A Multilevel, Parallel, Domain Decomposition, Finite-Difference Poisson Solver

Albert W. Schueller$^a$ and J. M. McDonough$^b$

$^a$Department of Mathematics, Whitman College, Walla Walla, WA USA 99362
E-mail: schuelaw@whitman.edu

$^b$Department of Mechanical Engineering, University of Kentucky, Lexington, KY USA 40506-0108
E-mail: jmmcd@uky.edu

We present a new implementation of multilevel domain decomposition algorithms based on use of a simple successive line overrelaxation technique for the basic solver, and an approximate Schur complement procedure that naturally provides subdomain overlaps in a formally non-overlapping construction. The procedure is parallelized using OpenMP on HP SPP-2200 and N-4000 SMPs, leading to essentially linear speedups through eight processors, and the ability to obtain solutions to elliptic Dirichlet problems in run times independent of problem size on the latter of these two machines.

1. INTRODUCTION

Domain decomposition methods (DDMs) have been under development since at least the mid 1980s, and early intuitive forms of the basic approach, originally due to Schwarz [1], had been used in computational fluid dynamics (CFD) in the guise of multi-block structured methods considerably earlier than 1985 (see Thompson et al. [2] and references therein). Evolution of multi-processor parallel supercomputers began by the late 1980s to early 1990s and has served as one of the prime motivators for studying DDM algorithms. By the early 1990s DDM theory had become well developed, and it was recognized that DDMs share many features with multigrid methods. Moreover, it is now known that in the context of elliptic partial differential equations (PDEs) optimal DDM convergence rates can be achieved only by versions that incorporate some multigrid technology, namely coarse-grid correction (see, e.g., Smith et al. [3]).

It is well known that solving the pressure Poisson equation (PPE)

$$\Delta p = \nabla \cdot U,$$  \hspace{1cm} (1)

with $p$ denoting pseudo pressure—a velocity potential and $U$ the velocity vector, can consume as much as 90% of the total time expenditure in computing solutions to the incompressible Navier–Stokes (N.-S.) equations. Thus, being able to solve the PPE effi-
ciently is crucial. This paper presents further work on an algorithm for this purpose, first described by McDonough et al. [4] and consisting of a multilevel DDM. The ideas embodied in the approach are similar to those presented in [3], but the implementation employed is rather different. In particular, although we view the overall DDM as preconditioning of a simple algorithm, as is done in [3], we here utilize finite-difference discretization rather than the finite elements used in [3], and we employ an easily parallelized red-black ordered successive line overrelaxation (SLOR) as our basic algorithm in place of the Krylov subspace-based methods of [3]. In addition, we introduce overlap of subdomains through an efficient approximate Schur complement procedure.

The principal goal of this study is to produce a general elliptic solver having the following properties: i) total arithmetic scales “nearly” linearly with number of unknowns; ii) run times are essentially independent of number of equations when parallelization is used; iii) properties i) and ii) hold even without symmetry or positive definiteness of the system matrix, and hence for problems posed in generalized coordinates with other than Dirichlet boundary conditions; iv) the method retains its favorable properties for three-dimensional problems. In the present work efforts will be focused on properties i) and ii).

The remainder of this paper is organized as follows. In Sec. 2 we provide a fairly detailed discussion of our multilevel DDM algorithm, and in Sec. 3 we present a model problem with an exact solution. Parallelization results are presented in Sec. 4, followed by summary and conclusions in a final section.

2. ALGORITHM DESCRIPTION

The goal of producing this algorithm is to minimize the time necessary to solve the pressure Poisson equation common in the numerical solution of the incompressible N.-S. equations of fluid dynamics and other elliptic boundary-value problems. Figures 1a,b provide a schematic to aid in understanding the features of this three-level algorithm consisting of the following parts.

![Diagram of three-level red-black ordered grid arrangement](image)

Figure 1. Three-level red-black ordered grid arrangement: (a) basic grid decomposed in red and black regions; (b) approximate Schur complement construction near interior domain boundaries.
A simple multilevel interpolation scheme equivalent to the nested iteration of a full multigrid (FMG) method is used to start the fine grid iteration with a better approximation of the solution and thus save iterations. Domain decomposition is then employed to speed the convergence of the full fine grid SLOR. It is well known that the convergence rate of SLOR slows as the problem size increases; indeed, for 2-D Dirichlet problems the required number of iterations is proportional to \( N^{1/2} \), where \( N \) is the total number of equations, \( i.e., \) grid points, in the system (see, \( e.g., \) Young [5]). Thus after a few iterations on the entire fine grid, the algorithm shifts to artificial Dirichlet problems on subdomains using the current iterate from the fine grid plus an approximate Schur complement to determine boundary conditions on the interior (artificial) boundaries of the subdomains. Because the subdomains are generally much smaller than the original problem and because of stability of the finite-difference equations with respect to perturbations in boundary data, the SLOR on the subdomain shows relatively rapid convergence and overall accuracy is quickly improved on the subdomain.

But because the interior boundary data are in error, there is a limit to how much improvement the subdomain relaxation provides. Thus, after the subdomain relaxations, we return to the full grid for a few more iterations to improve the approximation near the interior boundaries. Indeed, before we return to the full grid, we relax on strips around the vertical and horizontal artificial boundaries (analogous to constructing a Schur complement), again taking advantage of the smaller problem size and stability of the finite-difference equations to perturbations in boundary data.

Details of this algorithm are outlined for the three-grid (coarse, intermediate, fine) multilevel structure displayed in Fig. 1.

**Algorithm** (Multilevel Approximate Schur Complement DDM)

1. Exactly solve problem on coarsest grid.

2. Interpolate this solution to the intermediate grid, and apply (parallelized) RB-ordered SLOR until the asymptotic convergence rate is observed.

3. Interpolate this result to the finest grid, and again apply parallelized RB-ordered SLOR iterated until asymptotic convergence rate is observed.

4. Apply domain decomposition of the finest grid decomposed into red and black subdomains (see Fig. 1). Process all red domains simultaneously on separate processor complexes employing parallelized RB-ordered SLOR within each subdomain. Then repeat for the black subdomains.

5. Compute approximate Schur complements to update subdomain boundary values using RB-ordered SLOR iterated to the overall problem final required convergence tolerance.

6. Perform fixed number of iterations on the global fine grid using parallelized RB-ordered SLOR; check residual at last iteration.

7. If residual satisfies convergence tolerance, stop; otherwise compute approximate Schur complements and return to step 4.
The rationale and specific details associated with this algorithm are contained in the following remarks.

**Remark 1.** The first three steps of the algorithm correspond to “nested iteration,” and hence comprise the right-hand side of a multigrid \( \mathbf{V} \) cycle.

**Remark 2.** The convergence tolerance employed in steps 2 and 3 may need to be relaxed in practice. In any case it is easily checked via a theorem proved in Hageman and Young [6] that

\[
\lim_{n \to \infty} \frac{\| d^{(n+1)} \|}{\| d^{(n)} \|} = \lambda_1, 
\]

where \( \lambda_1 \) is the spectral radius of the iteration matrix of the method, and \( d^{(n)} \) is the iteration error defined for the \( n^{th} \) iteration by

\[
d^{(n)} = u^{(n)} - u^{(n-1)}. \tag{3}
\]

Hence, the solution process actually yields the spectral radius of the iteration matrix via the power method. Values of \( \lambda_1 \) begin to stabilize (converge) as the asymptotic convergence rate is approached, and we terminate iterations in steps 2, 3 and 4 based on this.

**Remark 3.** The red-black ordering of the DDM subdomains is not actually necessary, but it does reduce the amount of shared storage required by separate processors. Furthermore, on systems with only a small number of processors this provides a well-defined way to assign them to subdomains. Clearly, load balancing considerations will dictate that these subdomains be of similar sizes.

**Remark 4.** We also comment here that subdomain solves produce an effect on convergence rate similar to that of coarse-grid corrections in the multigrid context. In particular, recall that the the spectral radius of, e.g., the Jacobi iteration matrix, \( \rho(\mathbf{B}) \), corresponding to a second-order centered discretization of the Dirichlet problem for Laplace’s equation on \( \Omega \equiv (0,a) \times (0,b) \) is

\[
\rho(\mathbf{B}) = \frac{1}{2} \left( \cos \frac{\pi h}{a} + \cos \frac{\pi h}{b} \right) \tag{4}
\]

\[
\simeq 1 - \frac{1}{4} \left[ \left( \frac{\pi h}{a} \right)^2 + \left( \frac{\pi h}{b} \right)^2 \right],
\]

where \( h = h_x = h_y \sim \mathcal{O}(N^{-1/2}) \) with \( N = N_x \times N_y \), the total number of grid points. It is clear that reducing \( a \) and/or \( b \) (i.e., restricting calculations to subdomains) has a similar effect to increasing \( h \), the grid spacing (i.e., coarsening the grid).

**Remark 5.** The method employed to compute the approximate Schur complement not only updates subdomain boundary values in our nonoverlapping decomposition, but also provides *effective* overlap. Thus, the overall method achieves multiplicative Schwarz con-
vergence rates even though it is formally an additive Schwarz method.

**Remark 6.** We have employed RB-ordered SLOR as our basic solver throughout this algorithm. This can be replaced with one's favorite elliptic solver, but we have in the past found this method to be very robust (even for systems that are neither symmetric nor positive definite) and easily parallelizable.

We also note that of necessity non-parallelized versions of SLOR were employed within the subdomain solves, which themselves are done in parallel, because current hardware seems unable to handle multilevel parallelization in practice, especially in a multi-user environment, despite its theoretical attractiveness.

Two additional items associated with this algorithm deserve comment. First, it should be noted that one of the main shortcomings of FMG in the context of N.-S. solutions is that the coarse-grid solutions tend to be highly underresolved, and the solutions (even on the fine-grid) are not necessarily smooth. It is then unclear whether anything will actually be gained by prolongating nonsmooth, extremely inaccurate solutions to the fine grid. Except during the initiation phase, our algorithm avoids this difficulty completely because once the domain decomposition loop (steps 4 through 7 of the algorithm) is begun all calculations are performed at the resolution of the fine grid. Second, it is also important to note that our overall goal is to obtain a method for solving elliptic PDEs for which run times (i.e., wall-clock times) are nearly independent of problem size. For this to be possible on current machine architectures (with considerable communication overhead), it is essential that the underlying method be able to compute solutions in \( \mathcal{O}(N) \) arithmetic operations, where \( N \) is the total number of grid points.

Then if the problem size increases by, say a factor of four, only a factor of four increase in number of processors should be required—which is to some extent possible on current SMP-type machines. In the context of our implementation, this leads to the requirement that subdomain sizes and number of subdomains per processor remain constant as the problem size increases. If the algorithm is performing as we would hope, then in this scenario, run times should be nearly independent of problem size.

### 3. Test Problem

To test the performance of the algorithm described in the preceding section we employ a basic 2-D elliptic Dirichlet problem

\[
-\Delta u = f, \quad (x, y) \in \Omega \equiv (0, 1) \times (0, 1),
\]

with

\[
u(x, y) = 0, \quad (x, y) \in \partial \Omega.
\]

We have chosen the function \( f \) to be

\[
f(x, y) = 13\pi^2 \sin 3\pi x \sin 2\pi y.
\]
Then the exact solution to Prob. (1) is

\[ u(x, y) = \sin 3\pi x \sin 2\pi y, \quad (x, y) \in \Omega, \]

shown in Fig. 2. This solution is \( C^\infty \) but sufficiently complicated to provide a nontrivial model.

![Figure 2. Exact solution to model problem.](image_url)

4. RESULTS

Figures 3 and 4 display results obtained for the above problem. Figure 3 shows that the required “nearly” linear increase in run time with problem size is not being exceeded, even for very large problems. The data points shown were obtained in single-processor runs in non-dedicated (hence, realistic practical conditions) mode on a HP SPP-2200 at the University of Kentucky Computing Center. However, it should be noted that these results do not represent fully-converged solutions, but instead are for a fixed decrease in residual. It is also worth noting that despite the increase in overall algorithmic complexity, compared with the base SLOR solver alone (applied to the fine-grid problem), total run times are approximately 33% faster for the complete algorithm due to improved convergence rate of the multilevel DDM combination.

Parallelization was performed with OpenMP compiler directives (shared-memory mode) on the HP SPP-2200 and HP N-4000 at the University of Kentucky Computing Center. The former was a 64-processor machine that has now been replaced by the latter, which is
a 96-processor SMP. For this part of the study Problem (1), discretized using a standard second-order, centered-difference approximation, was solved on fine grids ranging in size from $101 \times 101$ to $401 \times 401$ points. The domain decomposition aspect of the algorithm employed $101 \times 101$ non-overlapping subdomains of the fine grid. The multilevel portion of the scheme utilized a $9 \times 9$ coarse grid, an $81 \times 81$ intermediate grid and the aforementioned $401 \times 401$ fine grid, with analogous grid-size ratios used for the $101 \times 101$ and $201 \times 201$ grids. Actual speedups for fixed multilevel grid arrangements are displayed in Fig. 4, for both SPP-2200 and N-4000 parallelizations. Results from the N-4000 show very good scaling through four processors, and continue to exhibit fairly good speedups even through eight processors. In fact, one sees essentially the same slope ($\approx 1.0$) in speedup even between five and eight processors, but with actual speedup shifted (lower). The precise reason for this shift in these data is not currently known.

![Figure 3](image3.png)  
**Figure 3.** Run time vs. number of grid points.

![Figure 4](image4.png)  
**Figure 4.** Speedup vs. number of processors.

With the ratio $\rho_{s/p}$ of number of subdomains $n_s$ to number of processors $n_p$ fixed, we obtained the following results in dedicated runs on the SPP-2200. For $\rho_{s/p} = 1$ the run time increased by only a factor 1.556 in going from a $101 \times 101$ to a $201 \times 201$ grid (factor of four increase in total number of grid points). When $\rho_{s/p} = 2$, using $201 \times 201$ and $401 \times 401$ grids, the factor was 0.885, and for $\rho_{s/p} = 4$ using these same grids led to a factor of 0.685. On the HP N-4000 with $\rho_{s/p} = 2$, the factor is 1.187, again employing dedicated runs. A value of 1.0 for this factor implies run times independent of problem size. Hence, we conclude that rather generally, the goal of run times independent of problem size is being achieved, at least in an average sense. Moreover, this is being accomplished with a very simple widely-used (in the context of commercial CFD codes) algorithm, SLOR.
5. SUMMARY AND CONCLUSIONS

In this paper we have presented a new implementation of the well-known multilevel DDM concept. The main advantage of the present method is its overall simplicity, being based on successive line overrelaxation, and thus ease of parallelization. We have tested the algorithm with a model Dirichlet problem having known exact $C^\infty$ (but nontrivial) solution and demonstrated good parallel speedups in addition to nearly $O(N)$ total arithmetic scaling with problem size. This combination leads to run times essentially independent of problem size when parallelization is applied.

Continuing efforts with this algorithm will address problems in generalized coordinates, non-Dichlet boundary conditions and further parallelization to move the method toward applicability in actual N.-S. equation solution procedures.

ACKNOWLEDGEMENTS

Support of the second author (JMM) by AFOSR Grant #F49620-00-1-0258 and NASA/EPSCoR Grant #WKU-522635-00-10 is gratefully acknowledged.

REFERENCES