steps 1-b and 4-b is $pm_0C_{t1}n$.

Finally, half of the operations can be saved in (6.17) and (6.18) by taking advantage of the fact that $\sin\left(\frac{m_0 s\pi}{m_0+1}\right) = (-1)^{s+1}\sin\left(\frac{s\pi}{m_0+1}\right)$. Then, Step 1-c requires $2pm_0n$ operations. In Step 4-a, $\overline{h}_{\Omega_i}$ is computed as $\overline{h}_{\Omega_i} = \overline{f}_{\Omega_i} + \overline{p}_i$, where $\overline{p}_i$ is given by (6.19), requiring $3pnm_0$ flops.

Then, the complexity of the first solve is:

$$pnC_s(m_0+1) + C_{t1}pm_0n + 2pm_0n + \mathcal{O}(pn)$$

and the complexity of the second solve:

$$3pm_0n + C_{t1}pm_0n + pnC_s(m_0+1) + \mathcal{O}(pn) \quad .$$

Thus

$$C_{DD2}(m,n,p) = 2pnC_s(\frac{m+1}{p}) + (2C_{t1}+5)mn + 2(p-1)C_s(n+1) + \mathcal{O}(pn) \quad .$$

Numerical Results (sequential case)

In Table 6.1 we compare DD2 with two different implementations of matrix decomposition (MD) using Fourier analysis. The model problem solved was Poisson's equation on the unit square. For reference, we also include times for routine POIS from FISHPACK. In order to be consistent with the parallel experiments, we used one node of an Intel Hypercube IPSC/2 to run these sequential experiments.

Table 6.1:

Sequential runtime (in seconds) for Algorithms MD1 (row-wise fast sine transforms), MD2(column-wise fast sine transforms), routine POIS and DD2(p) with $m+1 = n+1 = \frac{1}{h}$.

SOLVER	$h = 2^{-7}$	$h = 2^{-8}$	$h = 2^{-9}$
MD1	3.37	15.58	63.72
MD2	3.61	16.61	68.09
POIS	5.29	24.45	110.44
DD2(4)	3.66	17.16	69.66
DD2(8)	3.77	14.90	69.61
DD2(16)	3.47	15.33	60.11
DD2(32)	3.91	14.21	61.67
DD2(64)		16.27	57.23
DD2(128)			65.56

The fast sine transforms are applied to the gridpoints row-wise in MD1 and column-wise in MD2. Although these two solvers have the same operation count, their runtimes are different because the data is stored row-wise. The domain-decomposed solver on p strips is DD2(p), with DD2(1) (one domain) defined to be MD2.

The times in Table 6.1 do not have a unique minimum at $p = \mathcal{O}(\frac{(n+1)}{\log_2(n+1)})$ as predicted, because as p approaches n, the complexity for the sine transforms of a vector of length $m_0 = \frac{n+1}{p}$ is not well represented by a function of the form $Cm_0 \log m_0$. This is shown by the runtimes given in Table 6.2 for subroutine SINT of FFTPack [33] (times do not include the initialization routine SINTI). As a result, we see more than one relative minimum on runtimes, but we can see, however, that in all cases DD2(p) is fastest when $p \approx \frac{(n+1)}{\log_2(n+1)}$.

In order to illustrate how the FFT package overhead affects runtimes when the subdomains are narrow and sine transforms are applied to short vectors, we replaced the call to the FFT package with a direct sine transform calculation when $m_0 = 3$, given by a loop of the form:

Direct Sine Transform for $m_0 = 3$

```
do  i= 1, n
    temp1 = (f(i) + f(i+2*n)) / 2.
    temp2 = (f(i) - f(i+2*n)) * sq2
    temp3 =  f(i+n) * sq2
    f(i)      = temp1 + temp3
    f(i+2*n) = temp1 - temp3
    f(i+n)    = temp2
enddo
```

Table 6.2: Fast Sine Transform
Time (msec) for routine SINT

n	3	7	15	31	63	127	255	511	1023
Time	0.19	0.39	1.04	2.11	5.50	11.28	26.50	54.48	130.84

For $n = 127$ and 32 subdomains, for example, we have $m_0 = 3$ and by replacing the call to FFTpack with the above loop, runtime is reduced from 3.91 seconds to 1.88 seconds. Similarly, for $n = 255$ and 64 subdomains, runtime decreases from 16.27 secs. to 8.09 secs. and for $n = 511$ and 128 subdomains, runtime decreases from 65.56 secs. to 32.76 secs.

6.4 Other Approaches

6.4.1 Subdomain Solvers using Block Cyclic Reduction

Another important class of fast direct solvers is based on odd-even block cyclic reduction (CR). Consider the block tridiagonal system

$$\begin{pmatrix} T & Q & & \\ Q & T & \ddots & \\ & \ddots & \ddots & Q \\ & & Q & T \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ x_m \end{pmatrix} \quad , \tag{6.20}$$

where T and Q are n by n symmetric matrices and $x_j, f_j \in R^n$ (note that on a subdomain, we would replace m with m_0). Assume, moreover, that T and Q commute and $m = 2^s - 1$. Briefly, the method consists of $s - 1$ elimination steps and a backsubstitution phase. By multplying the odd rows by Q and the even rows by $-T$ and adding the $(j-1)$-th and the $(j+1)$-th rows to the j-th row for all even j, we get

$$Q^2 x_{j-2} + (2Q^2 - T^2)x_j + Q^2 x_{j+2} = Q f_{j-1} - T f_j + Q f_{j+1}$$

where $x_0 = x_{m+1} = 0$. All the unknowns with odd indexes are thus eliminated and (6.20) is reduced to two decoupled systems, namely a block tridiagonal system for the even unknowns:

$$\begin{pmatrix} T^{(1)} & Q^{(1)} & & \\ Q^{(1)} & T^{(1)} & \ddots & \\ & \ddots & \ddots & Q^{(1)} \\ & & Q^{(1)} & T^{(1)} \end{pmatrix} \begin{pmatrix} x_2 \\ x_4 \\ \vdots \\ x_{m-1} \end{pmatrix} = \begin{pmatrix} f_2^{(1)} \\ f_4^{(1)} \\ \vdots \\ f_{m-1}^{(1)} \end{pmatrix} \quad , \tag{6.21}$$

where $T^{(1)} = 2Q^2 - T^2$, $Q^{(1)} = Q^2$ and $f_j^{(1)} = Q f_{j-1} - T f_j + Q f_{j+1}$, and a block diagonal for the odd unknowns.

$$\begin{pmatrix} T & & & \\ & T & & \\ & & \ddots & \\ & & & T \end{pmatrix} \begin{pmatrix} x_1 \\ x_3 \\ \vdots \\ x_m \end{pmatrix} = \begin{pmatrix} f_1 - Q x_2 \\ f_3 - Q x_2 - Q x_4 \\ \vdots \\ f_m - Q x_{m-1} \end{pmatrix} \quad . \tag{6.22}$$

Figure 6.6: Block Cyclic Reduction Algorithm

Let $T^{(0)} = T$ and $f_j^{(0)} = f_j$ for $j = 1, \ldots, m$.
Reduction Phase:
For $r = 1, \ldots, s-1$, define
$T^{(r)} = 2I - (T^{(r-1)})^2$
and for j multiples of 2^r, let
$f_j^{(r)} = f_{j-2^{r-1}}^{(r-1)} + f_{j+2^{r-1}}^{(r-1)} - T^{(r-1)} f_j^{(r-1)}$
with $f_0^{(r)} = f_{m+1}^{(r)} = 0$ for all r.
Back-Substitution Phase:
The component $x_{2^{s-1}}$ of the solution is computed first
by solving $T^{(s-1)} x_{2^{s-1}} = f^{(s-1)}$.
Then, compute $x_{2^{s-2}}$ and $x_{3\ 2^{s-2}}$ and so on.
At the r-th step ($r = s-2, \ldots, 0$), the components x_j for
$j = 2^r, 32^r, \ldots$ are computed by solving:
$T^{(r)} x_j = f_j^{(r)} - x_{j-2^r}^{(r)} - x_{j+2^r}^{(r)}$
with $x_0 = x_{m+1} = 0$.

The same elimination procedure can be applied recursively to the reduced system (6.21). At the r-th reduction step we have a system of the form

$$\begin{pmatrix} T^{(r)} & Q^{(r)} & & \\ Q^{(r)} & T^{(r)} & \ddots & \\ & \ddots & \ddots & Q^{(r)} \\ & & Q^{(r)} & T^{(r)} \end{pmatrix} \begin{pmatrix} x_{2^r} \\ x_{2\ 2^r} \\ \vdots \\ x_{m+1-2^r} \end{pmatrix} = \begin{pmatrix} f_{2^r}^{(r)} \\ f_{2\ 2^r}^{(r)} \\ \vdots \\ f_{m+1-2^r}^{(r)} \end{pmatrix} . \tag{6.23}$$

By induction it can be shown that $T^{(r)}$ is a polynomial of degree 2^r in the matrices T and Q and moreover,

$$T^{(r)} = -\prod_{j=1}^{2^r} (T + \beta_{rj} Q)$$

with $\beta_{rj} = 2\cos\frac{(2j-1)\pi}{2^{r+1}}$. After $s-1$ reduction steps, a one block system is left to solve, namely

$$T^{(s-1)} x_{2^{s-1}} = f^{(s-1)} \quad .$$

This system can be solved as a sequence of systems of the form $(T + \beta_{rj} Q)u = v$ or by Fourier analysis if T and Q are diagonalizable by Fourier modes. In the backsubstitution phase, block-diagonal systems are solved for indexes which are multiples of 2^{s-2}, 2^{s-3} and so forth, solving (6.22) in the final step.

The block cyclic reduction algorithm is outlined in Fig. 6.6 for the case $Q = I$. For the case of the Laplacian operator, the matrix T is tridiagonal and $T = -2I - K$, where P is given by (2.7). T is also diagonalizable by Fourier modes, with eigenvalues given by $-2 - \sigma_i$. A system of the form $T^{(r)}u = v$ can be solved as a sequence of tridiagonal systems or by Fourier analysis, with the eigenvalues of $T^{(r)}$ given by $\lambda_i = \prod_j (-2 - \sigma_i + 2\cos\frac{(2j-1)\pi}{2^{r+1}})$.

As presented in Fig. 6.6, the algorithm can become very unstable. By computing the right hand sides in a different way, Buneman [9] derived a stable version of the block cyclic reduction algorithm.

The arithmetic complexity of CR is $\mathcal{O}(nm \log m)$. As with the other fast solvers mentioned before, CR can also be used as the subdomain solver in algorithm DDFAST and, as in the previous sections, a factor

of two can be saved in the leading term of (6.16) by exploiting the particular structure of the subdomain problems. First, instead of solving the subproblems in Step 1 completely, the solution z could be computed only at those gridpoints near the interfaces. In the context of the CR algorithm, this corresponds to solving for x_1 and x_m only. Second, the right-hand side for the second solve differs from that of the first solve only at the mentioned gridpoints. This fact can be exploited during the elimination phase of Algorithm CR. The following two lemmas can be easily proved from the description of the algorithm. Similar results hold for Buneman's stable variation.

Lemma 6.1 *If only x_1 and x_m are needed, they can be obtained by computing only x_{2^r} and x_{m+1-2^r} at each step of the back-substitution phase.*

Proof: By induction: $x_{2^{s-1}}$ is computed first and suppose that at the $(r+1)$-th step of the back-substitution phase ($r = s-2, \ldots, 0$), only the components $x_{2(r+1)}$ and $x_{m+1-2(r+1)}$ are computed. Then, since $x_0 = x_{m+1} = 0$, x_{2^r} and x_{m+1-2^r} can be computed by

$$\begin{aligned} x_{2^r} &= \left(T^{(r)}\right)^{-1} (f_{2^r}^{(r)} - x_{2^{r+1}}^{(r)}) \\ x_{m+1-2^r} &= \left(T^{(r)}\right)^{-1} (f_{m+1-2^r}^{(r)} - x_{m+1-2^{r+1}}^{(r)}) \quad . \end{aligned}$$

∎

Lemma 6.2 *If $f_j = 0$ for all $2 \le j \le m-1$, then $f_{2^r}^{(r)} = f_1$, $f_{m+1-2^r}^{(r)} = f_m$ and $f_j^{(r)} = 0$ for $2^r < j < m+1-2^r$.*

Proof: Suppose that $f_j = 0$ for all $2 \le j \le m-1$. Then, since

$$f_j^{(1)} = f_{j-1} + f_{j+1} - Tf_j \quad ,$$

we have $f_2^{(1)} = f_1$, $f_{m-1}^{(1)} = f_m$ and $_j^{(1)} = 0$ for $j = 4, 6, \ldots, m-3$. Similarly, by induction, suppose that $f_{2^{(r-1)}}^{(r-1)} = f_1$, $f_{m+1-2^{(r-1)}}^{(r-1)} = f_m$ and $f_j^{(r-1)} = 0$ for $2^{r-1} < j < m+1-2^{r-1}$. Since

$$f_j^{(r)} = f_{j-2^{r-1}}^{(r-1)} + f_{j+2^{r-1}}^{(r-1)} + -T^{(r-1)} f_j^{(r-1)} \quad ,$$

then we have $f_{2^r}^{(r)} = f_1$, $f_{m+1-2^r}^{(r)} = f_m$ and $f_j^{(r)} = 0$ for $2^r < j < m+1-2^r$. ∎

Lemma 6.1 can be applied to the first subdomain solve (Step 1 of DDFAST). By Lemma 6.2, if only the first and last components of the right hand side are modified, then at each step of the reduction phase, only the first and last components of $f^{(r)}$ need to be updated. This result can be applied to the second subdomain solve (Step 4).

6.4.2 Algorithm FACR

Instead of applying the reduction technique until only one block is left, the process can be stoped at any step and the reduced system (6.23) can be solved by another method. A combination of cyclic reduction and matrix decomposition using Fourier analysis is called FACR(l), which consists of l steps of block cyclic reduction followed by the MD method using Fourier Analysis applied to the reduced equations [27]. The arithmetic complexity of this fast solver can be minimized with respect to l, so that when it is applied to the subdomain problems, its complexity is $\mathcal{O}(nm_0 \log\log m_0)$.

In order to save the leading factor of two in the complexity analysis, the algorithm is considered in two separate parts: first, for the CR part, Lemmas 6.1 and 6.2 can be applied to the l reduction steps and the l back-substitution steps. The second part is the solution of the reduced equations by Fourier analysis. This part is similar to the subdomain solver in DD1. As in DD1, one application of Fourier transforms can be saved in Step 1 and in Step 4, by solving the interface system in Fourier space. During the back-substitution phase for the first solver and the elimination phase of the second solver, the matrices $T^{(r)}$ become diagonal matrices.

6.4.3 Generalized Marching Algorithm

Bank and Rose's Generalized Marching Algorithm (GMA) [2] can be viewed as a particular implementation of the algorithm DDFAST, where the marching method is applied to the subdomain problems. While in the above implementations of DDFAST the number of strips p can be chosen arbitrarily, in the case of GMA, the value of p is chosen in order to preserve stability. The marching method, as a subdomain solver, has optimal order $\mathcal{O}(nm_0)$, but it becomes unstable for large values of m_0.

6.5 Parallel Implementations

Consider a distributed memory multiprocessor with p processors and a decomposition of the rectangular domain Ω into p equal strips. Suppose that the strip Ω_i and its corresponding lower interface Γ_i are mapped into processor $P(i)$ in such a way that adjacent subdomains are assigned to nearest neighbor processors. We also assume that an n by m grid is imposed over the domain, such that $m+1$ is a multiple of p and $m_0 + 1 = \frac{m+1}{p} \geq 4$.

The parallel implementation of algorithm DDFAST is quite straightforward. Step 1 can be done completely in parallel: each processor solves an independent problem on each strip. In Step 2, each processor needs to send and receive a vector of size n in order to compute the right-hand side g for the interface system. In Step 3, the interface system is solved by matrix decomposition. The fast sine transforms are applied to vectors that are local to each processor. The right-hand sides for the resulting n tridiagonal systems are distributed in such a way that each processor contains one element of each right-hand side. Therefore, global data movement is required at this step. This is the part of the algorithm that will be most strongly dependent on the particular machine.

The two best known ways for solving the distributed tridiagonal systems are by balanced cyclic reduction and by data transposition. Which of these is faster will depend upon the architecture, the size of the problem and implementation details (see for example [29, 30]). In our experiments, we used data transposition, i.e., data is moved in such a way that the entire right-hand sides for n/p of the n tridiagonal systems end up in each processor, where the systems are solved locally and then the solutions are transposed back into the original distribution.

Finally, nearest neighbor communication is required in order to update the right-hand side for the second solve (Step 4). The subdomain problems are then solved locally in each processor. The algorithm is outlined in Fig. 6.7.

All versions of DDFAST corresponding to different fast subdomain solvers mentioned in this chapter and the corresponding versions saving a factor of two in the high order term have parallel implementations analogous to Par-DDFAST (Par-DD1, Par-DD2, etc.) Algorithm DD1 is almost equivalent to a parallel implementation of the MD method, in which the sine transforms are computed locally and the tridiagonal systems are solved by *substructuring*, where Gaussian elimination is applied locally to obtain a reduced system that is distributed among the processors, one equation per processor. For the constant-coefficient case, the reduced system is in fact the transformed interface system (6.14). The elimination phase is simplified in Algorithm DD1, where the coefficients for the reduced system are not computed by Gaussian elimination, but given by the eigenvalues λ_j and δ_j.

We next analyze in detail the complexity and numerical properties of a hypercube implementation of Par-DD2, which is outlined in Fig. 6.8. In order to analyze the complexity of algorithm Par-DD2, we consider it in five parts:

T_{Prepr} : *Preprocessing.* Compute λ_j's and δ_j's, initialize sine transform routines (routine SINTI), etc.
T_{First} : *First solve*, i.e. Step 1.
T_{Interf} : Computation of g, i.e. Step 2, and solution of the interface system, i.e. Step 3.
T_{rhs2} : Exchange interface values and update the sine transform of the right hand side for second solve, i.e. Step 4-a.
T_{second} : *Second solve*, i.e. Steps 4-b and 4-c.

Figure 6.7: **Algorithm Par-DDFAST**

Program for processor P_i.

Step 1: (First Solve) Solve $A_{ii}z_i = f_{\Omega_i}$ locally.

Step 2: Form the right-hand side for the interface system:

a) if $i > 1$, send $P^T_{i,i-1}z_i$ to processor $P(i-1)$,

b) if $i < p$, receive $P^T_{i+1,i}z_{i+1}$ from processor $P(i+1)$

c) compute $g_{\Gamma_i} = f_{\Gamma_i} - P^T_{ii}z_i - P^T_{i+1,i}z_{i+1}$.

Step 3: Solve the interface system (6.3):

compute $\hat{g}_{\Gamma_i} = W^T g_{\Gamma_i}$ locally.

Solve the transformed system (6.14):

transpose distributed right-hand sides,

solve $(n+1)/p$ tridiagonal systems of size $p-1$ locally,

back-transpose the solution.

Finally, compute $u_{\Gamma_i} = W\hat{u}_{\Gamma_i}$ locally using fast sine transforms.

Step 4: (Second Solve)

a) if $i < p$, send u_{Γ_i} to processor $P(i+1)$,

if $i > 1$, receive $u_{\Gamma_{i-1}}$ from $P(i-1)$ and update right-hand side:

$h_{\Omega_i} = f_{\Omega_i} - P_{i,i-1}u_{\Gamma_{i-1}} - P_{ii}u_{\Gamma_i}$.

b) solve $A_{ii}u_{\Omega_i} = h_{\Omega_i}$.

The complexity for the preprocessing is $T_{Prepr} = \mathcal{O}(n)$. The first and second solves are completely parallel. All the communication operations are performed in parts 3 and 4.

The arithmetic complexity for the first solve is:

$$C_{First}(n,m,p) = nC_s(\frac{m+1}{p}) + (C_{t1}+2)n\frac{m+1}{p} + \mathcal{O}(n) + \mathcal{O}(\frac{m+1}{p})$$

In order to form g, each processor sends and receives a vector of length n to and from nearest neighbor processors. The arithmetic complexity for forming g and solving the interface system is:

$$C_{Interf}(n,p) = 2C_s(n+1) + C_t(p-1)\frac{n+1}{p} + 2n \quad .$$

In our hypercube implementation, this routine also requires transpose and back-transpose operations, where $2\log p$ exchanges of vectors of length $\frac{n+1}{2}$ involving nearest neighbor processors are necessary. The rest of the transpose routine does not involve floating-point arithmetic, but it requires some data reshuffling.

After solving the interface system, the interface values are sent to and received from nearest neighbors. The sine transform of the right-hand side for the second solve can be updated with $C_{rhs2}(n,m,p) = 4n\frac{m+1}{p}$ flops.

The arithmetic complexity for the rest of the second solve is

$$C_{second}(n,m,p) = nC_s(\frac{m+1}{p}) + C_{t1}n\frac{m+1}{p} + \mathcal{O}(n) + \mathcal{O}(\frac{m+1}{p}) \quad .$$

In summary, the arithmetic complexity of algorithm Par-DD2 is

$$\begin{aligned} C_{Par-DD2} \quad (n,m,p) &= C_{prepr} + C_{first} + C_{interf} + C_{rhs2} + C_{second} \\ &= 2nC_s(\frac{m+1}{p}) + (2C_{t1}+6)n\frac{m+1}{p} + 2C_s(n+1) + \mathcal{O}(n) + \mathcal{O}(\frac{m+1}{p}) \quad . \end{aligned}$$

Figure 6.8: **Algorithm Par-DD2**

Program for node $P(i)$.

Step 1: (First Solve) Solve $A_{ii}z_i = f_{\Omega_i}$ locally:

a) compute column-wise sine transforms of the right hand side (dimension m_0).

b) solve $\overline{A}_{ii}\hat{z}_i = \hat{f}_{\Omega_i}$ (m_0 tridiagonal systems of dimension n),

c) compute (only) $P_{i,i}^T z_i = (z_{1m_0}^i, \ldots, z_{nm_0}^i)$ (last row) by (6.17) and $P_{i,i-1}^T z_i = (z_{11}^i, \ldots, z_{n1}^i)$ (first row) by (6.18).

Step 2: Form the right-hand side for the interface system:

a) if $i > 1$, send $P_{i,i-1}^T z_i$ to processor $P(i-1)$,

b) if $i < p$, receive $P_{i+1,i}^T z_{i+1}$ from processor $P(i+1)$,

c) compute $g_{\Gamma_i} = f_{\Gamma_i} - P_{ii}^T z_i - P_{i+1,i}^T z_{i+1}$.

Step 3: Solve the interface system (6.3):

compute $\hat{g}_{\Gamma_i} = W^T g_{\Gamma_i}$ locally.

Solve the transformed system (6.14):

transpose distributed right-hand sides,

solve $(n+1)/p$ tridiagonal systems of size $p-1$ locally,

back-transpose the solution.

Finally, compute $u_{\Gamma_i} = W\hat{u}_{\Gamma_i}$ locally using fast sine transforms.

Step 4: (Second solve)

a) If $i < p$, send u_{Γ_i} to $P(i+1)$,

if $i > 1$, receive $u_{\Gamma_{i-1}}$ from $P(i-1)$,

update the sine transforms of f_{Ω_i} by transforming only $-P_{i,i-1}u_{\Gamma_{i-1}} - P_{ii}u_{\Gamma_i}$, which is a sparse vector (direct calculation, i.e. no fast sine transforms).

b) Solve the m_0 tridiagonal systems $\overline{A}_{ii}\hat{u}_i = \hat{h}_{\Omega_i}$ locally.

c) Compute u_{Ω_i} locally by fast sine transforms.

The communication cost is

$$2\log p\, T_{send-recv}(\frac{n+1}{2}) + 2T_{send-recv}(n) \quad ,$$

where $T_{send-recv}(n)$ is the time for sending and receiving vectors of length n to and from a nearest neighbor processor.

6.6 Numerical Experiments

The black-box and the efficient versions of algorithm DD2 were implemented on an Intel Hypercube IPSC/2 multiprocessor system, for the solution of Poisson's equation on the unit square. The number of processors used ranged from one to 64.

Runtime split. In Table 6.3 we show how the total runtime is split among the different steps of the algorithm. T_P represents preprocessing time, i.e. compute λ's and δ's, initialize the sine transform routines,

Table 6.3: Percentage of total time
for each step of Algorithm Par-DD2

T_P = preprocessing T_I = interface system.
T_1 = first solve (Step 1). T_2 = second solve (Step 4).

	p	Total Time (sec)	Perc. of Total Time			
			T_P	T_1	T_I	T_2
$h = 2^{-7}$	4	0.93	0.6	46.6	11.1	41.7
	8	0.49	0.8	44.2	15.8	39.1
	16	0.24	1.3	38.6	27.5	32.6
	32	0.15	2.7	30.4	41.2	25.0
$h = 2^{-8}$	4	4.30	0.3	48.0	7.8	43.9
	8	1.87	0.4	46.5	11.5	41.6
	16	0.97	0.6	44.2	16.0	39.1
	32	0.47	1.1	38.2	28.1	32.5
	64	0.29	1.7	31.0	41.8	25.2
$h = 2^{-9}$	4	17.42	0.1	48.5	6.8	44.5
	8	8.66	0.2	48.0	7.9	43.9
	16	3.74	0.3	46.5	11.6	41.6
	32	1.96	0.5	44.0	16.4	39.0
	64	0.94	1.1	38.5	27.8	32.6
$h = 2^{-10}$	4	78.15	0.1	48.8	5.8	45.4
	8	35.06	0.1	48.4	7.0	44.5
	16	17.46	0.1	47.9	8.2	43.8
	32	7.63	0.3	46.1	12.4	41.2
	64	3.97	0.5	43.6	17.3	38.6

etc. T_1 corresponds to the first solve, i.e. Step 1. T_I is the time for computing the right-hand side g, solving the interface system and exchanging interface values to form the right hand side for the second solve (Steps 2, 3 and 4-a). T_2 is the time for the second solve (Step 4-b and c).

A least squares fit of the times was computed for each part of the algorithm. Since the times for the sine transform do not follow a well defined function of n and m_0, especially when m_0 takes small values, we used the tabulated times of Table 6.2. If, for example, T_{first} is the time taken by any node for the first subdomain solve, we computed a least square fit of the form $a_1 n m_0 + a_2 n + a_3 m_0 + a_4$ for $T_{first} - nC_s(m_0 + 1)$, with $m_0 = \frac{n+1}{p} - 1$. We then have (times given in milliseconds):

$$\begin{aligned} T_{first} &\approx nC_s(m_0+1) + 0.0333nm_0 - 0.0014n - 0.4243m_0 + 4.4235 \\ T_{second} &\approx nC_s(m_0+1) + 0.0232nm_0 - 0.0502n - 0.5362m_0 + 6.3744 \end{aligned}$$

and

$$\begin{aligned} T_{Interf} &\approx 2C_s(n+1) + (2.8689 + 0.0162n)\log p + 0.0248n + 0.9585 \\ T_{rhs2} &\approx 0.0160nm_0 - 0.0029n + 0.0034m_0 + 1.2077 \quad . \end{aligned}$$

Therefore, for a square domain ($m = n$) and excluding preprocessing time, we have

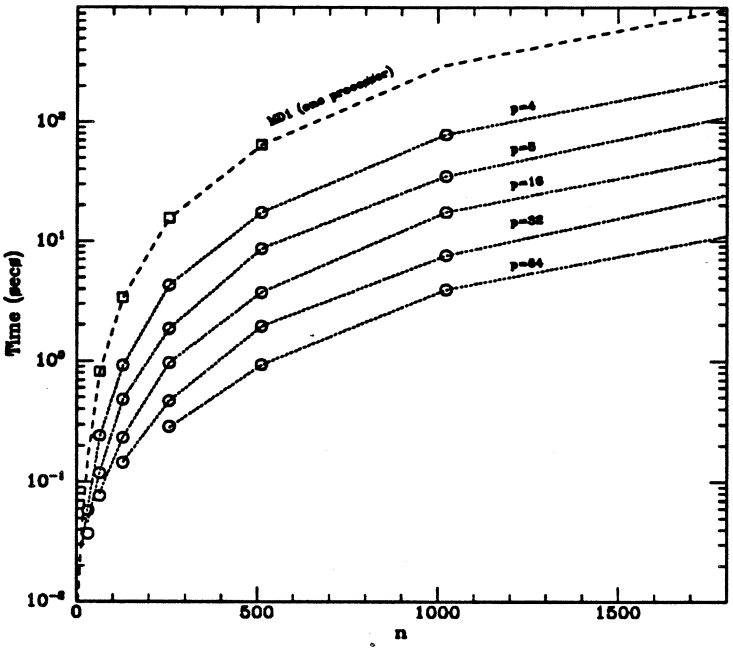

Figure 6.9: Runtime for $MD1$ (one processor times) and $Par-DD2(p)$ for $p = 4$, 8, 16, 32 and 64 vs. n. The dotted lines join the least-squares fit for the given values of n.

$$T_{Par-DD2}(n,p) \approx 2nC_s(\frac{n+1}{p}) + 0.0725\frac{(n+1)^2}{p} + (2.8689 + 0.0162n)\log p$$
$$-0.07257n - 1.0592\frac{(n+1)}{p} + 13.95 \qquad (6.24)$$

In Fig. 6.9 we plot runtime vs. problem size for 4, 8, 16, 32 and 64 processors, and for the standard fast solver MD1 running in one processor. The dotted lines join the least-squares fit (6.24) for the given values of n.

Speed-up

The parallel algorithm was compared against the solver MD1 to compute speed-up.

In analyzing a parallel algorithm, the concept of *speed-up* is loosely defined as the relative savings in run-time achieved by solving a given problem in p processors, compared to the time for solving the same problem in one processor, i.e.

$$S(p) = \frac{T(1)}{T(p)} \quad .$$

This definition is ambiguous, because it does not specify what method is used to solve the problem in one processor. Usually, $T(1)$ is defined as the time taken by *the fastest sequential* algorithm known for solving the particular given problem. Given this definition — which may be specified as *absolute* speed-up — it is clear that $S(p)$ is always bounded by p. When $S(p) = p$, the algorithm is said to have "perfect" or "linear" speed-up.

When $T(1)$ and $T(p)$ refer to the same algorithm, $S(p)$ alone is not a good measure of the efficiency of the algorithm, but it does measure how well it parallelizes. In other words, it shows how the overhead grows when more processors are added. We call this *relative* speed-up.

Table 6.4: Speed-up for Algorithm DD2

Speed-up is computed with respect to sequential fast solver MD1.

Black-box implementation

	$h = 2^{-6}$	$h = 2^{-7}$	$h = 2^{-8}$	$h = 2^{-9}$
$p = 4$	2.22	2.31	2.21	2.22
$p = 8$	4.78	4.49	5.33	4.49
$p = 16$	7.81	9.86	10.32	10.85
$p = 32$		16.29	22.68	20.99
$p = 64$			37.82	46.30

Efficient version of algorithm DD2

	$h = 2^{-6}$	$h = 2^{-7}$	$h = 2^{-8}$	$h = 2^{-9}$
$p = 4$	3.41	3.63	3.62	3.66
$p = 8$	6.85	6.94	8.34	7.35
$p = 16$	10.72	14.29	15.98	17.01
$p = 32$		22.78	32.88	32.54
$p = 64$			53.00	67.35

In the context of domain decomposition, $T(p)$ corresponds to a decomposition of the domain into p subdomains, i.e the p-processor version of $DD(p)$. The choice of the method to be used as the one-processor solver is not obvious, because it is not always known which sequential algorithm is the fastest for a given problem size, due to the many factors involved, such as the machine characteristics and implementation details. As long as a reasonably efficient fast solver is applied to the subdomain problems, it makes sense to compare the resulting domain-decomposed algorithm with the same subdomain solver applied to the whole domain, namely

$$S(p) = \frac{T(DD(1))}{T(\text{Par-DD}(p))} \quad . \tag{6.25}$$

As we showed before, Algorithm DDFAST can have lower complexity than the solver used on the subdomains. For example, in Table 6.1 we showed that for certain values of p Algorithm DD2(p) may be faster than the original fast solver MD1 or MD2 applied to the subdomain problems. It is therefore possible, for this last definition of speed-up, to have super-linear speed-up, or $S(p) > p$.

Table 6.4 shows speed-ups for the black-box version and the more efficient version of DD2 for various problem sizes (h ranging from 2^{-6} to 2^{-9}) and cube dimensions 2 to 6. In both cases, the one-processor solver was MD1, i.e. matrix decomposition using Fourier analysis. In other words, for this table we used the definition of speed-up given by (6.25), except that in our one processor solver the sine transforms were computed in the direction of the natural ordering while in the p-processor solver the sine transforms were computed in the shorter direction i.e. column-wise. The times for MD1 were given in Table 6.1. Efficiency is defined as $E = 100S(p)/p$ and is plotted in Fig. 6.10 for the efficient version of DD2, for three different problem sizes.

We point out that for each value of p, we can view $DD2(p)$ as a different algorithm. Therefore, we cannot expect speed-up to increase monotonically with problem size because an algorithm of different complexity was used on one processor for comparison. We can verify, however, that when the same domain

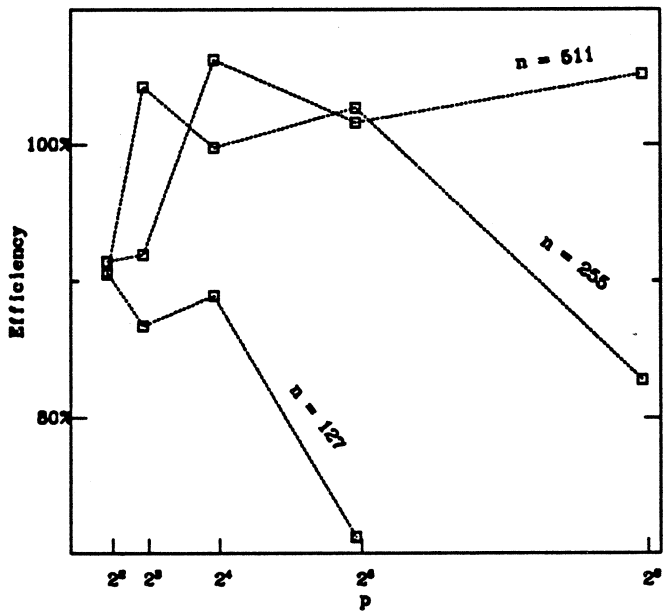

Figure 6.10: Efficiency for Algorithm DD2

decomposition algorithm is considered on one processor, the values of speed-up thus computed (i.e. relative speed-up) do increase monotonically with problem size for a given p.

Relative speed-up. According to the definition, the relative speed-up of a domain-decomposed solver is given by the ratio between the sequential time for DD(p) and the p-processor parallel version of the same algorithm. Relative speed-up and efficiencies are plotted in Fig. 6.11. When the subdomains are not extremely narrow, we can see almost perfect relative speed-up values, with efficiencies approaching 100%. The efficiency decreases as the number of processors p approaches m and the subdomain solvers become less dominating, but it remains high for larger problems (e.g. $E > 82\%$ for a 127 x 127 problem and $E > 86\%$ for a 255 x 255 problem). This indicates that the algorithm is highly parallelizable.

6.7 Three Dimensional Problems

Given a regular three dimensional domain Ω, consider a seven-point discretization of the problem

$$\begin{aligned} Lu &= f \quad \text{in} \quad \Omega \\ u &= u_b \quad \text{on} \quad \partial\Omega \quad , \end{aligned} \tag{6.26}$$

on an n_1 by n_2 by m grid. We divide Ω into k "slices" and assume that $m+1$ is a multiple of k with $\frac{m+1}{k} \geq 4$.

Let the system

$$\begin{pmatrix} A_\Omega & P \\ P^T & A_\Gamma \end{pmatrix} \begin{pmatrix} u_\Omega \\ u_\Gamma \end{pmatrix} = \begin{pmatrix} f_\Omega \\ f_\Gamma \end{pmatrix} \tag{6.27}$$

represent the discretization of (6.26) in Ω, where u_Ω corresponds to values on the interior of the slices and u_Γ corresponds to values on the interfaces and we assume the natural ordering of the gridpoints inside each

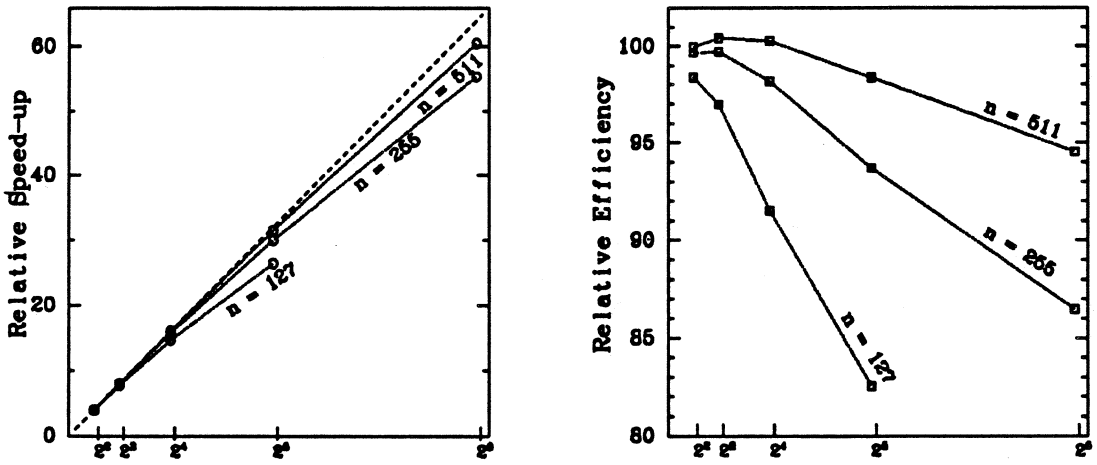

Figure 6.11: Relative speed-up and efficiency for Algorithm DD2

subdomain. When block elimination is applied to (6.27), the problem is reduced to independent problems on the subdomains and the linear system

$$Cu_\Gamma = g$$

for the interface unknowns, where $C = A_\Gamma - P^T A_\Omega^{-1} P$ and $g = f_\Gamma - P^T A_\Omega^{-1} f_\Omega$. The interface operator has the following block-tridiagonal form:

$$C = \begin{pmatrix} H_1 & B_2 & & \\ B_2 & H_2 & \ddots & \\ & \ddots & \ddots & B_{k-1} \\ & & B_{k-1} & H_{k-1} \end{pmatrix} .$$

In the constant-coefficient case, the blocks B_i and H_i are diagonalizable by two-dimensional Fourier modes, i.e. the matrix $W_{n_2} \otimes W_{n_1}$. Let L be given by the operator:

$$Lu \equiv -\frac{\partial}{\partial x}(a\frac{\partial u}{\partial x}) - \frac{\partial}{\partial y}(b\frac{\partial u}{\partial y}) - \frac{\partial}{\partial z}(c\frac{\partial u}{\partial z}) + du$$

where the coefficients a, b, c and d take constant values a_i, b_i, c_i and d_i on each subdomain Ω_i. For this case, the eigenvalues of H_i are given by

$$\lambda_{ij} = c_i \left(\frac{1+\gamma_{ij}^{m_i+1}}{1-\gamma_{ij}^{m_i+1}}\right)\sqrt{\frac{\mu_{ij}^2}{4}+\mu_{ij}} + c_{i+1}\left(\frac{1+\gamma_{i+1,j}^{m_{i+1}+1}}{1-\gamma_{i+1,j}^{m_{i+1}+1}}\right)\sqrt{\frac{\mu_{i+1j}^2}{4}+\mu_{i+1j}}$$

and the eigenvalues of B_i are given by

$$\delta_{ij} = -2c_i \left(\frac{(\sqrt{\gamma_{ij}})^{m_i+1}}{1-\gamma_{ij}^{m_i+1}}\right)\sqrt{\frac{\mu_{ij}^2}{4}+\mu_{ij}} \quad ,$$

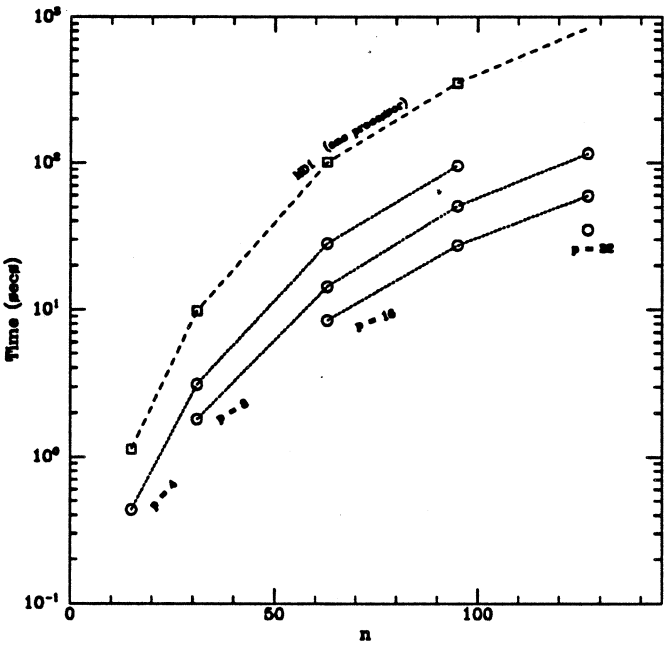

Figure 6.12: Runtime for $MD1$ and ParDD2(p) for the solution of Poisson's equation in three dimensions.

where

$$\gamma(\mu_{ij}) = \left(1 + \frac{\mu_{ij}}{2} - \sqrt{\frac{\mu_{ij}^2}{4} + \mu_{ij}}\right)^2$$

and $\mu_{ij} = \frac{1}{c_i}(a_i\sigma_{j_1} + b_i\sigma_{j_2} + d_i h^2)$, with $j = j_1 + (j_2 - 1)n_1$, for $j_1 = 1, \ldots, n_1$ and $j_2 = 1, \ldots, n_2$.

6.7.1 Parallel Three Dimensional Solver

The implementation of Algorithm DD2 for the three dimensional case is analogous to the two-dimensional case. Since the subdomains are thin three-dimensional slices with short vectors in the z-direction, the subdomain problems are solved by applying fast Fourier sine tranforms in the y and z directions and solving tridiagonal systems in the x direction. The algorithm was implemented on an Intel Hypercube IPSC/2 multiprocessor system, for the solution of Poisson's equation on the unit cube. The number of processors used ranged from one to 32.

Fig. 6.12 shows runtimes for problem sizes ranging from 15 x 15 x 15 to 127 x 127 x 127. The speed-up values in Table 6.5 were computed with respect to the one-processor time given by the matrix decomposition algorithm, with fast sine tranforms applied in the x and y directions and tridiagonal systems in the z direction. The corresponding efficiency values are also given in the same table. The largest problem solved in one processor was on a 95 x 95 x 95 grid. For $h = 2^{-6}$, we can see that these values are similar to the speed-ups reported in Table 6.4 for the two-dimensional case for the same mesh spacing (efficient version of DD2).

6.8 Discussion and Conclusions

We showed that domain decomposition techniques can be used successfully in the implementation of parallel fast Poisson solvers. They provide fast solvers that are not only naturally parallelizable, but also faster than conventional fast solvers even in the sequential case.

Table 6.5: Speed-up and efficiency for three-dimensional Poisson solver

Speed-up is computed with respect to sequential fast solver MD1.

	$h = 2^{-4}$		$h = 2^{-5}$		$h = 2^{-6}$		$h = 2^{-5}3^{-1}$	
	S	E	S	E	S	E	S	E
$p = 4$	2.60	65%	3.15	78.7%	3.57	89.2%	3.70	92.5%
$p = 8$			5.41	67.6%	7.02	87.7%	6.97	87.1%
$p = 16$					11.93	74.6%	12.91	80.7%

As the number of processors increases and the strips or slices become narrow, the solution of the interface system dominates. In the 2-D case, for instance, instead of subdividing into more strips, we can consider subdividing each strip into boxes and replace the subdomain solvers by a multistrip DD solver inside each strip. One problem with this approach is that there is no obvious way to save the factor of two due to the two solvers. A better alternative is to assign one subdomain to more than one processor. For example, for a DD2 solver, we can consider a decomposition into p/q strips and divide the work for each subdomain solve among q processors, for example the short fast sine transforms can be performed in parallel and the tridiagonal systems can be solved by transposing the right-hand sides and solving m_0/q systems locally.

Chapter 7

Parallel Preconditioners for Non-Separable Problems

7.1 Introduction

Consider the Dirichlet problem

$$\begin{aligned} Lu &= f \quad \text{in} \quad \Omega = (0,1) \times (0,\beta) \\ u &= u_b \quad \text{on} \quad \partial\Omega \end{aligned}$$

where L is the linear self-adjoint elliptic differential operator:

$$L \equiv -\frac{\partial}{\partial x}\left(a(x,y)\frac{\partial}{\partial x}\right) - \frac{\partial}{\partial y}\left(b(x,y)\frac{\partial}{\partial y}\right) + c(x,y) \quad . \tag{7.1}$$

Assume that the coefficients satisfy $c(x,y) \geq 0$ and $a(x,y), b(x,y) \geq \delta$ for all $(x,y) \in \Omega$ and some positive constant δ. The five-point finite difference approximation on a regular $n \times m$ grid of mesh size $h = \frac{1}{n+1}$, where $h(m+1) = \beta$, leads to a linear system of equations of the form

$$Au = f \quad , \tag{7.2}$$

where the matrix A is block tridiagonal.

If the coefficients are such that $a(x,y) = a(x)$, $b(x,y) = b(y)$ and $c(x,y) = c_1(x) + c_2(y)$, then L is separable and fast direct solvers such as block cyclic reduction or matrix decomposition with Fourier analysis [10, 27, 32] can be applied.

In the non-separable case, the discretized equations are often solved by iterative methods such as preconditioned conjugate gradients. Standard preconditioners include incomplete factorizations such as the Dupont, Kendall and Rachford (DKR) factorization [23].

Separable approximations of L — which can be solved by fast direct methods and are *spectrally equivalent* to L — can be used as preconditioners. Concus and Golub [17] use constant coefficient approximations in conjunction with Chebyshev acceleration. Bank [3] combines the generalized marching method with the D'Yakanov-Gunn iteration. Elman and Schultz [25] investigate extensions to the non-selfadjoint case.

Let the operator L_S be defined as

$$L_S \equiv -\frac{\partial}{\partial x}\left(\tilde{a}(x)\frac{\partial}{\partial x}\right) - \frac{\partial}{\partial y}\left(\tilde{b}(y)\frac{\partial}{\partial y}\right) + (\tilde{c}_1(x) + \tilde{c}_2(y)) \quad , \tag{7.3}$$

where the coefficients $\tilde{a}(x)$, $\tilde{b}(y)$, $\tilde{c}_1(x)$ and $\tilde{c}_2(y)$ are chosen so that

$$\min_{y\in[0,\beta]} a(x,y) \quad \leq \tilde{a}(x) \quad \leq \max_{y\in[0,\beta]} a(x,y)$$

$$\begin{aligned}
\min_{x\in[0,1]} b(x,y) &\le \tilde{b}(y) &\le \max_{x\in[0,1]} b(x,y) \\
\min_{y\in[0,\beta]} c^{(1)}(x,y) &\le \tilde{c}_1(x) &\le \max_{y\in[0,\beta]} c^{(1)}(x,y) \\
\min_{x\in[0,1]} c^{(2)}(x,y) &\le \tilde{c}_2(y) &\le \max_{x\in[0,1]} c^{(2)}(x,y) \quad ,
\end{aligned}$$

with $c^{(1)}, c^{(2)} \ge 0$ such that $c^{(1)}(x,y) + c^{(2)}(x,y) = c(x,y)$. Let M represent a five point discretization of L_S. Then, it can be shown [3] that A and M are spectrally equivalent, i.e., there exist positive constants μ_1 and μ_2 such that, for all $v \in R^{mn}, v \neq 0$,

$$\mu_1 \le \frac{v^T A v}{v^T M v} \le \mu_2 \quad .$$

The constants μ_1 and μ_2 are independent of the mesh size h and depend on how closely the separable coefficients in L_S approximate the non-separable coefficients in L. The bound (2.2) on the rate of convergence of PCG with preconditioner M depends on the spectral condition number of $M^{-1}A$, which is bounded by $\frac{\mu_2}{\mu_1}$.

Since the rate of convergence is independent of the problem size, the number of iterations required for solving a non-separable elliptic PDE to an accuracy of $\mathcal{O}(h^2)$ with a spectrally equivalent preconditioner is $\mathcal{O}(\log n)$. Thus, the asymptotic complexity is, for example, $\mathcal{O}(n^2 \log^2 n)$ for an $n \times n$ grid on a square domain. The constant in front of the leading term depends on the coefficients in (7.1) and on the choice of the separable approximation, as well as on the fast method used to apply the preconditioner.

On the other hand, when the DKR preconditioner is used on an $n \times n$ grid, the number of PCG iterations increases with $\sqrt{n} \log n$, giving an overall complexity of $\mathcal{O}(n^{2.5} \log n)$ [11]. Asymptotically, separable spectrally equivalent preconditioners are faster than incomplete factorizations. The cross over value of n depends on the particular problem and it might be impractically large for problems that cannot be well approximated by a separable operator.

7.2 Piece-wise Separable Approximations and Domain Decomposition

Consider a partition of the domain Ω into k subregions Ω_i. We will define a domain decomposed spectrally equivalent preconditioner by the five point discretization of an operator of the form L_S, where the coefficients $\tilde{a}$, $\tilde{b}$ and $\tilde{c}$ are defined by different locally separable approximations in each subregion. Let

$$\underline{a}_i(x) = \inf_{y:(x,y)\in\Omega_i} a(x,y) \quad , \quad \overline{a}_i(x) = \sup_{y:(x,y)\in\Omega_i} a(x,y)$$

$$\underline{b}_i(y) = \inf_{x:(x,y)\in\Omega_i} b(x,y) \quad , \quad \overline{b}_i(y) = \sup_{x:(x,y)\in\Omega_i} b(x,y)$$

and

$$\underline{c}_i^{(1)}(x) = \inf_{y:(x,y)\in\Omega_i} c^{(1)}(x,y) \quad , \quad \overline{c}_i^{(1)}(x) = \sup_{y:(x,y)\in\Omega_i} c^{(1)}(x,y)$$

$$\underline{c}_i^{(2)}(y) = \inf_{x:(x,y)\in\Omega_i} c^{(2)}(x,y) \quad , \quad \overline{c}_i^{(2)}(y) = \sup_{x:(x,y)\in\Omega_i} c^{(2)}(x,y) \quad .$$

Define the operator L_{DD} as

$$L_{DD} \equiv -\frac{\partial}{\partial x}\left(\tilde{a}\frac{\partial}{\partial x}\right) - \frac{\partial}{\partial y}\left(\tilde{b}\frac{\partial}{\partial y}\right) + \tilde{c} \tag{7.4}$$

such that $\tilde{a}(x,y) \equiv \tilde{a}_i(x)$, $\tilde{b}(x,y) \equiv \tilde{b}_i(y)$ and $\tilde{c}(x,y) \equiv \tilde{c}_i^{(1)}(x) + \tilde{c}_i^{(2)}(y)$ for $(x,y) \in \Omega_i$ (i.e., the coefficients are separable inside each subregion), with

$$\underline{a}_i(x) \leq \tilde{a}_i(x) \leq \overline{a}_i(x) \quad , \quad \underline{b}_i(y) \leq \tilde{b}_i(y) \leq \overline{b}_i(y) \quad , \tag{7.5}$$

$$\underline{c}_i^{(1)}(x) \leq \tilde{c}_i^{(1)}(x) \leq \overline{c}_i^{(1)}(x) \quad \text{and} \quad \underline{c}_i^{(2)}(y) \leq \tilde{c}_i^{(2)}(y) \leq \overline{c}_i^{(2)}(y) \quad . \tag{7.6}$$

Let the matrix M_{DD} be defined as the five-point discretization of the operator L_{DD} on an $n \times m$ grid (see page 9). Although the coefficients $\tilde{a}, \tilde{b}, \tilde{c}^{(1)}$ and $\tilde{c}^{(2)}$ are not defined at the interior boundaries, we define extensions of the form (2.10-12), i.e., if (x,y) is a point on a horizontal interface between subdomains Ω_i and Ω_j, for example, then $\tilde{a}(x,y)$ is redefined as $\tilde{a}(x,y) = \frac{1}{2}(\tilde{a}_i(x) + \tilde{a}_j(x))$.

The proof of spectral equivalence in [3] could be easily applied to this case, where the coefficients are not necessarily separable but piece-wise separable. However, their upper and lower bounds for $(v^T A v)/(v^T M_{DD} v)$ are not general, because they depend on the ratios $\frac{\underline{c}^{(1)}}{\tilde{c}^{(1)}}$ and $\frac{\underline{c}^{(2)}}{\tilde{c}^{(2)}}$ and they do not apply to the case when either $\tilde{c}^{(1)}$ or $\tilde{c}^{(2)}$ take the value zero. The bounds derived here are independent of h and the subdomain decomposition and they hold even when $\tilde{c} = 0$. We also need the bounds to express the fact that, as the partition is refined, the condition number may approach one.

In what follows, given grid points (x_i, y_j), where $x_i = ih$ and $y_j = jh$, we will use the notation $\tilde{a}_{i+\frac{1}{2},j}$ to denote $\tilde{a}(x_i + \frac{h}{2}, y_j)$. The elements of a vector of grid values $v \in R^{mn}$ are denoted by v_{ij}. As in [3], by applying the summation by parts identity:

$$\sum_{i=1}^{n} u_i \left(-\alpha_{i-\frac{1}{2}} u_{i-1} + (\alpha_{i-\frac{1}{2}} + \alpha_{i+\frac{1}{2}}) x_i - \alpha_{i+\frac{1}{2}} u_{i+1} \right) = \sum_{i=0}^{n} \alpha_{i+\frac{1}{2}} (u_{i+1} - u_i)^2$$

with $x_0 = x_{n+1} = 0$, we can write

$$v^T A v = \sum_{\substack{0 \leq i \leq n \\ 0 \leq j \leq m}} a_{i+\frac{1}{2},j} \, (v_{i+1,j} - v_{ij})^2 + b_{i,j+\frac{1}{2}} \, (v_{i,j+1} - v_{ij})^2 + h^2 c_{ij} v_{ij}^2$$

with $v_{i0} = 0$ for all i and $v_{0j} = 0$ for all j and

$$v^T M_{DD} v = \sum_{\substack{0 \leq i \leq n \\ 0 \leq j \leq m}} \tilde{a}_{i+\frac{1}{2},j} \, (v_{i+1,j} - v_{ij})^2 + \tilde{b}_{i,j+\frac{1}{2}} \, (v_{i,j+1} - v_{ij})^2 + h^2 \tilde{c}_{ij} \, v_{ij}^2$$

where $\tilde{a}$, $\tilde{b}$ and $\tilde{c}$ are defined by average values at discontinuities.

For simplicity, assume that the domain Ω is square and $m = n$.

Lemma 7.1 *For all $v \neq 0, v \in R^{n^2}$ and all n,*

$$v^T M_{DD} v \geq h^2 \sum_{i,j=1}^{n} (8\delta + \tilde{c}_{ij}) \, v_{ij}^2 \quad .$$

Proof: Since $a(x,y), b(x,y) \geq \delta$, we also have $\tilde{a}(x,y), \tilde{b}(x,y) \geq \delta$. Then,

$$\frac{1}{h^2} v^T M_{DD} v \geq \frac{1}{h^2} \delta \sum_{i,j=1}^{n} (v_{i+1j} - v_{ij})^2 + (v_{ij+1} - v_{ij})^2 + \sum_{i,j=1}^{n} \tilde{c}_{ij} \, v_{ij}^2$$

Let L represent the discrete Laplacian operator on an $n \times n$ grid. Then

$$v^T L v = \sum_{i,j=1}^{n} (v_{i+1j} - v_{ij})^2 + (v_{ij+1} - v_{ij})^2 \geq \lambda_{\min} \, v^T v \quad ,$$

where $\lambda_{\min}$ is the smallest eigenvalue of L and it is given by $\lambda_{\min} = 8\sin^2\frac{\pi}{2}h$. Since $\frac{\lambda_{\min}}{h^2} \geq 8$ for all $h \geq 1$, then we have:

$$\frac{1}{h^2} v^T M_{DD} v \geq 8\delta v^T v + \sum_{i,j=1}^{n} \tilde{c}_{ij}\ v_{ij}^2 = \sum_{i,j=1}^{n} (8\delta + \tilde{c}_{ij})\ v_{ij}^2$$

■

Lemma 7.2 *For all $v \neq 0, v \in R^{n^2}$ and all n,*

$$-(1-\underline{\nu}_c) \leq \frac{h^2 \sum_{i,j=1}^{n} (c_{ij} - \tilde{c}_{ij})\ v_{ij}^2}{v^T M_{DD} v} \leq \overline{\nu}_c - 1 \quad , \tag{7.7}$$

where

$$\underline{\nu}_c = \inf_{\Omega} \left\{ \frac{8\delta + c(x,y)}{8\delta + \tilde{c}(x,y)} \right\} \quad \textit{and} \quad \overline{\nu}_c = \sup_{\Omega} \left\{ \frac{8\delta + c(x,y)}{8\delta + \tilde{c}(x,y)} \right\} \quad .$$

Proof: Since

$$-1 + \underline{\nu}_c = \inf_{\Omega} \left\{ \frac{c(x,y) - \tilde{c}(x,y)}{8\delta + \tilde{c}(x,y)} \right\} \qquad \text{and} \qquad \overline{\nu}_c - 1 = \sup_{\Omega} \left\{ \frac{c(x,y) - \tilde{c}(x,y)}{8\delta + \tilde{c}(x,y)} \right\} \quad ,$$

we have

$$-1 + \underline{\nu}_c \leq \frac{\sum_{i,j=1}^{n} (c_{ij} - \tilde{c}_{ij})\ v_{ij}^2}{\sum_{i,j=1}^{n} (8\delta + \tilde{c}_{ij})\ v_{ij}^2} \leq \overline{\nu}_c - 1 \quad .$$

Then, by applying Lemma 7.1, and since $-1 + \underline{\nu}_c \leq 0$, we have (7.7). ■

Define

$$\underline{\nu}_{ab} = \inf_{\Omega} \left\{ \frac{a(x,y)}{\tilde{a}(x,y)}, \frac{b(x,y)}{\tilde{b}(x,y)} \right\} \quad \text{and} \quad \overline{\nu}_{ab} = \sup_{\Omega} \left\{ \frac{a(x,y)}{\tilde{a}(x,y)}, \frac{b(x,y)}{\tilde{b}(x,y)} \right\} \quad .$$

Theorem 7.1

a) *M_{DD} is spectrally equivalent to A: for all $v \neq 0, v \in R^{n^2}$,*

$$\mu_1 \leq \frac{v^T A v}{v^T M_{DD} v} \leq \mu_2 \quad . \tag{7.8}$$

where $\mu_1 = \underline{\nu}_{ab} \cdot \underline{\nu}_c$ and $\mu_2 = \overline{\nu}_{ab} \cdot \overline{\nu}_c$.

The constants μ_1 and μ_2 depend on the particular decomposition of the domain, but they are independent of h, as long as the coefficients of L_{DD} are defined independently of h.

b) *μ_1 and μ_2 are bounded independently of the domain decomposition and the number of subdomains k. Moreover, μ_1 and μ_2 approach one as the subdomains become smaller.*

Proof:

a) Given $v \neq 0$, we have

$$v^T A v \ \geq \ \underline{\nu}_{ab}\ v^T M_{DD} v + \sum_{i,j=1}^{n} h^2 (c_{ij} - \underline{\nu}_{ab} \tilde{c}_{ij})\, v_{ij}^2$$

$$v^T A v \ \leq \ \overline{\nu}_{ab}\ v^T M_{DD} v + \sum_{i,j=1}^{n} h^2 (c_{ij} - \overline{\nu}_{ab} \tilde{c}_{ij})\, v_{ij}^2 \quad .$$

Since $\underline{\nu}_{ab} \leq 1$ and $\overline{\nu}_{ab} \geq 1$, we have

$$v^T A v \geq \underline{\nu}_{ab} \left(v^T M_{DD} v + \sum_{i,j=1}^{n} h^2 (c_{ij} - \tilde{c}_{ij}) v_{ij}^2 \right)$$

$$v^T A v \leq \overline{\nu}_{ab} \left(v^T M_{DD} v + \sum_{i,j=1}^{n} h^2 (c_{ij} - \tilde{c}_{ij}) v_{ij}^2 \right) \quad .$$

Then, by Lemma 7.2,

$$\mu_1 = \underline{\nu}_{ab}\, \underline{\nu}_c \leq \frac{v^T A v}{v^T M_{DD} v} \leq \overline{\nu}_{ab}\, \overline{\nu}_c = \mu_2 \quad .$$

b) Define $\rho_{ab} = \max\limits_{i=1,\ldots,k} \{\rho_{a_i}, \rho_{b_i}\}$, where

$$\rho_{a_i} = \sup_{x:(x,y)\in\Omega_i} \frac{\overline{a}_i(x)}{\underline{a}_i(x)} \quad \text{and} \quad \rho_{b_i} = \sup_{y:(x,y)\in\Omega_i} \frac{\overline{b}_i(y)}{\underline{b}_i(y)}$$

and also $\rho_c = \max\limits_{i=1,\ldots,k} \{\rho_{c_i^{(2)}}, \rho_{c_i^{(2)}}\}$, for

$$\rho_{c_i^{(1)}} = \sup_{x:(x,y)\in\Omega_i} \frac{4\delta + \overline{c}_i^{(1)}(x)}{4\delta + \underline{c}_i^{(1)}(x)} \quad \text{and} \quad \rho_{c_i^{(2)}} = \sup_{y:(x,y)\in\Omega_i} \frac{4\delta + \overline{c}_i^{(2)}(y)}{4\delta + \underline{c}_i^{(2)}(y)} \quad .$$

By (7.5) and (7.6), we have:

$$\frac{1}{\rho_{ab}}\, \tilde{a}(x,y) \leq a(x,y) \leq \rho_{ab}\, \tilde{a}(x,y) \quad , \tag{7.9}$$

$$\frac{1}{\rho_{ab}}\, \tilde{b}(x,y) \leq b(x,y) \leq \rho_{ab}\, \tilde{b}(x,y) \tag{7.10}$$

and also

$$\frac{1}{\rho_c}\, (8\delta + \tilde{c}(x,y)) \leq 8\delta + c(x,y) \leq \rho_c (8\delta + \tilde{c}(x,y)) \tag{7.11}$$

for all $(x,y) \in \Omega_i$. We must prove that these inequations also hold for points on the interior boundaries. Suppose, for example, that at a point (x,y) on a horizontal interface between subdomains Ω_i and Ω_j, $\tilde{a}(x,y)$ is redefined as $\tilde{a}(x,y) = \frac{1}{2}(\tilde{a}_i(x) + \tilde{a}_j(x))$ and, without loss of generality, suppose also that $\tilde{a}_i(x) \geq \tilde{a}_j(x)$. Thus

$$\underline{a}_j(x) \leq \tilde{a}_j(x) \leq \tilde{a}(x,y) \leq \tilde{a}_i(x) \leq \overline{a}_i(x) \quad .$$

Also, since $\underline{a}_i(x) \leq a(x,y) \leq \overline{a}_j(x)$, we have

$$\frac{1}{\rho_{ab}} \tilde{a}(x,y) \leq \frac{1}{\rho_{a_i}} \tilde{a}(x,y) \leq a(x,y) \leq \rho_{a_j} \tilde{a}(x,y) \leq \rho_{ab} \tilde{a}(x,y) \quad .$$

Similarly, we can prove that (7.10) and (7.11) also hold for points on the interior boundaries. Therefore, $\underline{\nu}_{ab} \geq \dfrac{1}{\rho_{ab}}$ and $\overline{\nu}_{ab} \leq \rho_{ab}$ and also, $\underline{\nu}_c \geq \dfrac{1}{\rho_c}$ and $\overline{\nu}_c \leq \rho_c$.

The constants ρ_{ab} and ρ_c depend on the partition of Ω. They reach their maximum value when the partition contains only one subdomain, namely Ω. We denote these maximum values by $\rho_{ab}(\Omega)$ and $\rho_c(\Omega)$ and we have

$$\mu_1 \geq \frac{1}{\rho_{ab}(\Omega).\rho_c(\Omega)} \quad \text{and} \quad \mu_2 \leq \rho_{ab}(\Omega).\rho_c(\Omega) \tag{7.12}$$

for all partitions and all values of k.

Conversely, as the partition is refined and the subdomains become smaller, ρ_{a_i}, ρ_{b_i}, $\rho_{c_i^{(1)}}$ and $\rho_{c_i^{(2)}}$ tend to one [1]. Thus, ρ_{ab} and ρ_c decrease, approaching the value one as the partition is refined. Therefore, μ_1 and μ_2 also approach one as the partition is refined. ∎

By (7.8) and (7.12), the spectral condition number of $M_{DD}^{-1}A$ is bounded by ρ^2, where $\rho = \rho_{ab}(\Omega).\,\rho_c(\Omega)$. This bound represents the worst case, but for particular ways of choosing the approximate coefficients, we can prove better bounds. Suppose, for example, that the separable coefficients are defined by:

$$\tilde{a}_i(x) = \frac{1}{2}\left(\underline{a}_i(x) + \overline{a}_i(x)\right) \quad , \quad \tilde{b}_i(y) = \frac{1}{2}\left(\underline{b}_i(y) + \overline{b}_i(y)\right)$$
$$\tilde{c}_i^{(1)}(x) = \frac{1}{2}\left(\underline{c}_i^{(1)}(x) + \overline{c}_i^{(1)}(x)\right) \quad , \quad \tilde{c}_i^{(2)}(y) = \frac{1}{2}\left(\underline{c}_i^{(2)}(y) + \overline{c}_i^{(2)}(y)\right) \quad ,$$

then

$$\underline{\nu}_{ab} \geq \frac{2}{1+\rho_{ab}} \quad , \quad \overline{\nu}_{ab} \leq \frac{2}{1+\rho_{ab}^{-1}} \quad ,$$
$$\underline{\nu}_c \geq \frac{2}{1+\rho_c} \quad \text{and} \quad \overline{\nu}_c \leq \frac{2}{1+\rho_c^{-1}} \quad . \tag{7.13}$$

Therefore, the spectral condition number of $M_{DD}^{-1}A$ is bounded by ρ instead of ρ^2. When the minimums and maximums are taken only over grid values, the bounds depend slightly on the meshsize, but it is easy to show that they are always bounded by the continuous case.

Theorem 7.1 gives us the flexibility of defining different approximations on each subdomain. The advantage of this approach is that L can be more closely approximated on the smaller regions, without losing the efficiency of fast solvers for the subdomain problems. The disadvantage is that, in general, fast solvers do not apply to piece-wise separable coefficient problems, and only for some special cases can the solution at the interfaces be computed exactly.

7.3 Parallel Solution of Non-Separable Problems

As in the sequential case, when we consider the parallel solution of non-separable problems by PCG, it is important to analyze the trade-off between convergence rate and cost per iteration. But a new issue appears in the parallel context— some preconditioners are better suited for parallel implementation than others. Incomplete factorizations, for example, are known to perform poorly on distributed memory systems, due to the sequential nature of the forward and back-substitution phases and the communication cost (see, e.g. [16]). The overhead increases as more processors are added and is multiplied by the number of iterations.

A common approach to this problem is to look for preconditioners which minimize or reduce overhead (at the possible cost of loweringing the convergence rate). Extreme examples of this approach are no preconditioning and block diagonal preconditioning. Also, red-black and multicolor orderings are designed to reduce the amount of overhead. Another alternative is preconditioners with reasonably low overhead that keep the iteration count small. In this class we find preconditioners which are spectrally equivalent or nearly spectrally equivalent to A and can be efficiently solved in parallel by domain decomposition techniques.

In Chapter 6 we showed that the Poisson equation on a rectangle can be efficiently solved in parallel by subdividing the domain into strips. Similarly, piece-wise constant coefficient problems can also be efficiently solved by using the results in Chapter 4. Given the problem (7.1) on a rectangular domain, a

[1] This result is true only if we assume that the coefficients a, b and c are continuous or they have jump discontinuities which can be eventually confined to subdomain interfaces, i.e, it is posible to subdivide Ω in such a way that, inside each subdomain, the coefficients are continuous.

piece-wise constant operator can be defined which is spectrally equivalent to L. The communication cost of one application of this preconditioner is comparable to that of the conjugate gradient part of the algorithm. Since the ratio of communication to computation is reasonably small, the cross over value of n for which a spectrally equivalent domain decomposed preconditioner will perform better than incomplete factorizations in parallel, should be considerably smaller than the cross over value for sequential implementations.

7.4 A Brief History of Preconditioners for Multi-domain Decomposition

Keyes and Gropp [28] classify domain decomposition preconditioners in two groups. In the first group, the so called Schur complement methods (SCM), block elimination is applied to the partitioned system

$$\begin{pmatrix} A_\Omega & P \\ P^T & A_\Gamma \end{pmatrix} \begin{pmatrix} u_\Omega \\ u_\Gamma \end{pmatrix} = \begin{pmatrix} f_\Omega \\ f_\Gamma \end{pmatrix} \quad . \tag{7.14}$$

The subdomain problems — i.e. systems with matrix A_Ω — are solved exactly and the interface system

$$Cu_\Gamma = g \quad , \tag{7.15}$$

where C is the Schur complement $C = A_\Gamma - P^T A_\Omega^{-1} P$, is solved iteratively by PCG. A preconditioner M is given by an easily invertible approximation of C.

In the second group, the so called partitioned matrix methods (PMM), (7.14) is solved by PCG. Based on a block LU-decomposition of A:

$$A = \begin{pmatrix} I & 0 \\ P^T A_\Omega^{-1} & I \end{pmatrix} \begin{pmatrix} A_\Omega & 0 \\ 0 & C \end{pmatrix} \begin{pmatrix} I & A_\Omega^{-1} P \\ 0 & I \end{pmatrix} \tag{7.16}$$

and an approximation M of the Schur complement C (M can be any member of the first group), a block preconditioner for A is defined by:

$$\tilde{M} = \begin{pmatrix} I & 0 \\ \tilde{P}^T \tilde{A}_\Omega^{-1} & I \end{pmatrix} \begin{pmatrix} \tilde{A}_\Omega & 0 \\ 0 & M \end{pmatrix} \begin{pmatrix} I & \tilde{A}_\Omega^{-1} \tilde{P} \\ 0 & I \end{pmatrix} \quad . \tag{7.17}$$

For example, $\tilde{A}_\Omega$ and $\tilde{P}$ may correspond to a discrete separable approximation of the operator (7.1). These block-preconditioners are used when the subdomain problems cannot be solved by fast direct methods. At each iteration, a system of the form $\tilde{M}v = u$ can be solved efficiently by block elimination, assuming that efficient solvers exist for $\tilde{A}_\Omega$ (subdomains) and for M (interfaces).

When $\tilde{A}_\Omega = A_\Omega$ and $\tilde{P} = P$, the methods SCM and PMM are equivalent, as shown by the following theorem, which is proved in [28].

Theorem 7.2 *If $\tilde{A}_\Omega = A_\Omega$ and $\tilde{P} = P$, then algorithm PCG applied to (7.15) with preconditioner M and initial guess y_0 is equivalent to algorithm PCG applied to (7.14) with preconditioner $\tilde{M}$ and initial guess*

$$u_0 = \begin{pmatrix} A_\Omega^{-1}(f_\Omega - P y_0) \\ y_0 \end{pmatrix} \tag{7.18}$$

in the sense that, at all iterations,

$$u_i = \begin{pmatrix} A_\Omega^{-1}(f_\Omega - P y_i) \\ y_i \end{pmatrix}$$

and u_i minimizes the A-norm of the error for all iterates generated from initial guesses of the form (x_0, y_0).

This theorem indicates that, without loss of generality, we can consider only preconditioners for A — i.e., PMM — SCM being just a particular case for which the subdomain problems are solved exactly.

7.4.1 Block-Diagonal Preconditioners for Strip-wise Decompositions

The interface matrix C for a multistrip decomposition of a rectangle was described in Chapter 4 as a block tridiagonal matrix of the form

$$C = \begin{pmatrix} H_1 & B_2 & & \\ B_2 & H_2 & \ddots & \\ & \ddots & \ddots & B_{k-1} \\ & & B_{k-1} & H_{k-1} \end{pmatrix} , \tag{7.19}$$

Dryja and Proskurowski [20, 21] proposed a block diagonal preconditioner M_{DP} for C. The diagonal blocks correspond to approximations of the blocks H_i. In [20], they use $\sqrt{K}$ (see (3.4)) and $\sqrt{K^2+4K}$ (see (3.5)) on the diagonal. In [21], they use alternating Neumann-Dirichlet conditions. The idea can be extended to any other approximation of H_i on the diagonal. The bound on the condition number of the preconditioned interface system grows like k^2.

The preconditioner deteriorates as k increases and the strips become narrow, an undesirable property in the context of parallel computation because the number of iterations increases as more processors are added. There are two intuitive reasons for this limitation: first, when the strips become narrow, their aspect ratios decrease, making approximations such as $\sqrt{K}$ and $\sqrt{K^2+4K}$ less accurate, as discussed in Chapter 3. Second and more important, the blocks B_i, which are ignored in the construction of M_{DP}, represent the coupling between interfaces. This coupling will obviously become stronger as the interfaces get closer to each other.

One way of overcoming these limitations is by the construction of preconditioners that take into account the aspect ratios of the subdomains and the coupling between interfaces.

Another way is by decomposing the domain in such a way that the aspect ratios are kept small and the coupling is considered in a coarser level.

7.4.2 Preconditioners for Decompositions with Cross-Points

Bramble et al. [8] addressed both the aspect ratio and the interface coupling problems by allowing the interfaces to cross at the interior of the domain, thus allowing the subdomains to have good aspect ratios and only considering the coupling at the cross-points, where a coarse-grid approximation of the original problem is solved. With their preconditioner M_{BPS}, the condition number of the preconditioned system is bounded by $\mathcal{O}(1+\log\left(\frac{1}{h\sqrt{k}}\right)^2)$, where k is the number of subdomains (boxes).

We briefly describe their method as follows. A system of the form $M_{BPS}u = g$ is solved by first decomposing u into $u_P + u_H$. The component u_P satisfies the PDE on each subdomain with zero boundary conditions on the interfaces and it can be computed by solving independent Dirichlet problems on the subdomains. The component u_H, called the discrete harmonic component, equals u on the interfaces (edges and cross-points) and it satisfies the PDE with zero right-hand side. This component is further decomposed into $u_H = u_E + u_V$, where u_V is linear along the edges and agrees with u at the cross points, and u_E vanishes at the vertices (or cross-points).

In order to compute u_H on the interfaces, a small sparse system is solved first for u_V at the cross-points, by using a finite element basis defined for the coarse grid formed by these points. Then, in order to find u_E on each edge, independent systems with matrices of the form $\sqrt{K}$ (cf. Dryja's preconditioner on Chapter 3) must be solved.

Once u_H is known on the interfaces, it can be extended to the interior of the subdomains by solving another set of independent Dirichlet problems.

For a fixed k, this preconditioner is not spectrally equivalent to the operator A, because the condition number grows as h tends to zero, although the growth is very slow. The convergence rate improves as more subdomains are added. This is a desirable property, although it is not surprising, since the system for the cross-points approaches the original problem $Au = f$.

Dryja et al. [22] proposed a preconditioner that solves alternate Dirichlet and Neumann problems on the subdomains in a checker-board fashion. Its convergence properties are similar to M_{BPS}'s.

These preconditioners have the advantage that they can be applied to a range of domain shapes and operators and they are nearly optimal in the sense that the convergence rate is practically independent of the mesh size. If we, however, restrict ourselves to separable problems on rectangular domains, they cannot compete with fast direct solvers, because they do not compute the exact solution to the original problem at the crosspoints or at the interfaces. For example, seven iterations are necessary to reduce the initial residual by a factor of 10^{-4}, when the preconditioner M_{BPS} is used on the Poisson equation on a rectangle divided into boxes. Since at each iteration, two Poisson problems must be solved on each subdomain, this is clearly not the most efficient way to solve Poisson's equation, neither sequentially nor in parallel. This indicates that, whenever a fast direct method exists to find the exact solution at the interfaces, it should be preferred over an iterative method with a preconditioner which only approximates the solution at the interfaces. But no fast method is known for computing the solution at the interfaces when they include cross-points.

7.5 Strips Revisited

Instead of improving the aspect ratio of the subdomains, we propose to take them into account, as well as the coupling between interfaces. We again consider multistrip decompositions of Ω.

As we mentioned before, the block-diagonal preconditioners proposed for strip-wise decompositions of rectangular domains are not suited for parallel applications, because they ignore the issues of aspect ratios and interface coupling, thus obtaining convergence rates which deteriorate as the strips become narrow. In Chapter 4 we have shown, however, that for constant coefficient and piece-wise constant coefficient problems, the coupling can be treated exactly.

Let the operator $\tilde{L}_{DD}$ be defined as in (7.4), where the coefficients $\tilde{a}, \tilde{b}$ and $\tilde{c} = \tilde{c}^{(1)} + \tilde{c}^{(2)}$ take constant values a_i, b_i and c_i on each strip Ω_i. The constants a_i, b_i and c_i are chosen so that they represent average values of the coefficients $a(x,y), b(x,y)$ and $c(x,y)$ of (7.1) on the subdomains. For example, we can use:

$$\tilde{a}_i = \frac{1}{2}\Big(\inf_{\Omega_i} a(x,y) + \sup_{\Omega_i} a(x,y)\Big) \tag{7.20}$$

$$\tilde{b}_i = \frac{1}{2}\Big(\inf_{\Omega_i} b(x,y) + \sup_{\Omega_i} b(x,y)\Big) \tag{7.21}$$

$$\tilde{c}_i = \frac{1}{2}\Big(\inf_{\Omega_i} c(x,y) + \sup_{\Omega_i} c(x,y)\Big) \quad . \tag{7.22}$$

Let $\tilde{M}_{DD}$ (or $\tilde{M}_{DD}(k)$) be defined as the discretization of $\tilde{L}_{DD}$ on the same n by m grid.

Since this represents a special case of the operator given by (7.4), the results of Theorem 7.1 hold for $\tilde{M}_{DD}$, i.e. $\kappa(\tilde{M}_{DD}^{-1}A)$ is bounded by a constant independent of h. Also, by taking the coupling between interfaces into account, the preconditioner does not deteriorate as k increases. In fact, $\kappa(\tilde{M}_{DD}^{-1}A)$ is also bounded independent of k. Moreover, in some cases the condition number decreases as k increases. The bounds (7.12) hold in this case with

$$\rho_{a_i} = \frac{\sup_{\Omega_i} a(x,y)}{\inf_{\Omega_i} a(x,y)}$$

$$\rho_{b_i} = \frac{\sup_{\Omega_i} b(x,y)}{\inf_{\Omega_i} b(x,y)}$$

$$\rho_{c_i} = \frac{8\delta + \sup_{\Omega_i} c(x,y)}{8\delta + \inf_{\Omega_i} c(x,y)} \quad .$$

Also, (7.13) holds when the approximate coefficients are defined by (7.20-22).

Since the subdomains can only become smaller in one dimension, the ultimate bound depends on the variability of the coefficients in the direction parallel to the strips. In the particular case for which the coefficients a, b and c are constant with respect to x (for horizontal strips) or y (vertical strips), the bound ρ tends to one as the strips become narrow. The reason for choosing piece-wise constant as opposed to piece-wise separable coefficients is that the operator $\tilde{M}_{DD}$ can be solved efficiently by domain decomposition techniques.

In Chapter 4 we showed that $\tilde{M}_{DD}$ has a block decomposition of the form (7.17) with a Schur complement M given by (7.19). Even though $\tilde{M}_{DD}$ can be defined independently as the discretization of a global operator on Ω, a block decomposition shows that a system of the form $\tilde{M}_{DD}v = r$ can be easily solved by block elimination. Fast Poisson solvers can be applied to the solution of the constant coefficient subdomain problems (with matrix $\tilde{A}_\Omega$) and the reduced system (with matrix M) can be solved by Fourier analysis.

The details of the parallel implementation are similar to those described in Chapter 6 for the Poisson equation. The subdomain problems are solved in parallel. The solution of the reduced system involves two FFT computations per interface, which can be done in parallel and the solution of n tridiagonal systems distributed among the processors.

If a block diagonal matrix such as the preconditioner M_{DP} of subsection 7.4.1 was replaced by the exact Schur complement M in (7.17), the work per iteration might be reduced, because it would not be necessary to solve a coupled system for the interfaces, but the savings would be immediately overcome by a deterioration of the convergence rate.

7.6 The Spectrum of $\tilde{M}^{-1}A$

When looking for block preconditioners of the form (7.17), one would like to treat the approximation of the Schur complement C by the operator M and the approximation of the subdomain blocks A_Ω and P as two independent problems. Theorem 7.2 shows that if $\tilde{A}_\Omega = A_\Omega$ and $\tilde{P} = P$, then algorithm PCG applied to (7.15) with preconditioner M is equivalent to algorithm PCG applied to (7.14) with preconditioner $\tilde{M}$.

When the subdomain problems are not solved exactly, the preconditioner may loose the spectral equivalence property. Borgers [6] shows that, even when $\tilde{A}_\Omega$ is spectrally equivalent to A_Ω and M is spectrally equivalent to C, the condition number of $\tilde{M}^{-1}A$ may grow linearly with $1/h$. In his example, this is a consequence of not scaling the off-diagonal block $\tilde{P}$ accordingly.

We shall show now that when $\tilde{A}_\Omega^{-1}\tilde{P} = A_\Omega^{-1}P$, the approximation of the Schur complement C by M and the approximation of the subdomain blocks A_Ω and P can in fact be treated as two independent problems, because the spectrum of $\tilde{M}^{-1}A$ is the union of the spectra of $\tilde{A}_\Omega^{-1}A_\Omega$ and $M^{-1}C$. In general, $\tilde{M}^{-1}A$ can be written in terms of $\tilde{A}_\Omega^{-1/2}A_\Omega\tilde{A}_\Omega^{-1/2}$, $M^{-1/2}CM^{-1/2}$ and the operator $A_\Omega^{-1}P - \tilde{A}_\Omega^{-1}\tilde{P}$.

Theorem 7.3

i) *By a similarity transformation, the preconditioned matrix $\tilde{M}^{-1}A$ becomes:*

$$\begin{pmatrix} I & 0 \\ Y^T & I \end{pmatrix} \begin{pmatrix} \tilde{A}_\Omega^{-1/2}A_\Omega\tilde{A}_\Omega^{-1/2} & 0 \\ 0 & M^{-1/2}CM^{-1/2} \end{pmatrix} \begin{pmatrix} I & Y \\ 0 & I \end{pmatrix} \quad ,$$

where $Y = \tilde{A}_\Omega^{1/2}XM^{-1/2}$ and

$$X = A_\Omega^{-1}P - \tilde{A}_\Omega^{-1}\tilde{P} \quad .$$

ii) *If* $\tilde{A}_\Omega^{-1}\tilde{P} = A_\Omega^{-1}P$, *then* $\sigma(\tilde{M}^{-1}A) = \sigma(\tilde{A}_\Omega^{-1}A_\Omega) \bigcup \sigma(M^{-1}C)$.

Proof:
i) By (7.17), we have $\tilde{M} = Q^T Q$, where

$$Q = \begin{pmatrix} \tilde{A}_\Omega^{1/2} & 0 \\ 0 & M^{1/2} \end{pmatrix} \begin{pmatrix} I & \tilde{A}_\Omega^{-1}\tilde{P} \\ 0 & I \end{pmatrix} \quad . \tag{7.23}$$

Consider the following similarity transformation of $\tilde{M}^{-1}A$:

$$\hat{A} \equiv Q\tilde{M}^{-1}AQ^{-1} = Q(Q^TQ)^{-1}AQ^{-1} = Q^{-T}AQ^{-1} \quad .$$

Then we have:

$$\begin{aligned} \hat{A} &= \begin{pmatrix} \tilde{A}_\Omega^{-1/2} & 0 \\ 0 & M^{-1/2} \end{pmatrix} \begin{pmatrix} I & 0 \\ X^T & I \end{pmatrix} \begin{pmatrix} A_\Omega & 0 \\ 0 & C \end{pmatrix} \begin{pmatrix} I & X \\ 0 & I \end{pmatrix} \begin{pmatrix} \tilde{A}_\Omega^{-1/2} & 0 \\ 0 & M^{-1/2} \end{pmatrix} \\ &= \begin{pmatrix} I & 0 \\ Y^T & I \end{pmatrix} \begin{pmatrix} \tilde{A}_\Omega^{-1/2}A_\Omega\tilde{A}_\Omega^{-1/2} & 0 \\ 0 & M^{-1/2}CM^{-1/2} \end{pmatrix} \begin{pmatrix} I & Y \\ 0 & I \end{pmatrix} \quad . \end{aligned}$$

ii) When $X = 0$, we have

$$\hat{A} = \begin{pmatrix} \tilde{A}_\Omega^{-1/2}A_\Omega\tilde{A}_\Omega^{-1/2} & 0 \\ 0 & M^{-1/2}CM^{-1/2} \end{pmatrix} \quad .$$

∎

It is easy to see that Theorem 7.2 can be derived as a particular case of Theorem 7.3, because if $\tilde{A}_\Omega = A_\Omega$ and $\tilde{P} = P$, then $\sigma(\tilde{M}^{-1}A) = \{1\} \bigcup \sigma(M^{-1}C)$. Thus, the convergence properties of algorithm PCG applied to (7.15) with preconditioner M are the same as for PCG applied to (7.14) with preconditioner $\tilde{M}$ (cf. Lemma 2.5).

It is clear from Theorem 7.3 that if $\tilde{A}_\Omega^{-1}\tilde{P} = A_\Omega^{-1}P$ and the condition numbers of $\tilde{A}_\Omega^{-1}A_\Omega$ and $M^{-1}C$ are small, then we get rapid convergence, even when the condition number of $\tilde{M}^{-1}A$ may be large. Borgers [6] uses the example $\tilde{A}_\Omega = \alpha A_\Omega^{-1}$ for $\alpha > 1$ and $\tilde{P} = P$ with M spectrally equivalent to C, to show that the condition number of $\tilde{M}^{-1}A$ may grow linearly with $1/h$, However, if we allow for the off-diagonal block $\tilde{P}$ to be scaled accordingly, i.e., $\tilde{P} = \alpha P$, then by Theorem 7.3-*ii* we have $\sigma(\tilde{M}^{-1}A) = \alpha \bigcup \sigma(M^{-1}C)$ and therefore $\tilde{M}$ is spectrally equivalent to A and the convergence rate does not depend on α (even though the condition number of $\tilde{M}^{-1}A$ does).

7.7 Separable Approximations on the Subdomains

Since a variable coefficient operator can be usually better approximated by a separable coefficient operator than by constant coefficients, and since the amount of work required to solve separable problems is comparable to the amount of work required to solve constant coefficient problems [32], one can consider replacing the subdomain blocks with separable coefficient approximations instead of constant coefficients. More specifically, if $\tilde{M}_{DD}$ is decomposed as in (7.17), the new preconditioner would be given by

$$\overline{M} = \begin{pmatrix} I & 0 \\ \overline{P}^T\overline{A}_\Omega^{-1} & I \end{pmatrix} \begin{pmatrix} \overline{A}_\Omega & 0 \\ 0 & M \end{pmatrix} \begin{pmatrix} I & \overline{A}_\Omega^{-1}\overline{P} \\ 0 & I \end{pmatrix} \quad ,$$

where $\overline{A}_\Omega$ and $\overline{P}$ represent separable approximations of A_Ω and P, respectively, but M is still the Schur complement for the piece-wise constant approximation.

7.8 Numerical Examples

Example 1 In our experiments, we first considered the problem

$$\frac{\partial}{\partial x}(e^{\alpha xy}\frac{\partial u}{\partial x}) + \frac{\partial}{\partial y}(e^{-\alpha xy}\frac{\partial u}{\partial y}) = f \tag{7.24}$$

on the unit square, for $\alpha = 1$ and for $\alpha = 3$. The right hand side f was chosen so that $u(x,y) = xe^{xy}\sin(\pi x)\sin(\pi y)$ is the solution to (7.24). The problem was solved by the PCG method with the strip preconditioners $\tilde{M}_{DD}(k)$ and $\overline{M}(k)$. The piece-wise constant coefficients for $\tilde{M}_{DD}(k)$ were defined by the arithmetic average of the variable coefficients at all interior gridpoints on each subdomain. Similar results were obtained when the constant coefficients were defined according to the minimax criterion. The preconditioner $\overline{M}(k)$ was described in Section 7.7. The separable coefficients on each subdomain were chosen so that $\tilde{a}_i(x) = k \int_{y_{i-1}}^{y_i} e^{\alpha xy}dy$, where $y_i = \frac{i}{k}$ and $\tilde{b}_i(y) = \int_0^1 e^{-\alpha xy}dx$.

The larger the value of α, the larger the variation of the coefficients inside the subdomains. When $\alpha = 1$, for example, the partial derivatives of the coefficients with respect to x are bounded by $|a_x| < 2.72$ and $|b_x| < 1$. For $\alpha = 3$, we have $|a_x| < 60.26$ and $|b_x| < 3$. While the variation of the coefficients with respect to y can be better represented by using more horizontal strips, the variation with respect to x cannot. For that reason, we expect the convergence rate to be worse for larger values of α and to not improve dramatically as the strips become thin. We will also see that the rate of convergence is significantly improved when the subdomain solvers are replaced by separable approximations.

In Table 7.1 we show the number of iterations needed to reduce the 2-norm of the initial residual by a factor of 10^{-4}, for the cases $\alpha = 1$ and $\alpha = 3$ and for mesh widths $h = 2^{-4}$, $h = 2^{-5}$, $h = 2^{-6}$ and $h = 2^{-7}$. The numbers in parentheses are the average residual reduction per iteration, defined as $(\|r_I\|_2/\|r_0\|_2)^{1/I}$. For comparison, the table also shows the number of iterations and the average residual reduction per iteration when the Laplacian operator is used as preconditioner. Both the Laplacian operator and $\tilde{M}_{DD}(k)$ are spectrally equivalent to the matrix A. Although for the problem sizes shown the average residual reduction per iteration still grows with problem size, this growth slows down more rapidly for preconditioners $\tilde{M}_{DD}(k)$ and $\overline{M}(k)$ than it does for the Laplacian operator. For a given problem size, the reduction rate is not only bounded but in fact, it improves slightly with increasing number of strips when preconditioner $\tilde{M}_{DD}(k)$ is used. On the other hand, with $\overline{M}(k)$ we can see that a two-strip solver represents a significant improvement over a separable approximation on the whole domain, but the reduction rate gets worse as more strips are added. This fact gives an indication of the two approximations that are taking place: first, the approximation of the operator inside the subdomains, which improves as the subdomains get smaller; and second, the approximation of the interface operator, which takes greater significance as it gets larger. In general, for preconditioner $\overline{M}(k)$ we can expect one particular value of k to produce the optimal reduction rate.

In Fig. 7.1 we plot the 2-norm of the residual vs. iteration number, for the case $\alpha = 3$ when $h = 1/32$ and for four different preconditioners: L is the Laplacian operator, $\overline{M}(1)$ is a separable approximation on the whole domain, $\tilde{M}_{DD}(4)$ is the domain-decomposed four-strip preconditioner with piece-wise constant coefficients and $\overline{M}(4)$, the domain-decomposed four-strip preconditioner with separable approximations on the subdomains.

Table 7.1: Numerical Example 1

$$\frac{\partial}{\partial x}(e^{\alpha xy}\frac{\partial u}{\partial x}) + \frac{\partial}{\partial y}(e^{-\alpha xy}\frac{\partial u}{\partial y}) = f$$

		$\alpha = 1$			$\alpha = 3$		
h	k	$M_{DD}(k)$	$\overline{M}(k)$	L	$M_{DD}(k)$	$\overline{M}(k)$	L
1/16	1	6 (0.208)	6 (0.167)	8 (0.298)	15 (0.519)	14 (0.492)	18 (0.589)
	2	6 (0.194)	5 (0.107)		14 (0.506)	8 (0.297)	
	4	6 (0.180)	5 (0.128)		13 (0.483)	9 (0.325)	
1/32	1	7 (0.255)	6 (0.190)	9 (0.351)	18 (0.597)	16 (0.549)	26 (0.696)
	2	7 (0.227)	5 (0.119)		18 (0.589)	9 (0.343)	
	4	6 (0.210)	5 (0.144)		16 (0.559)	10 (0.367)	
	8	6 (0.204)	6 (0.161)		15 (0.537)	11 (0.410)	
1/64	1	8 (0.303)	6 (0.203)	10 (0.387)	21 (0.639)	17 (0.578)	33 (0.753)
	2	7 (0.248)	5 (0.128)		20 (0.627)	10 (0.382)	
	4	7 (0.231)	5 (0.157)		19 (0.603)	11 (0.416)	
	8	7 (0.222)	6 (0.176)		17 (0.580)	12 (0.463)	
	16	7 (0.221)	6 (0.187)		17 (0.571)	13 (0.483)	
1/128	1	8 (0.303)	6 (0.211)	11 (0.411)	23 (0.666)	18 (0.595)	40 (0.793)
	2	7 (0.261)	5 (0.135)		22 (0.652)	10 (0.398)	
	4	7 (0.242)	6 (0.169)		20 (0.631)	12 (0.452)	
	8	7 (0.232)	6 (0.193)		19 (0.610)	14 (0.505)	
	16	7 (0.230)	6 (0.200)		18 (0.599)	15 (0.530)	
	32	7 (0.230)	6 (0.207)		18 (0.596)	16 (0.541)	

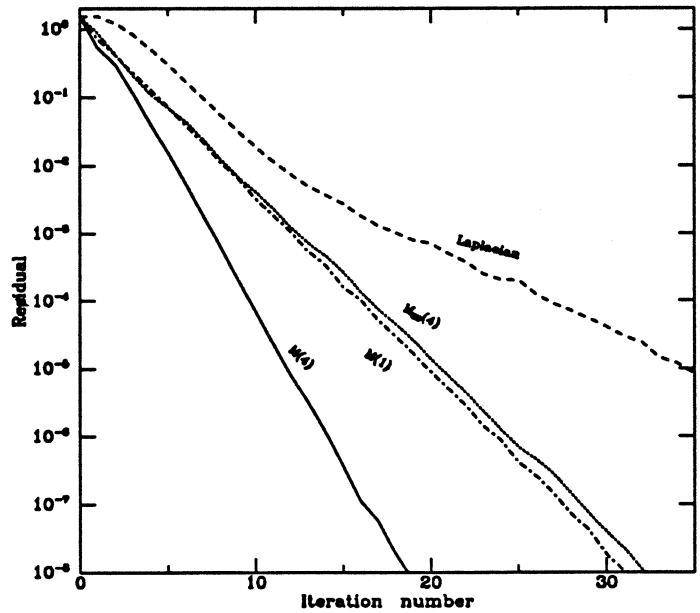

Figure 7.1: Residual reduction for Example 1, with $\alpha = 3$, $h = 2^{-5}$.

μ = 3.E+02	μ = 1.E-04	μ = 3.14E+4	μ = 5
μ = 5.E-02	μ = 8	μ = 7.E-02	μ = 2.7E+03
μ = 1.E+06	μ = 1.E-01	μ = 2.E+02	μ = 9
μ = 1	μ = 6.E+03	μ = 4	μ = 1.4E+5

Figure 7.2: Coefficients for Example 2

Example 2 The second example was given in [7] and it is a problem for which the coefficients are piecewise constant. The operator L has the form 7.1, with $a(x,y) = b(x,y) = \mu(x,y)$ and $c(x,y) = 0$, where μ is a piecewise constant function which takes different values on the sixteen regions pictured in Fig. 7.2. The right hand side f was chosen so that the solution to $Lu = f$ is

$$u(x,y) = (1-4x)^2\,(1-\frac{4}{3}x)^2(1-4y)^2\,(1-\frac{4}{3}y)^2$$

The problem was solved by the PCG method with the strip preconditioners $\tilde{M}_{DD}(4)$ and $\overline{M}(4)$. The piece-wise constant coefficients for $\tilde{M}_{DD}(4)$ were chosen to be the average of μ inside each strip, i.e.

$$\begin{aligned}
\tilde{a}_1 &= \tilde{b}_1 = 36501.25 \\
\tilde{a}_2 &= \tilde{b}_2 = 250052.275 \\
\tilde{a}_3 &= \tilde{b}_3 = 677.03 \\
\tilde{a}_4 &= \tilde{b}_4 = 7926.250025
\end{aligned}$$

and $\tilde{c}_i = 0$. In $\overline{M}(4)$, the subdomain problems are solved exactly by a strip solver after reordering the gridpoints. In fact, given the block LU-decomposition 7.16 for the matrix A, the preconditioner $\overline{M}(4)$ is given by:

$$\overline{M}(4) = \begin{pmatrix} I & 0 \\ P^T A_\Omega^{-1} & I \end{pmatrix} \begin{pmatrix} A_\Omega & 0 \\ 0 & M \end{pmatrix} \begin{pmatrix} I & A_\Omega^{-1} P \\ 0 & I \end{pmatrix} \quad ,$$

where M is the Schur complement corresponding to the interface points for a piece-wise constant coefficient approximation on four horizontal strips.

According to Theorem 7.2, applying $\overline{M}(4)$ is equivalent to applying the preconditioner M to the solution of the interface system.

In Table 7.2 we show the number of iterations needed to reduce the 2-norm of the initial residual by a factor of 10^{-4} and the average residual reduction per iteration for mesh widths $h = 2^{-5}$, $h = 2^{-6}$ and $h = 2^{-7}$ and for preconditioners $\tilde{M}_{DD}(4)$, $\overline{M}(4)$ and the Laplacian operator. As we can see, the number of iterations remains bounded for increasing problem size. The rate of convergence improves significantly when the preconditioner $\overline{M}(4)$ is used. In Fig. 7.3 the 2-norm of the residual is plotted vs. the iteration number for the case $h = 1/32$.

Table 7.2: Numerical Example 2

h	L	$\tilde{M}_{DD}(4)$	$\overline{M}(4)$
1/32	57 (0.844)	24 (0.673)	11 (0.429)
1/64	57 (0.849)	24 (0.673)	12 (0.448)
1/128	61 (0.859)	22 (0.644)	12 (0.441)

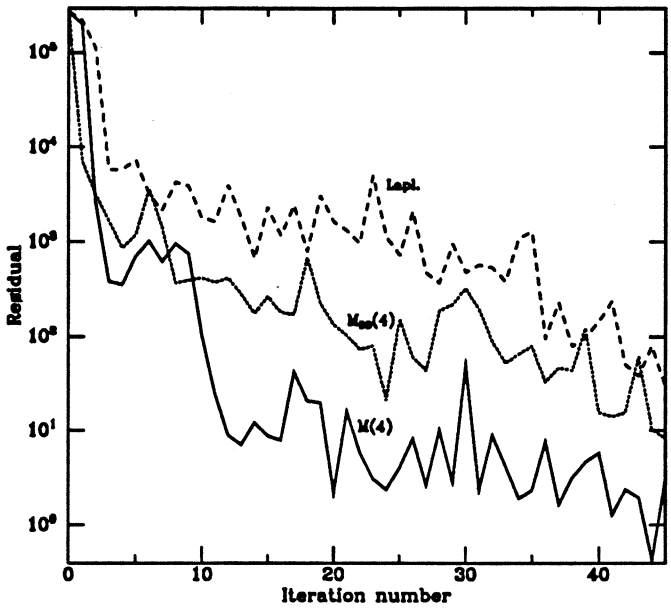

Figure 7.3: Residual reduction for Example 2

7.9 Concluding Remarks

Preconditioners for domain decomposition that have the property of spectral equivalence are often called optimal, because they provide the same complexity as a fast solver, except for a constant factor which depends on the problem and the particular preconditioner. The term optimal can be misleading. First, it might create the impression that these preconditioners are always faster than other so called non-optimal preconditioners, when this is only an asymptotic result. Second, because it might appear that, given two domain decomposition preconditioners which are both "optimal", they are basically equivalent. But considerations such as arithmetic cost, parallelizability, etc., should not be ignored. Finally, we should not ignore the problem of choosing a good separable approximation. In the sense just described, the discrete Laplacian is also *optimal*, because it is spectrally equivalent to any regular second order self-adjoint operator. If we treated all separable aproximations as equivalent, then the Laplacian is the simplest choice, and then we would only be concerned with the problem of solving Poisson's equation efficiently, either sequentially or in parallel. But in fact, the choice of a separable approximation does matter, because it affects the iteration count directly, and each iteration is expensive.

We conclude this section with some comparisons between preconditioners for decompositions with cross-points, such as M_{BPS}, and the strip preconditioners presented in this chapter. Cross-point decompositions apply to more general domains, while strip decompositions only apply to rectangular domains. Cross-point decompositions give the possibility of approximating the operator more closely on boxes than we could on strips, but this must be balanced against the fact that the cross-point system cannot be solved exactly, even for piece-wise constant problems.

Another feature of strip decompositions is that, as we showed in Chapter 6, the computation for the subdomain solves can be organized in such a way that the complexity for solving two problems on each strip does not introduce a factor of two. It would be less trivial to save such factor of two for box domains.

As long as the subdomains are not trivially small, the subdomain solves dominate the computation. In this case, we can say that all domain decomposition preconditioners have similar cost per iteration (except for, possibly, the factor of two). Then, the convergence rate is a reasonably good comparison criterion. For problems that can be well approximated by piece-wise constant coefficients on strips, our strip preconditioner will perform better than box preconditioners.

7.9.1 A Note on Granularity

An important issue in the study of domain decomposition is granularity. The whole reason why the idea of domain decomposition is appealing for parallel computation is based on the assumption that the amount of work required to solve the subdomain problems is large compared to the rest of the computation, which involves most of the overhead. In other words, we usually assume coarse granularity. All domain decomposition algorithms can of course be applied to cases where the subdomains are very small (or thin, in the case of strip-wise decompositions), i.e. fine granularity, but the analysis of the method should be approached from a different point of view.

For example, when we say that it is important to choose a good separable approximation, it is because we assume that the subdomain solves will dominate the computation, and then we want to replace the subdomain problems by operators that can be solved by fast methods. At the same time, for preconditioners which solve a cross-point system, it is not so crucial to specify the method for solving this cross-point system, because it represents a lower order in the complexity analysis.

On the other hand, when the subdomains are small enough, it is no longer necessary to approximate the subdomain operators, because the smaller size problems can be solved exactly. But more emphasis should be put on the description of how to handle the interfaces or the cross points.

Bibliography

[1] C. R. ANDERSON, *On Domain Decomposition*, Manuscript CLaSSiC-85-09, Center for Large Scale Scientific Computation, Stanford University, Stanford, CA, 1985.

[2] R. BANK and D. ROSE, *Marching Algorithms for Elliptic Boundary Value Problems. I: The Constant Coefficient Case*, SIAM J. Numer. Anal., 14 (1977), pp. 792-828.

[3] R. BANK, *Marching Algorithms for Elliptic Boundary Value Problems. II: The Variable Coefficient Case*, SIAM J. Numer. Anal., 14 (1977), pp. 950-970.

[4] P. E. BJORSTAD and O. B. WIDLUND, *Iterative Methods for the Solution of Elliptic Problems on Regions Partitioned into Substructures*, SIAM J. Numer. Anal., 23 (1986), pp. 1097-1120.

[5] P. E. BJORSTAD and O. B. WIDLUND, *To Overlap or not to Overlap: A Note on a Domain Decomposition Method for Elliptic Problems*, Tech. Rep., Institute of Informatics, University of Bergen, Bergen, Norway, 1988. To appear in SIAM J. Sci. Stat. Comput.

[6] C. BÖRGERS, *The Neumann-Dirichlet Domain Decomposition Method with Inexact Solvers on the Subdomains*, Numer. Math. 55 (1989), pp. 123-136.

[7] J. H. BRAMBLE, J. E. PASCIAK and A. H. SCHATZ, *An Iterative Method for Elliptic Problems on Regions Partitioned into Substructures*, Math. Comp. 46 (1986), pp. 361-369.

[8] J. H. BRAMBLE, J. E. PASCIAK and A. H. SCHATZ, *The Construction of Preconditioners for Elliptic Problems by Substructuring I*, Math. Comp. 47 (1986), p. 103.

[9] O. BUNEMAN, *A Compact non-Iterative Poisson Solver*, Rep. 294, Stanford University Institute for Plasma Research, Stanford, CA, 1969.

[10] B. L. BUZBEE, G. H. GOLUB and C. W. NIELSON, *On Direct Methods for Solving Poisson's Equation*, SIAM J. Numer. Anal. 7 (1970), pp. 627-656.

[11] R. CHANDRA, *Conjugate Gradient Methods for Partial Differential Equations*, PhD Thesis, Dept. of Computer Science, Yale Univ., 1978.

[12] T. F. CHAN, *Analysis of Preconditioners for Domain Decomposition*, SIAM J. of Numer. Anal. 24 (1987), pp. 382-390.

[13] T. F. CHAN and D. GOOVAERTS, *Schwarz=Schur: Overlapping Versus Nonoverlapping Domain Decomposition*, CAM Report 88-21, UCLA, Los Angeles, CA, 1988.

[14] T. F. CHAN, T. Y. HOU and P. L. LIONS, *Geometry-Independent Convergence Results for Domain Decomposition Algorithms*, CAM Report 89-11, UCLA, Los Angeles, CA, 1989.

[15] T. F. CHAN and D. C. RESASCO, *A Domain-Decomposed Fast Poisson Solver on a Rectangle*, SIAM J. Sc. Stat. Comp. 8 (1987), pp. s14-s26.

[16] D. BAXTER, J. SALTZ, M. SCHULTZ, S. EISENSTAT and K. CROWLEY, *An Experimental Study of Methods for Parallel Preconditioned Krylov Methods*, Research Report 629, Yale University, New Haven, CT, 1988.

[17] P. CONCUS and G. GOLUB, *Use of Fast Direct Methods for the Efficient Numerical Solution of Nonseparable Elliptic Equations*, SIAM J. Num. Anal. 10 (1973), pp. 309-332.

[18] M. DRYJA, *A Capacitance Matrix Method for Dirichlet Problem on Polygonal Region*, Numer. Math. 39 (1982), pp. 51-64.

[19] M. DRYJA, *A Finite Element-Capacitance Method for Elliptic Problems on Regions Partitioned into Subregions*, Numer. Math. 44 (1984), pp. 153-168

[20] M. DRYJA and W. PROSKUROWSKI, *Fast Elliptic Solvers on Rectangular Regions Subdivided into Strips*, Advances in Computer Methods for Partial Differential Equations. V, R. Vichnevetsky and R. Stepleman (Eds.), Publ. IMACS, 1984.

[21] M. DRYJA and W. PROSKUROWSKI, *Capacitance Matrix Method using Strips with Alternating Neumann and Dirichlet Boundary Conditions*, Appl. Numer. Math., 1 (1985), pp. 285-298.

[22] M. DRYJA, W. PROSKUROWSKI and O. WIDLUND, *Numerical Experiments and Implementation of a Domain Decomposition Method with Cross Points for the Model Problem*, Technical Report CRI-86-37, Univ. of South. Cal., Los Angeles, CA, 1986.

[23] T. DUPONT, R. P. KENDALL and H. H. RACHFORD, *An Approximate Factorization Procedure for Solving Self-Adjoint Elliptic Difference Equations*, SIAM J. on Num. Anal., 6 (1968), pp. 753-782.

[24] H. C. ELMAN, *Iterative Methods for Large, Sparse, Nonsymmetric Systems of Linear Equations* , PhD Thesis, Dept. of Computer Science, Yale University, New Haven, CT, 1982.

[25] H. ELMAN and M. SCHULTZ, *Preconditioning by Fast Direct Methods for Nonself-adjoint Nonseparable Elliptic Equations*, SIAM J. on Num. Anal., 23 (1986), pp. 44-57.

[26] G. H. GOLUB and D. MAYERS, *The Use of Pre-Conditioning over Irregular Regions*, R. Glowinski and J.L. Lions, ed., *Computing Methods in Applied Sciences and Engineering VI*, North-Holland, 1984, pp. 3-14.

[27] R. W. HOCKNEY, *A Fast Direct Solution of Poisson's Equation using Fourier Analysis*, J. ACM, 12 (1965) pp. 95-113.

[28] D. KEYES and W. GROPP, *A Comparison of Domain Decomposition Techniques for Elliptic Partial Differential Equations*, SIAM J. Sc. Stat. Comp. 8 (1987), pp. s166-s202.

[29] F. SAIED, C.-T. HO, S. L. JOHNSSON and M. H. SCHULTZ, *Solving Schrödinger's equation on the Intel iPSC by the Alternating Direction Method*, in Hypercube Multiprocessors 1987, M. T. Heath (Ed.), SIAM, Philadelphia, 1987.

[30] F. SAIED, *Numerical Techniques for the Solution of Schrödinger's Equation and their Parallel Implementation*, PhD Thesis, Dept. of Computer Science, Yale Univ., 1990.

[31] H. A. SCHWARZ, *Gesammelte Mathematische Abhandlungen* Berlin, Springer, 2 (1890), pp. 133-134.

[32] P. N. SWARZTRAUBER, *A Direct Method for the Discrete Solution of Separable Elliptic Equations*, SIAM J. Numer. Anal., 11 (1974), pp. 1136-1150.

[33] P. N. SWARZTRAUBER, *A Package of Fortran Subprograms for the Fast Fourier Transform of Periodic and other Symmetric Sequences*, National Center for Atmospheric Research, Boulder, Colorado.

[34] P. N. SWARZTRAUBER and R. SWEET, *Efficient Fortran Subprograms for the Solution of Elliptic Partial Differential Equations*, Technical Note TN/IA-109, July 1975, National Center for Atmospheric Research, Boulder, Colorado.

[35] R. S. VARGA, *Matrix Iterative Analysis* (1962) Prentice-Hall, Inc.