

IBM Research Report RC 13778 (revised)

A Parallel Domain Reduction Method

Franco Brezzi

Dipartimento di Meccanica Strutturale e
Istituto di Analisi Numerica del C. N. R.
Università di Pavia
27100 Pavia (Italia)

Craig C. Douglas

Mathematical Sciences Department
IBM Research Division
Thomas J. Watson Research Center
P. O. Box 218
Yorktown Heights, NY 10598

L. Donatella Marini

Istituto di Analisi Numerica del C. N. R.
Università di Pavia
27100 Pavia (Italia)

Abbreviated Title: Domain Reduction Methods

Abstract: We relate a particular version of a parallel multigrid method analyzed by C. Douglas, W. L. Miranker, and B. F. Smith ([4, 6]) to the domain decomposition method of D. Funaro, L. D. Marini, A. Quarteroni, and P. Zanolli ([7, 8]). We show that the parallel multigrid method is reducing computation to a small portion of the domain and then extending the solution to the entire domain using the correct reflections to get the exact solution. We extend a particular example to double the parallelism in a nonobvious manner. While the techniques of this paper are applied to two dimensional problems, they can be applied to higher dimensional problems in an obvious manner.

Keywords: partial differential equations, parallel multigrid, domain decomposition, direct method

AMS(MOS) subject classification: 35, 65

To appear in *Numerical Methods for Partial Differential Equations*.

In [4], a particular version of a parallel multigrid method is analyzed in a very abstract manner, but does not offer much information in order to be useful for solving elliptic boundary value problems. This is rectified in [6], but missed half of the possible parallelism for a special case. In this paper, we introduce a technique involving folding the domain into one or more small portions of the domain, doing all computations in parallel in the reduced domains (hence the terminology *domain reduction method*), and then extending the solutions to the entire domain. We remark on the similarities and differences between the domain reduction method and domain decomposition methods. Under conditions derived in [6], the domain reduction algorithm results in a direct method. However, the cost is approximately equal to that of one iteration of a standard domain decomposition algorithm, which usually requires many iterations to reduce the error sufficiently. Hence, the domain reduction method may be orders of magnitude more efficient than domain decomposition methods.

In this paper, we approximate the solution to the elliptic boundary value problem

$$\begin{cases} \mathcal{L}u = f & \text{in } \Omega \\ u = 0 & \text{on } \partial\Omega_D \\ \frac{\partial}{\partial n}u = 0 & \text{on } \partial\Omega_N \end{cases}, \text{ where } \partial\Omega = \partial\Omega_D \cup \partial\Omega_N, \text{ and } \partial\Omega_D \cap \partial\Omega_N = \phi, \quad (1)$$

using a parallel multigrid algorithm. We assume that (1) is well posed, so that it has a unique solution $u \in \mathcal{H}$, where \mathcal{H} typically is a Sobolev space. Further,

$$\mathcal{L}u = -\sum_{i,j=1}^2 \frac{\partial}{\partial x_i}(a_{ij}(x)\frac{\partial u}{\partial x_j}) + a_0(x)u,$$

where the $\{a_{ij}\}$ are symmetric, uniformly positive definite, bounded, and piecewise smooth on Ω and $a_0 \geq 0$. Our approximate solution lies in a finite dimensional space \mathcal{M} , which is the natural space after (1) has been discretized using a finite difference, element, or volume method to form the discrete problem

$$\mathcal{A}U = F, \quad (2)$$

where $\mathcal{A} \in \mathbb{R}^{N \times N}$ and $U, F \in \mathbb{R}^N$. We use a set of auxiliary non-nested finite dimensional subspaces $\{\mathcal{M}_j\}_{j=1}^d$ to compute the approximate solution.

The subspaces $\{\mathcal{M}_j\}_{j=1}^d$ of \mathcal{M} are defined through the restriction operators $\{\mathcal{R}_j\}_{j=1}^d$:

$$\mathcal{R}_j : \mathcal{M} \rightarrow \mathcal{M}_j.$$

The prolongation operators \mathcal{P}_j map the other way:

$$\mathcal{P}_j : \mathcal{M}_j \rightarrow \mathcal{M}.$$

Frequently, each prolongation operator is the adjoint of one of the restriction operators. We require that

$$\sum_{j=1}^d \mathcal{R}_j^* \mathcal{R}_j = \text{identity operator on } \mathcal{M}. \quad (3)$$

We define operators \mathcal{L}_j and \mathcal{A}_j on the subspaces by

$$\mathcal{L}_j = \mathcal{R}_j(\mathcal{L}(\mathcal{P}_j)).$$

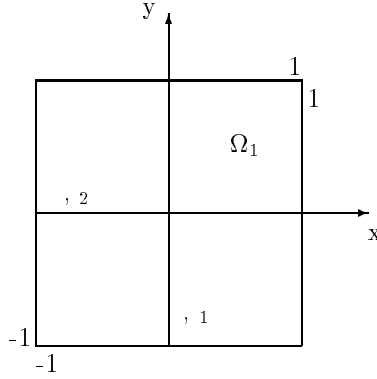
and

$$\mathcal{A}_j = \mathcal{R}_j \mathcal{A} \mathcal{P}_j.$$

Each operator \mathcal{L}_j and \mathcal{A}_j must have an inverse. We note that the existence of each \mathcal{L}_j is guaranteed if $\mathcal{M} \subset \mathcal{H}$. However, this is not always practical, nor necessary: if we are only interested in solving the discrete problem (2), we can ignore this requirement.

The parallel multigrid method is given by

Figure 1: Simple Two Dimensional Domain



```

Algorithm PMG (d, m, n, f, u,  $\mathcal{L}$ ,  $\{\mathcal{L}_j, \mathcal{P}_j, \mathcal{R}_j\}_{j=1}^d$ ) {
  | d = Number of subspaces ( $d > 0$ )
  | m = number of smoothing iterations ( $m \geq 0$ )
  | n = number of iterations ( $n \geq 0$ )
  | f = right hand side
  | u = initial guess/approximate solution
  |  $\mathcal{L}$  = PDE operator on solution space
  |  $\{\mathcal{L}_j\}$  = PDE operators on subspaces
  |  $\{\mathcal{P}_j\}$  = prolongation operators
  |  $\{\mathcal{R}_j\}$  = restriction operators

  Smooth m times on u to get  $u_0$ 
  Do i = 1, ..., n {
    Compute residual  $r_i = f - \mathcal{L}u_{i-1}$ 
    Solve in parallel, j = 1, ..., d:
       $\mathcal{L}_j c_j = \mathcal{R}_j r_i$ 
    Set  $c = u_{i-1} + \sum_{j=1}^d \mathcal{P}_j c_j$ 
    Smooth m times on c to get  $u_i$ 
  }
  Set u =  $u_n$ 
}

```

This generates an approximation to the solution u of (1). The discrete form of Algorithm PMG substitutes the symbol \mathcal{A} for each occurrence of the symbol \mathcal{L} (with or without a subscript) in the definition of PMG. Many properties of this algorithm can be found in [4, 5, 6].

We motivate the rest of this paper with an example which is defined on the domain $\Omega = (-1,1) \otimes (-1,1)$. We partition Ω into four equally sized squares about the origin and define $\Omega_1 = (0,1) \otimes (0,1)$, Ω_2 as y axis $\cap \bar{\Omega}$, and Ω_3 as x axis $\cap \bar{\Omega}$. Figure 1 summarizes these definitions.

We say an operator \mathcal{L} *preserves symmetry and antisymmetry about* Ω_k if \mathcal{L} applied to any even (odd) function about Ω_k is an even (odd) function about Ω_k . For example, the Δ operator preserves symmetry

and antisymmetry about both the x and y axes. We say an operator \mathcal{L} *reverses symmetry and antisymmetry about* $,_k$ if \mathcal{L} applied to any even (odd) function about $,_k$ is an odd (even) function about $,_k$. For example, the ∇ operator reverses symmetry and antisymmetry about both the x and y axes.

Let us introduce the restriction operators associated with Figure 1. For all $(x, y) \in \Omega$, define

$$\mathcal{R}_{00}U(x, y) = \frac{1}{2} [U(x, y) + U(-x, y) + U(x, -y) + U(-x, -y)],$$

$$\mathcal{R}_{11}U(x, y) = \frac{1}{2} [U(x, y) - U(-x, y) - U(x, -y) + U(-x, -y)],$$

$$\mathcal{R}_{01}U(x, y) = \frac{1}{2} [U(x, y) + U(-x, y) - U(x, -y) - U(-x, -y)],$$

and

$$\mathcal{R}_{10}U(x, y) = \frac{1}{2} [U(x, y) - U(-x, y) + U(x, -y) - U(-x, -y)],$$

where the $\frac{1}{2}$ is a scaling factor so that (3) is satisfied. In these definitions, the first index is associated with $,_1$ and the second with $,_2$. Moreover, the value of 0 in the k -th index denotes preserving symmetry and antisymmetry about $,_k$, while the value of 1 denotes reversing symmetry and antisymmetry about $,_k$. These operators annihilate functions which have certain symmetry and antisymmetry properties. For example, \mathcal{R}_{00} annihilates any function which is odd about either of the axes.

Suppose that \mathcal{L} preserves symmetry and antisymmetry about both $,_1$ and $,_2$. If we take

$$\mathcal{P}_{jk} = \mathcal{R}_{jk}^*, \quad j, k = 0, 1,$$

then algorithm PMG converges in one iteration with no smoothing. Suppose that \mathcal{L} reverses symmetry and antisymmetry about both $,_1$ and $,_2$. If we take

$$\mathcal{P}_{jk} = \mathcal{R}_{1-j, 1-k}^*, \quad j, k = 0, 1,$$

then algorithm PMG converges in one iteration with no smoothing. Suppose that \mathcal{L} reverses symmetry and antisymmetry about $,_1$, but preserves symmetry and antisymmetry about $,_2$. If we take

$$\mathcal{P}_{jk} = \mathcal{R}_{1-j, k}^*, \quad j, k = 0, 1,$$

then algorithm PMG converges in one iteration with no smoothing (i.e., $m = 0$ in PMG). Finally, suppose that \mathcal{L} reverses symmetry and antisymmetry about $,_2$, but preserves symmetry and antisymmetry about $,_1$. If we take

$$\mathcal{P}_{jk} = \mathcal{R}_{j, 1-k}^*, \quad j, k = 0, 1,$$

then algorithm PMG converges in one iteration with no smoothing. The proof of each of these statements is given in [6].

Let us analyze a particular class of problems. Suppose that \mathcal{L} preserves symmetry and antisymmetry about both $,_1$ and $,_2$ and that (1) only has Dirichlet boundary conditions, i.e., $\partial\Omega_N = \phi$ (marked by D's):

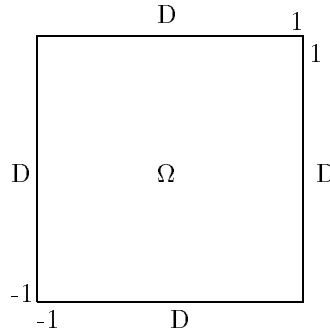
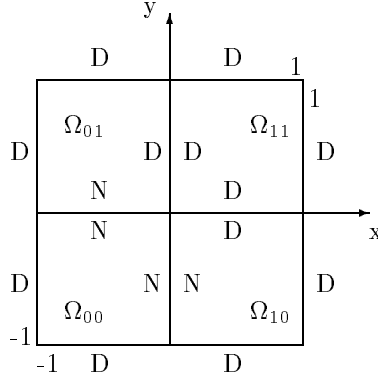


Figure 2: Four Way Domain Decomposition Misrepresentation



We define subspaces \mathcal{M}_{ij} using the \mathcal{R}_{ij} defined earlier. This leads to solving four problems with either homogeneous Dirichlet or Neumann boundary conditions (marked by N's) along each side of the domains:

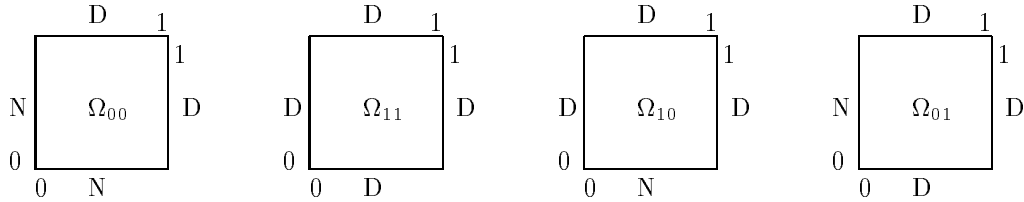


Figure 2 contains a domain decomposition misrepresentation of this. (We use the term misrepresentation because we do not actually distribute the four subdomains in this manner.) This formulation is remarkably similar to that of domain decomposition methods (see [7, 8]). There is one major computational difference between the two methods: in domain decomposition, we solve the four smaller problems encompassing the entire domain, pass boundary information to the neighbors, and continue solving smaller problems for a number of iterations. However, with domain reduction, we solve the four smaller problems, extend the solutions to the entire domain, and add them up to get the solution in one iteration. The cost of extending the four smaller problem solutions is trivial and can be done in parallel.

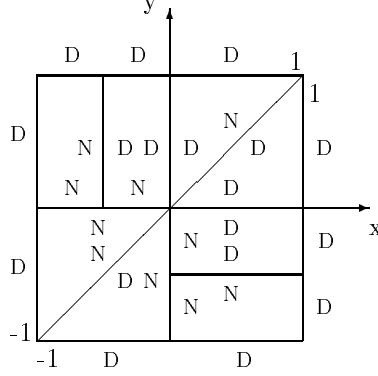
Let us now investigate the more general case of mixed boundary conditions in (1). Define \mathcal{Q}_k as a pointwise reflection operator about \cdot, k :

$$\mathcal{Q}_1(x, y) = (-x, y) \text{ and } \mathcal{Q}_2(x, y) = (x, -y).$$

Suppose $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$, where $\partial\Omega_D \cap \partial\Omega_N = \phi$, and $\partial\Omega_D$ is the part of the boundary with Dirichlet boundary conditions and $\partial\Omega_N$ is the part of the boundary with Neumann boundary conditions. If \mathcal{L} preserves symmetry and antisymmetry about \cdot, k , the techniques of this paper require

$$\mathcal{Q}_k |_{\partial\Omega_D} \subseteq \partial\Omega_D \quad \text{and} \quad \mathcal{Q}_k |_{\partial\Omega_N} \subseteq \partial\Omega_N. \tag{4}$$

Figure 3: Eight Way Domain Decomposition Misrepresentation



On the contrary, if \mathcal{L} reverses symmetry and antisymmetry about x, k , we require

$$\mathcal{Q}_k |_{\partial\Omega_D} \subseteq \partial\Omega_N \quad \text{and} \quad \mathcal{Q}_k |_{\partial\Omega_N} \subseteq \partial\Omega_D.$$

Hence, the use of a more general boundary condition in (1) is acceptable as long as the correct symmetries and antisymmetries in the boundary conditions exist. Otherwise, the operators \mathcal{R}_{jk} and \mathcal{P}_{jk} will no longer have the correct properties.

We can, in fact, do better than a four way domain reduction. Using the eight-fold symmetry property of the square, we know that we should be able to do an eight way domain reduction. The technique we employ is to work with smaller domains which are either triangles or rectangles which encompass one eighth of the domain. This technique is acceptable if the domain has the correct kind of triangulation (or mesh), so that the discrete scheme will adhere to all of the required symmetries.

Figure 3 shows a corresponding domain decomposition misrepresentation for the eight way domain reduction case. The partitioning of the domain suggested by Figure 3 is not unique. We note that (4) requires that Ω_{00} , Ω_{01} , and Ω_{10} be partitioned in a unique manner. However, we can partition Ω_{11} in four different ways: diagonally (as we did), diagonally about the opposite diagonal, or as we partitioned either Ω_{01} or Ω_{10} . Further, it can be partitioned recursively into eight subdomains instead of two.

The restriction operators \mathcal{R}_{ijk} are composed from the operators \mathcal{R}_{ij} , $i, j, k \in \{0, 1\}$. Let

$$sgn(x) = \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases}.$$

For all $(x, y) \in \Omega$, define

$$\mathcal{R}_{000}U(x, y) = \frac{1}{\sqrt{2}} [\mathcal{R}_{00}U(x, y) + \mathcal{R}_{00}U(y, x)],$$

$$\mathcal{R}_{001}U(x, y) = \frac{1}{\sqrt{2}} [\mathcal{R}_{00}U(x, y) - \mathcal{R}_{00}U(y, x)],$$

$$\mathcal{R}_{110}U(x, y) = \frac{1}{\sqrt{2}} [\mathcal{R}_{11}U(x, y) + \mathcal{R}_{11}U(y, x)],$$

$$\mathcal{R}_{111}U(x, y) = \frac{1}{\sqrt{2}} [\mathcal{R}_{11}U(x, y) - \mathcal{R}_{11}U(y, x)],$$

$$\mathcal{R}_{010}U(x, y) = \frac{1}{\sqrt{2}} [\mathcal{R}_{01}U(x, y) + \mathcal{R}_{01}U(x, \text{sgn}(y) - y)],$$

$$\mathcal{R}_{011}U(x, y) = \frac{1}{\sqrt{2}} [\mathcal{R}_{01}U(x, y) - \mathcal{R}_{01}U(x, \text{sgn}(y) - y)],$$

$$\mathcal{R}_{100}U(x, y) = \frac{1}{\sqrt{2}} [\mathcal{R}_{10}U(x, y) + \mathcal{R}_{10}U(\text{sgn}(x) - x, y)],$$

and

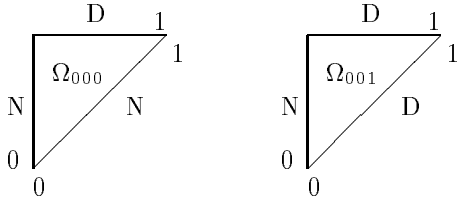
$$\mathcal{R}_{101}U(x, y) = \frac{1}{\sqrt{2}} [\mathcal{R}_{10}U(x, y) - \mathcal{R}_{10}U(\text{sgn}(x) - x, y)],$$

where the $\frac{1}{\sqrt{2}}$ is a scaling factor so that (3) is satisfied. Using these operators, the proof of one iteration convergence with no smoothing is a simple consequence of the linearity of the operator \mathcal{L} (cf., [6]).

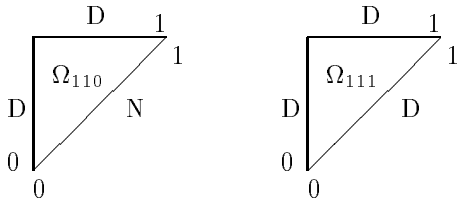
Consider a specific example of particular simplicity, namely Poisson's equation:

$$\begin{cases} -\Delta u = f & \text{in } \Omega \\ u = 0 & \text{on } \partial\Omega_D = \partial\Omega \end{cases} \quad (5)$$

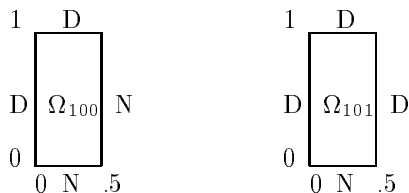
Using the operators \mathcal{R}_{ijk} , each of the operators $\mathcal{L}_{ijk} = -\Delta$. The boundary conditions are specified in the definitions of the corresponding Ω_{ijk} . We reduce the problem on Ω_{00} to solving two boundary value problems on, say, Ω_{000} and Ω_{001} with boundary conditions given by



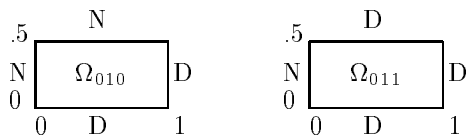
Similarly, we reduce the problem on Ω_{11} to



the problem on Ω_{10} to



and the problem on Ω_{01} to



If we assume that we are solving the subspace problems directly (instead of reducing the domains further), then the cost may be quite high. This depends on how many processors are available for the direct solve and the type of communication between processors (memory to memory, ethernet, bus, etc.). On typical SIMD (single instruction, multiple data) computers with thousands of processors, use of a preconditioned conjugate gradient method is preferable to, say, Gaussian elimination.

The computational cost can be quite low when we are willing to settle for accuracy on the order of the truncation error. In this case, a standard multigrid method, possibly parallelized, can be used to reduce the cost of solving each of the subproblems to (at worst) $O(N/d)$, where d is the number of subproblems (see [1, 2, 3]). (In contrast, some problems in pattern recognition are equivalent to solving a central difference discretization on a uniform mesh of (5), but must be solved more accurately.)

We conclude by noting that domain reduction is a technique which should be viewed as a mathematical preprocessing of a boundary value problem. We advocate transforming the original problem into a collection of problems on smaller domains, and then solving each by the fastest known numerical method (parallel or serial) which applies to the given computer architecture in use. Further, the techniques of this paper are being investigated for higher dimensional problems. Even in two dimensions, they can also be applied to more general domains than just a square.

References

- [1] C. C. Douglas, *Multi-grid algorithms with applications to elliptic boundary-value problems*, SIAM J. Numer. Anal., 21 (1984), pp. 236-254.
- [2] C. C. Douglas, *Multi-grid algorithms for elliptic boundary-value problems*, Ph.D. Thesis, Yale University, May, 1982. Also, available as Computer Science Department Technical Report 223.
- [3] C. C. Douglas and J. Mandel, *Convergence of the domain reduction method with inexact solvers*, IBM Research Report, 1988.

- [4] C. C. Douglas and W. L. Miranker, *Constructive interference in parallel algorithms*, SIAM J. Numer. Anal., 25 (1988), pp. 376-398.
- [5] C. C. Douglas and W. L. Miranker, *Some nontelescopng parallel algorithms based on serial multigrid/aggregation/disaggregation techniques*, in Multigrid Methods: Theory, Applications, and Supercomputing, S. F. McCormick (editor), Marcel Dekker, New York, 1988, pp. 167-176.
- [6] C. C. Douglas and B. F. Smith, *Using symmetries and antisymmetries to analyze a parallel multigrid algorithm*, to appear in SIAM J. Numer. Anal., 26 (1989).
- [7] D. Funaro and A. Quarteroni and P. Zanolli, *An iterative procedure with interface relaxation for domain decomposition methods*, to appear in SIAM J. Numer. Anal., 25 (1988).
- [8] L. D. Marini and A. Quarteroni, *A relaxation procedure for domain decomposition methods using finite elements*, submitted to Numer. Math.