CHAPTER 5

A Domain Decomposition Method with Locally Uniform Mesh Refinement*

William Gropp†
David E. Keyes**

Abstract. We describe an iterative algorithm based on domain decomposition for implicit linear systems arising from partial differential equation problems which require local mesh refinement. In order to keep data structures as simple as possible for parallel computing applications, the fundamental computational unit in the algorithm is a subregion of the domain spanned by a locally uniform tensor-product grid, suggestively called a tile. This is in contrast to local refinement techniques whose fundamental computational unit is a grid at a given level of refinement. The bookkeeping requirements of such algorithms are potentially substantial, since consistency of data must be enforced at points of space which may belong to grids at several different levels, and furthermore, the grids are not necessarily of tensor-product type, but more generally unions thereof. The tile-based domain decomposition approach condenses the number of levels in consideration at each point of the domain to two: a global coarse grid defined by tile vertices only and a local fine grid, where the degree of resolution of the fine grid can vary from tile to tile. Experimentally, we show in [13] that one global level and one local level provide sufficient flexibility to handle a diverse collection of two-dimensional problems which include irregular regions, non-simply-connected regions, non-self-adjoint operators, mixed boundary conditions, non-smooth coefficients, or non-smooth solutions. We employ up to $\mathcal{O}(10^3)$ tiles on problems containing up to $\mathcal{O}(10^4)$ degrees of freedom. This contribution focuses specifically on the robustness of domain decomposition with respect to local refinement.

1. Introduction. The combination of domain decomposition with preconditioned iterative methods provides a framework which extends the usefulness of numerical techniques for certain special partial differential equation problems to those of more general structure. Non-smooth features, non-separable geometries, or massive sizes of practical

*The work of the first author was supported in part by the Office of Naval Research under contract N00014-86-K-0310 and the National Science Foundation under contract number DCR8521451. The work of the second author was supported in part by the National Science Foundation under contract number EET-8707109.

†Department of Computer Science, Yale University, New Haven, CT 06520.

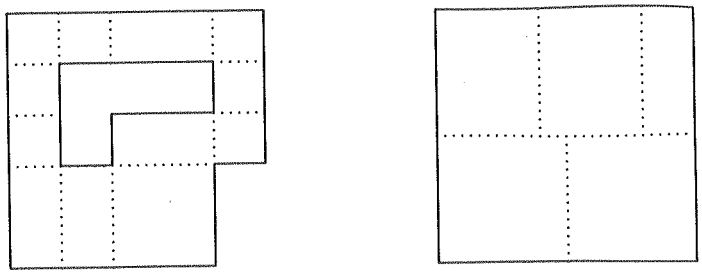
**Department of Mechanical Engineering, Yale University, New Haven, CT 06520.

problems limit the application of many "standard" numerical techniques. Direct methods are rapidly defeated by problem size. "Fast" methods which take advantage of special coefficient and grid structure often do not apply globally. Iterative methods often depend for efficient implementation on regular grids which, if global in extent, are inconsistent with accurate and economical resolution of the physics of the problem. However, the domains of problems with these features can often be decomposed into smaller subdomains of simpler structure, increasing the utility of extant software libraries, particularly as components of preconditioners. Moreover, the domain decomposition can be made to produce a transparent mapping of many problems onto medium-scale parallel computers. Our primary focus in this paper is the incorporation of spatially-varying mesh refinement requirements into a finite-difference-based domain decomposition algorithm. We illustrate the convergence behavior of the algorithm on a set of two-dimensional elliptic PDE problems, including non-self-adjoint, non-separable geometry cases. We also point out features of the method which are relevant to a parallel implementation but defer the corresponding complexity analysis to a subsequent paper.

Many PDE problems which are "large" in the discrete sense are so because the continuous problems from which they are generated require resolution of several different length scales for the production of a meaningful solution. The value of compromising between the extremes of globally uniform refinement, which leads to simple and usually vectorizable algorithms but wastes time and memory, and pointwise adaptive refinement, which minimizes the discrete problem size for a given accuracy requirement but leads to complicated data structures, has been recognized for some time and described in contexts too numerous to mention. Locally Uniform Mesh Refinement (LUMR) characterizes one such class of discretizations, based on composites of highly structured subgrids. Many treatments of LUMR in the literature pertain to explicit methods for transient problems, a class with its own advantages (see [1] and references therein) and limitations [21] which is somewhat distinct from ours. Implicit treatments of LUMR for elliptic problems include approaches arising out of classical multigrid (see [17] and references therein), a nonconforming spectral technique [16], and methods rooted in iterative substructuring for finite element problems [2].

Computationally practical locally uniform grids are usually expressible as the union of a coarse uniform tensor-product grid covering the entire domain with one or more refined tensor-product grids defined over subregions, including the possibility of multiple, nested levels. Generalizations of this within the LUMR framework include allowing the grids at any particular level of refinement to themselves be the union of tensor-product subgrids, and reinterpreting "uniform" as "quasi-uniform" to allow general curvilinear coordinates for custom body- or solution-fitting. We select for consideration a rather restricted form of LUMR in which refinement occurs exclusively within complete cells of a quasi-uniform coarse grid, as described in section 2 below.

The goal of the present contribution is an LUMR methodology which is simple enough for efficient portability to a variety of parallel machines. It borrows from the mesh refinement and domain decomposition literature and from the authors' own experience in these areas and in parallel computation [10, 12, 15]. In our pursuit of convenience and overall parallel performance, in which we include both absolute speedup and efficiency, we are ready, potentially, to compromise "optimality" as defined by conventional serial computing measures. For example, by refining only in units of full coarse grid cells, we may impose a tendency towards refinement in regions where it would be unnecessary from a truncation error point of view alone. As another example, our convergence rate is dependent upon a coarse grid resolution which may be chosen with criteria beyond convergence rate in view, such as the balance of work among multiple processors. Fortunately, the methodology



(a) (b)

Figure 1: Sample Tessellations. (a) is permissible, (b) is not.

survives such compromises and is even sequentially advantageous in many problems (see [13]).

The domain decomposition algorithms we employ (section 3) involve "nearly" parallel preconditioners in conjunction with generalized minimum residual (GMRES) iteration, a non-stationary method not dependent upon operator symmetry. In two dimensions, the preconditioner involves three separate phases: a global coarse grid solve, independent solves along interfaces between subdomains, and independent solves in the subdomain interiors. The global coarse grid solve, which we do directly, is an essential feature as it provides the only global exchange of information in the preconditioner itself.

We show the results of some numerical experiments on two-dimensional elliptic boundary value problems in section 4. We use up to $\mathcal{O}(10^3)$ coarse grid elements on problems containing up to $\mathcal{O}(10^4)$ degrees of freedom.

2. Mesh refinement by tiles. In this section we describe a simple mesh refinement philosophy based on a regular tessellation of the global domain into subdomains which we call "tiles" in two dimensions. Mathematically, a tile is the tensor-product of half-open intervals in each coordinate direction, except that a tile abutting a physical boundary along what would ordinarily be one of its open edges is closed along that edge. Each tile possesses its own tensor-product discretized interior, two of its four sides, and one of its four corners. Although the specific convention is arbitrary, we assume for definiteness that in its own local right-handed coordinate system, each tile contains its origin and its x and y axes.

In contrast to physical boundary segments, we refer to the artificial decomposition-induced boundaries of the tiles as "interfaces". We refer to the points at the intersection of all boundaries, physical or artificial, as "cross-points". We require that the cross-points be embeddable in a tensor-product global quasi-uniform coarse grid, from which only points lying exterior to the (possibly multiply connected) boundary are missing. This rules out irregular tiling patterns such as in Figure 1b. However, there is no requirement that the domain itself be of tensor-product type; the decomposition in Figure 1a is permissible.

Associated with each tile is the data defined over a quasi-uniform grid covering its portion of the domain and a set of operators for executing its block-row portions of the

preconditioner solve to be described later. These operators can potentially be of different type for different tiles. For computational convenience, we assume throughout that the grids covering individual tiles are derived from the coarse grid of cross-points through refinement in ratios of powers of two. We can therefore indicate refinement levels using the graphical shorthand of Figure 5 where the integer indicates the logarithm of the refinement ratio.

2.1. Tile-tile interfaces. In order to minimize restrictions on the structure of adjacent tiles (and to eliminate redundant communication between tiles in a multiprocessor implementation, in which different tiles might be assigned to different processors), each tile stores and maintains, in addition to its own data, the data associated with a buffer region of phantom points equal in width to one-half of that of its associated finite difference stencil. Excluding the redundant phantom points, each point of the domain is uniquely associated with a single tile.

Data at the phantom points is supplied in a manner dependent upon the internal structure and refinement ratios of the adjacent tiles in question. A finer tile obtains bi-quadratically interpolated data from its coarser neighbor. Since the problems studied herein involve second-order operators, this allows the use of conventional finite difference techniques in generating the difference equations at the subdomain interfaces.

All of our examples employ strictly uniform local grids. The coarse tiles obtain their data by simple (unweighted) injection. That is, the value at the point in the finer neighboring tile that lies on the coarse grid stencil is used for the coarse grid point. If operator symmetry was important, other methods, such as a weighted averaging or a finite element discretization with transition elements along the interface, could be used.

The selection of general refinement criteria is a much studied, yet still open problem. However, it is orthogonal to the equation-solving aspect considered herein, except to the extent that computational work required by one of these tasks is a by-product of the other. In our examples, "good" refinement strategies are obvious enough to be done "by hand".

In general, tile interfaces can be the site of changes in the discretization besides just the refinement level. For instance, the discrete stencil can change order at interfaces. Even the form of the operators or their number can change at interfaces while still preserving the subdomain uniformity required for efficient subdomain solution algorithms. As a motivational example, a reacting flow problem frequently consists of large regions in which there is only transport of mass, momentum, and thermal energy but no reaction among constituents of known composition, to all adequate orders of approximation. In other regions it is essential to retain composition variables, because they diffuse differentially, and in a subset of these, reaction terms must also be retained in the equations. To accommodate such generality, the routines that pack the buffer regions are responsible for providing the necessary mappings.

2.2. Physical boundaries. For generality, the equations for the physical boundaries are incorporated into the overall system matrix, including Dirichlet conditions. Our implementation allows inhomogeneous Robin boundary conditions at all boundary points, though the examples in this paper all use Dirichlet boundary conditions.

2.3. Comparison with other approaches. In contrast to multi-level approaches in which the fundamental computational unit is a grid at a given level, our fundamental computational unit is a subregion of the domain. The present approach requires only one grid which possesses connectivity with arbitrarily distant regions of the domain, namely the coarsest one. In the language of the hierarchical basis function technique [25], we have simply a two-level hierarchy, but the level of the higher may be different in different

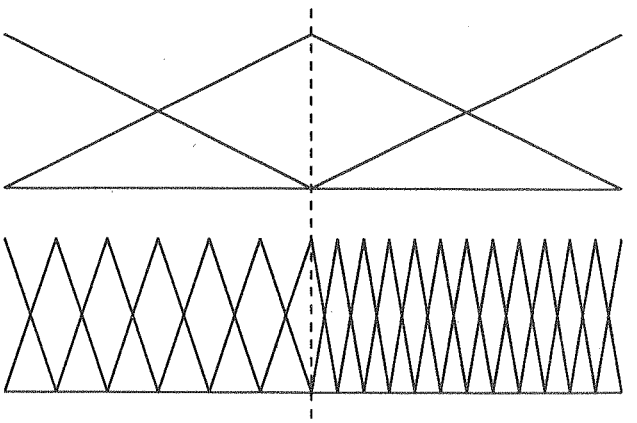


Figure 2: One-dimensional schematic of the tile basis functions.

subregions. Figure 2 gives a one-dimensional illustration. This represents a severe condensation of the range of intermediate scales present in multi-level local uniform refinement, on which much of the convergence theory is based. Tiles are much closer to being the software equivalent of the "geometry-defining processors" (GDPs) of Dewey and Patera [7].

The tile approach is also similar to the additive Schwartz method [8, 23] and the techniques of [3] in its reliance upon just a single domain-spanning grid. The main difference between these techniques and the tile approach is in the treatment of the interfacial degrees of freedom. In the additive Schwarz technique, interior problems are solved on extended overlapped subdomains, of which the interfacial degrees of freedom are interior points and thus demand no special consideration. In [3], good preconditioners for the interfacial degrees of freedom are derived theoretically, for self-adjoint operators. Optimal algebraic convergence (independent of degree of refinement) has been proved for both classes of algorithms in [9] and [2], so there are, intuitively, grounds for optimism about single global-grid algorithms even though we present no extensions of the theory to the non-self-adjoint problems we consider. The main disadvantage in condensing out intermediate scales is that the coarse grid, on which all optimal approaches require an exact solve, cannot necessarily become as coarse as one might like.

The field of locally uniform mesh refinement is spanned by a continuum of resolution strategies governed by clustering rules which control the size and shape of the refined subregions. Global refinement lies at one extreme and pointwise adaptive refinement at the other. As soon as the global tensor product mesh is abandoned a host of difficult practical decisions need to be made about data structures and clustering algorithms. The logic required to handle the numerous types of subgrid-subgrid interactions which can arise and to insure the consistency of the data structure is a significant impediment to efficient parallelism. In contrast, "horizontal" neighbor-neighbor interactions are simple. The sufficiency of a two-level approach in obtaining reasonable convergence is demonstrated in section 4. Compelling superiority of approaches with a greater richness of scales has not

yet been fully established in production parallel software, although it may be ultimately.

3. Iterative domain decomposition algorithms. As mentioned in the introduction, preconditioned iterative methods and domain decomposition provide a framework suitable for the description of a wide class of algorithms. The four common elements of this framework are: a global operator arising from the discretization of the PDE (or system of PDEs); an approximate inverse, or preconditioner, for the global operator; an iterative method relying only on repeated application of the preconditioned operator; and a geometry-based partition of the discrete unknowns so that size, locality, and uniformity can be exploited in applying the preconditioned operator. Since the numerical analysis literature contains many successful discretization schemes and iterative methods specialized for different operator properties, such as the presence or absence of definiteness and symmetry, the recent burgeoning effort in iterative domain decomposition algorithms has concentrated primarily (though not exclusively) on the interaction of the second and fourth of these elements. In the parallel context, this is a natural preoccupation because the bottleneck to parallelism usually (though not exclusively) lies in the requirement of the global transport of information in the preconditioner.

Many of the numerical examples described in section 4 rule out the use of iterative methods based on symmetry, but permit the assumptions of definiteness and diagonal-dominance. In particular, full or incomplete factorizations of subdomain matrices can be undertaken without pivoting. Because of its robustness, we join many recent users [6, 18, 20, 24] in adopting the parameter-free generalized minimum residual (GMRES) method [19] as the outer iteration. The main disadvantages of GMRES, its linear and quadratic (in iteration index) memory and execution time requirements, respectively, must be mitigated by scaling and preconditioning. The primary type of decomposition used herein involves roughly unit aspect ratio tiles, as opposed to thin strips. Ordering the interior points (including the physical boundary points other than cross-points) first, the cross-points last, and the interfaces connecting the cross-points in between, gives a nested-dissection-like "arrow" matrix appearance to the global discrete operator, which we denote A. The basic structure of our preconditioner B is the block-upper triangular portion of the arrow matrix. The application of B^{-1} thus begins with a cross-point solve, which updates the right-hand sides of a set of independent interface solves. These, in turn, update the right-hand sides of a set of interior solves. For a nine-point stencil, the cross-point result would also update the interior right-hand sides. However, there is no dependence, within a single iteration, of the interface solution upon the result of the interior solution, or of the cross-point solution upon either. (In [4], structurally symmetric arrow matrix preconditioners were compared against the corresponding triangular forms on a variety of strip-wise decomposed problems. It was found therein that retaining the interior-to-interface coupling in the preconditioner generally reduced the total number of iterations required to attain a fixed convergence criterion, but that the execution time of the structurally symmetric algorithm was greater, because of the cost of the extra set of subdomain solves in each iteration. The first and second sets of subdomain solves are inherently sequential.)

The derivation of the coefficients of the preconditioner blocks is as follows. The cross-point equations are simply a coarse grid discretization of the continuous PDE. Physical boundary points lying at tile corners are retained in the cross-point system in order to accommodate first-order Neumann or mixed conditions in this coarse grid discretization. Weighted averaging possibilities for the derivation of the coarse grid operator arise from the possession of the coefficients and right-hand side on finer grids surrounding each cross-point, but these are not currently exploited. The current implementation supports LU-based Gaussian elimination on the coarse-grid system. This solve is the chief parallel bottleneck in the preconditioner and can be performed in either of two ways: redundantly

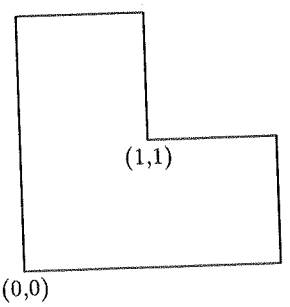


Figure 3: The domain considered in this paper.

in each tile after broadcasting coefficient data for small systems, or in a fully distributed fashion for large systems. Determination of the most efficient technique is generally domain and network dependent. If strip decompositions are used, there is no cross-point system, and the lower-right block of the preconditioner is the interface system described below.

The tile interior equations consist of fine grid discretizations of the PDE over local regions, with physically appropriate boundary conditions along any true boundary segments and Dirichlet boundary conditions at artificial interfaces. Only first-order differences are accommodated in the physical boundary conditions of the preconditioner, even if higher-order are employed in the operator A. The current implementation supports full LU Gaussian elimination. Each tile performs its interior solve completely independently.

Unlike the coarse grid and tile interior equations, which bear the physical dimension of the underlying PDE and have natural preconditionings, the lower-dimensional interfacial equations are properly derived from a related pseudo-differential operator, an approach we do not pursue here. Instead, we have used a preconditioner we call "tangential". This interface preconditioner is the one-dimensional discretization of the terms of the underlying operator which remain when the derivatives normal to the interface are set to zero. Alternative methods are described and evaluated in [13].

4. Numerical experiments. The numerical experiments of this section serve to illustrate the effectiveness of the domain decomposition methods employed in terms of the convergence of the iterations and also the effectiveness of the locally uniform mesh refinement in terms of the convergence of the discretization.

4.1. Model problems. We present three model problems, each containing a single dependent variable and two independent variables. One of the problems below is self-adjoint and could be discretized in a symmetric manner and perhaps solved more cheaply with conjugate gradients than with GMRES. Our main interest, however, is in the more extensible formulation. In the examples to follow, an exact solution of the continuous problem $\mathcal{L}u = 0$ is specified. From this u, the boundary condition inhomogeneities may be calculated. The domain used is pictured in Figure 3.

The three examples are obtained by taking three different values of the convection, respectively $c = 0$, $c = -10$, and $c = 1$, in the convection-diffusion problem below.

Problems #1–3: Cylindrically-separable reentrant corner convection-diffusion problem.

$$-\nabla^2 u + \frac{c}{r}\frac{\partial u}{\partial r} = 0$$

$$u(x,y) = r^\alpha \sin\left(\frac{2}{3}\left(\theta - \frac{1}{2}\pi\right)\right)$$

$$\text{where } r = \sqrt{(x-1)^2 + (y-1)^2}$$

$$\text{and } \theta = \arg((x-1) + i(y-1)),\ 0 \le \theta < 2\pi$$

$$\text{Dirichlet data on } \partial\Omega$$

$$\Omega = \text{L-shaped region}$$

The first of these corresponds to pure diffusion, and the second and third to convection in towards the reentrant corner, and away from it respectively, at a rate inversely proportional to radius. The respective values of the radial eigenfunction exponent α are $\frac{2}{3}$, $\frac{1}{3}$, and approximately 10.0442, from the Euler equation formula $\alpha = \left[c + \sqrt{c^2 + \frac{16}{9}}\right]/2$. The first two solutions of this trio lack derivatives at the reentrant corner. The last is everywhere twice-differentiable, but the solution is characterized by steep variation in the three *non*-reentrant corner regions, where $r > 1$. Local mesh refinement is critical to improving the accuracy of a finite-difference solution.

Perspective surface plots of the solutions to these three problems are given in Figure 4.

4.2. Parameters studied. Several categories of experiments are reported. First, a two-dimensional parameter space consisting of coarse grid resolution and overall (uniform) resolution is explored by numerical experiment. A non-restarted GMRES algorithm is used, block-triangularly preconditioned with exact solves on the subdomain interiors and on the coarse grid, and with tangential interface solves. The goal of these experiments is the evaluation of the algorithm over a range of resolutions, in terms of iteration count and execution time, for comparison with back-of-the-envelope complexity analyses.

Another set of experiments is performed on the problems with the goal of evaluating the economy of the local refinement technique. We show that local uniform mesh refinement is capable of significant CPU and memory savings with no sacrifice of accuracy relative to uniform refinement.

The timings given below are from a Multiflow Trace 14/200 computer using 64-bit reals. All of the code (primarily in C but with FORTRAN library routines) was compiled with the default (-O3) optimization and version 2.1.3 of the compilers.

4.3. Convergence as a function of coarse grid granularity. In order to test coarse grid granularity over a large range, we fix the finest mesh spacing at $h^{-1} = 128$ (dimensionless), and investigate the tradeoff between numbers of tiles and points per tile, as shown in Tables 1 and 2. The mesh is identical and locally uniform for all runs in these tables. The convergence criterion is a relative reduction in residual of five orders of magnitude. Table 1 shows that the iteration count peaks in the middle of the granularity range, at either 4 or 8 tiles per side. The bottom row of all 1's can be supplied without benefit of actual experiments, since it represents a direct solve on a single grid.

Table 2 shows the deceptiveness of iteration count as a measure of overall performance. In execution time, the limiting case suffers due to the high cost per iteration, even though the number of iterations required is very small. This table is a profound illustration of the title of [5]: *Domain Decomposition Beneficial Even Sequentially.* The most favorable total *sequential* execution times are found for multi-domain cases near the iteration count maxima, in particular at 16 tiles per side.

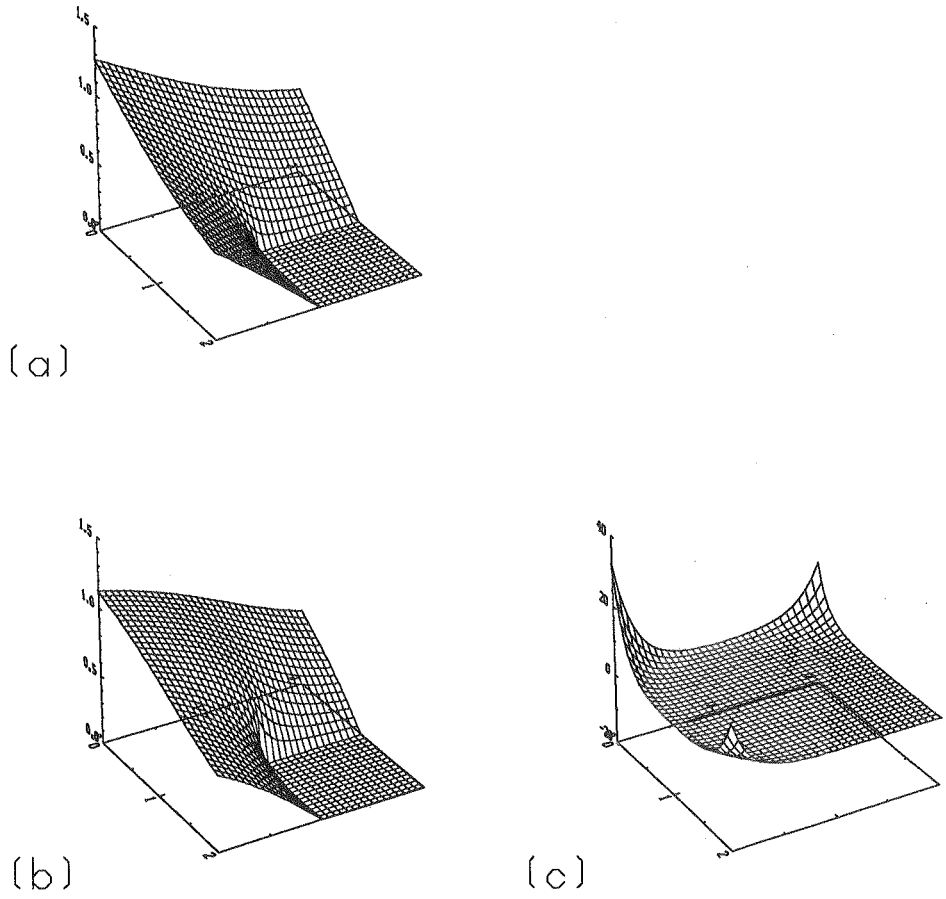


Figure 4: Surface plots of the test problem solutions: (a) #1, (b) #2, (c) #3.

t	m		#1	#2	#3
2	64		12	11	4
4	32		17	16	12
8	16		23	22	18
16	8		16	19	16
32	4		11	12	10
64	2		-	-	-
128	1		1	1	1

Table 1: Iteration count as a function of number of tiles per side of circumscribing square, t, and number of mesh points along a tile side, m, at constant refinement parameter, $h^{-1} = 128$, for a reduction in the initial residual of 10^{-5}. The last two lines of the table are not available experimentally due to minimum discrete subdomain size conventions in the code; however, the last line consists of all 1's by definition, when $t = h^{-1}$.

t	m		#1	#2	#3
2	64		84	85	81
4	32		25	25	23
8	16		14	14	12
16	8		9	14	12
32	4		36	21	17

Table 2: Execution time (sec) as a function of number of tiles per unit length, t, and number of mesh points along a tile side, m, at constant $h^{-1} = 128$, for a reduction in the initial residual of 10^{-5}.

t	h^{-1}		#1	#2	#3
2	16		6	5	3
4	32		12	11	11
8	64		17	14	16
16	128		16	13	16

Table 3: Iteration count as a function of number of tiles per side of circumscribing square, t, and refinement parameter, h^{-1}, at constant number of mesh points along a tile side, $m = 8$, for a reduction in the initial residual of 10^{-5}.

The behavior in Table 2 can be understood with reference to back-of-the-envelope complexity estimates for the solution and factorization operators of the preconditioner. We observe that there are $O(t^2)$ cross-point, interfaces, and interiors. Naturally ordered banded direct factorizations and solves require $O(Nb^2)$ and $O(Nb)$ operators respectively, where N is the number of unknowns and b the bandwidth. For the cross-point system, $N \approx t^2$ and $b \approx t$; for the interfaces, $N = m$ and $b = 1$; and for the subdomain interiors, $N = m^2$ and $b = m$. Thus, the interface operation counts are always asymptotically subdominant and can be omitted in the following. From choosing the larger of the cross-point and interior complexities, we see that factorization costs $\max_{(t,m)}\{O(t^4), O(t^2m^4)\}$ and solves cost $\max_{(t,m)}\{O(t^3), O(t^2m^3)\}$. Since $m = 128/t$ in these experiments, the first term grows with t and the second decays with it. Quick calculations reveal that (to the resolution of the table) the minima for both factorization and solve costs occur at or between $t = 16$ and 32 when $h^{-1} = 128$. The tendency of buffer overhead, neglected in the estimates, is to favor a slightly smaller number of tiles t than thus estimated. It is important to note that the memory requirements follow the solve complexities above. Thus, for a fixed memory size, an intermediate coarse grid granularity accommodates the largest problem in core. Of course, all of these per iteration complexity estimates need to be redone when the preconditioner blocks are other than exact solves, for instance, incomplete factorizations. However, incomplete and exact factorizations differ little in actual cost per iteration when the grid is narrow enough in the rapidly ordered direction, which includes the case of small, square tiles.

4.4. Convergence as a function of tile refinement. In contrast to the previous section, we here investigate iteration count as a function of overall resolution, for a fixed number of subintervals per tile. The results are shown in Table 3. The global mesh grows in refinement from 16 to 128 as the number of points per tile remains constant at 8. In spite of the fact that the truncation error improves roughly with h^{-1}, we use the same convergence tolerance of 10^{-5} as in the earlier tables. The fine grid in the last line of Table 3 corresponds to the $t = 16$ case of the earlier tables.

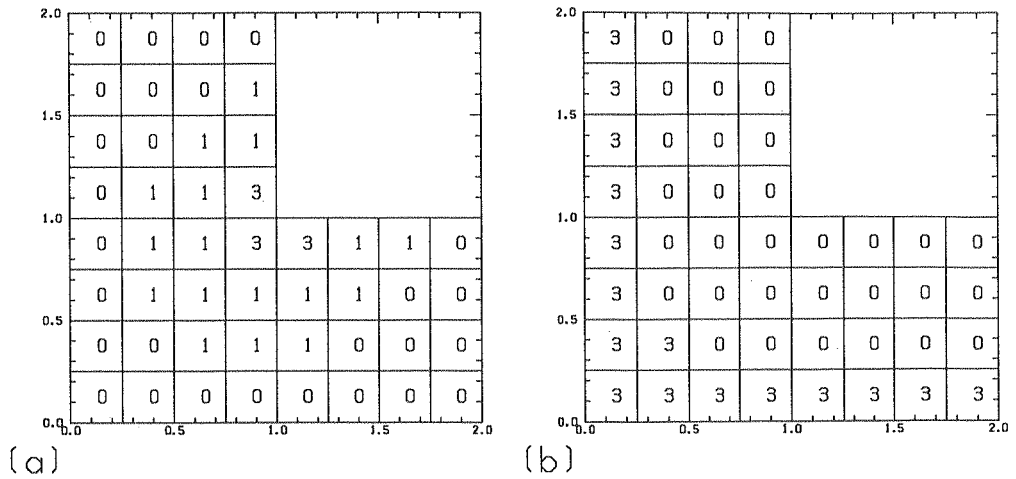


Figure 5: Refinement levels. The maximum (third level) local uniform refinements employed on problem #1 and #2, and #3. are shown in (a) and (b), respectively. In second level tests, all tiles showing "3" are set to "2". In first level tests, these are further reduced to "1". In zeroth level refinement, all tiles are set to "0", which here corresponds to $m = 4$.

The experiments suggest that the iteration count is bounded nearly independently of h, and thus that the two-level algorithm is nearly optimal asymptotically in the constant m limit. In fact, two of the finest mesh results are even relatively *better* than preceding coarser ones. This should not be regarded as surprising, since there is a steep price for this favorable iteration count when m is held constant and h^{-1} is increased, namely, a larger cross-point system. We have not pursued any theoretical justification for this bound, but the theory for conjugate gradient iteration for self-adjoint problems, see, *e.g.*, [3, 22], contains similar results, namely, constant upper bounds on the iteration count for constant m.

4.5. Economies of local mesh refinement. The examples allow us to display the well-known benefits of local uniform mesh refinement in elliptic problems: comparable accuracy in considerably fewer operations, compared with global uniform refinement. We solve these problems at refinement levels of $h^{-1} = 32$, 64, 128, and 256, based on the global grid, but perform both global and local refinements for comparison, where possible. (The finest global refinement does not fit into the memory available, which is, of course, another of the main motivations for LUMR, along with execution time savings.) All of these computations were made with a reduction in the residual of 10^{-8}. In all cases, the choice of where to refine is made by hand. Since we are interested in studing how domain decomposition and mesh refinement interact, given a good refinement strategy, we eliminate the latter question from this study.

Tables 4 through 6 compare of global refinement results on the left, and local on the right. Each side lists the number of unknowns, the sup-norm of the error, the number of iterations to reduce the discrete residual by 8 orders of magnitude, and the total execution time thus required. The right-most column gives the execution time ratios for each refinement level. All entries share a constant value of $t = 8$ in order to provide region of enhanced refinement that does not shrink as h does. Therefore, the "global" iteration columns of Tables 4 through 6 comprise a convergence study which is complementary to both Table 1 (in which h is constant) and Table 3 (in which m is constant).

		Global				Local				Ratio
h^{-1}	m	N_G	e_G	I_G	T_G	N_L	e_L	I_L	T_L	T_G/T_L
32	4	834	1.30(-2)	24	2.7	834	1.30(-2)	24	2.7	1.00
64	8	3202	8.30(-3)	32	6.8	1818	8.30(-3)	35	6.2	1.10
128	16	12546	5.25(-3)	41	27.1	2410	5.26(-3)	37	7.6	3.57
256	32		NA	NA	NA	4746	3.33(-3)	41	16.4	NA

Table 4: Number of unknowns N, sup-norm of the error e, iteration count I, and execution time T (sec) for problem #1 (reentrant corner, pure diffusion), globally and locally refined, along with execution time ratios, for a reduction in the initial residual of 10^{-8}.

		Global				Local				Ratio
h^{-1}	m	N_G	e_G	I_G	T_G	N_L	e_L	I_L	T_L	T_G/T_L
32	4	834	6.97(-2)	23	2.6	834	6.97(-2)	23	2.6	1.00
64	8	3202	5.65(-2)	37	8.2	1818	5.66(-2)	34	5.7	1.44
128	16	12546	4.53(-2)	40	26.1	2410	4.58(-2)	37	7.6	3.43
256	32		NA	NA	NA	4746	3.67(-2)	41	16.5	NA

Table 5: Number of unknowns N, sup-norm of the error e, iteration count I, and execution time T (sec) for problem #2 (reentrant corner, convective inflow), globally and locally refined, along with execution time ratios, for a reduction in the initial residual of 10^{-8}.

		Global				Local				Ratio
h^{-1}	m	N_G	e_G	I_G	T_G	N_L	e_L	I_L	T_L	T_G/T_L
32	4	834	7.35(-1)	22	2.4	834	7.35(-1)	22	2.4	1.00
64	8	3202	4.15(-1)	28	5.7	1610	4.30(-1)	25	3.6	1.58
128	16	12546	2.19(-1)	34	21.5	4698	2.40(-1)	29	8.5	2.53
256	32		NA	NA	NA	17018	1.98(-1)	35	51.6	NA

Table 6: Number of unknowns N, sup-norm of the error e, iteration count I, and execution time T (sec) for problem #3 (reentrant corner, convective outflow), globally and locally refined, along with execution time ratios, for a reduction in the initial residual of 10^{-8}. The error values here appear large, but are in fact small relative to the size of the solution.

The behavior of iteration count with each doubling of global refinement in the self-adjoint problem in Table 4 is consistent with the logarithmic growth in conditioning with h^{-1} proved for self-adjoint problems in [3]. The locally refined examples also worsen in conditioning with h^{-1} when t is held constant, but the CPU time advantage of local refinement increases with h^{-1}, overall.

The sup-norm of the error shows sublinear improvement in h in problems #1 and #2, as one expects with non-differentiable solutions. (A better discretization that fits in with this tile based approach is described in [11].) The first-order accurate treatment of convection in problem #3 leaves its signature as well (the ratio of errors is slightly less than 2 for each reduction of h by 2).

5. Conclusions and future directions. Experiments on a family of problems drawn from a larger set in [13] demonstrate that a two-level domain decomposition algorithm with a single global coarse grid can provide "nearly" optimal convergence and allow a great deal of flexibility in refinement strategy, while also permitting a data structure

amenable to parallel and vector implementations, as summarized in closing below. Although often motivated by parallelization, domain decomposition may also yield runtime and memory use benefits as a sequential programming paradigm. Furthermore, the simple structure of individual blocks of the domain-decomposed preconditioner means that new applications are found for the "standard solvers" in conventional software libraries.

The traditional economies of local uniform mesh refinement can be straightforwardly incorporated into the domain decomposition framework at the price of interface handlers with conditionals for refinement differences between adjacent subdomains. Because of the highly modular nature of a standardized tile-oriented domain decomposition code, custom discretizations for certain classes of singularities may be archived into applications libraries for reuse.

The tile algorithm demonstrated herein in a superscalar mode on a Multiflow computer is amenable to vectorization in either of two ways. The regular operation sequences on the tensor-product subgrid arrays are precisely the type for which vectorizing compilers were conceived. The vector lengths depend on the precise form of solvers used in the preconditioner, but would tend to be rather small for the rows of individual 8×8 or 16×16 tiles found best in the two-dimensional applications above. An alternative form of vectorization can be realized by grouping together all tiles of a given (discrete) size and shape and operating in lock step on corresponding elements in each tile, assuming an identical solver is applied to each. A vector in this approach consists of the i^{th} element from each of the subdomains. Our 8×8 arrays of tiles would be thus be optimal for machines with a vector length of 64.

Parallelization requires careful attention to the load balancer/mapper and also to the coarse grid solve in the preconditioner. Some complexity estimates pertaining to alternative forms of the latter may be found in [14]. The main disadvantage of the two-level algorithm in the parallel context is that the choice of coarse grid granularity is even more of an "over-determined" problem than in serial. Communication cost per iteration and convergence properties potentially inveigh against the lower bounds imposed by domain geometry, solution and coefficient smoothness, and parallel load balance. The key determination for future applications of the tile methodology will be whether this over-determination is consistent in practice. Inasmuch as the examples herein are representative of single-independent variable problems, and parallel communication costs generally comprise a relatively *smaller* proportion of the total work in coupled multi-component problems, there are substantial grounds for optimism that this will be the case.

References

[1] M. J. Berger & J. Oliger, *Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations,* J. Comp. Phys., 53 (1984), pp. 484–512.

[2] J. H. Bramble, R. E. Ewing, J. E. Pasciak & A. H. Schatz, *A Preconditioning Technique for the Efficient Solution of Problems with Local Grid Refinement,* Comp. Meths. Appl. Mech. Eng., 67 (1988), pp. 149–159.

[3] J. H. Bramble, J. E. Pasciak & A. H. Schatz, *The Construction of Preconditioners for Elliptic Problems by Substructuring, I,* Math. Comp., 47 (1986), pp. 103–134.

[4] T. F. Chan & D. E. Keyes, *Interface Preconditionings for Domain-Decomposed Convection-Diffusion Operators,* these proceedings.

[5] T. F. Chan & D. Goovaerts, *Domain Decomposition Beneficial Even Sequentially,* Technical Report 88-18, UCLA Comp. and App. Math., June 1988.

[6] A. Dervieux, L. Fezoui, H. Steve, J. Periaux & B. Stoufflet, Low-Storage Implicit Upwind-FEM Schemes for the Euler Equations, *Proceedings of the 11th International Conference on Numerical Methods in Fluid Dynamics,* Springer, Berlin, 1989, pp. 215–219.

[7] D. Dewey & A. T. Patera, Geometry-Defining Processors for Partial Differential Equations, B. J. Alder ed., *Architectures and Performance of Specialized Computer Systems,* Academic Press, New York, 1988.

[8] M. Dryja, Optimal Iterative Refinement Methods, T. F. Chan, R. Glowinski, J. Periaux & O. Widlund ed., *Second International Symposium on Domain Decomposition Methods,* SIAM, Philadelphia, 1989, pp. 168–172.

[9] M. Dryja & O. B. Widlund, *On the Optimality of an Additive Refinement Method,* 1989. (Prepared for the Fourth Copper Mountain Conference on Multigrid Methods).

[10] W. D. Gropp, *Local Uniform Mesh Refinement for Elliptic Partial Differential Equations,* Technical Report YALE/DCS/RR-278, Yale University, Department of Computer Science, July 1983.

[11] ———, *A Simple Finite Difference Approximation for the Laplacian near Reentrant Corners,* 1990. (In preparation).

[12] W. D. Gropp & D. E. Keyes, Domain Decomposition on Parallel Computers, T. F. Chan, R. Glowinski, J. Periaux & O. Widlund ed., *Second International Symposium on Domain Decomposition Methods,* SIAM, Philadelphia, 1989, pp. 260–268.

[13] ———, *Domain Decomposition with Local Mesh Refinement,* Technical Report YALE/DCS/RR-726, Yale University, Department of Computer Science, August 1989.

[14] ———, *The Parallel Complexity of Tile-based Substructuring of PDEs with Local Mesh Refinement,* 1990. (In preparation).

[15] D. E. Keyes & W. D. Gropp, Domain Decomposition Techniques for Nonsymmetric Systems of Elliptic Boundary Value Problems: Examples from CFD, T. F. Chan, R. Glowinski, J. Periaux & O. Widlund ed., *Second International Symposium on Domain Decomposition Methods,* SIAM, Philadelphia, 1989, pp. 321–339.

[16] Y. Maday, C. Mavriplis & A. T. Patera, Nonconforming Mortar Element Methods: Application to Spectral Discretizations, T. F. Chan, R. Glowinski, J. Periaux & O. Widlund ed., *Second International Symposium on Domain Decomposition Methods,* SIAM, Philadelphia, 1989, pp. 392–418.

[17] S. McCormick & J. Thomas, *The Fast Adaptive Composite Grid (FAC) Method for Elliptic Equations,* Math. Comp., 45 (1986), pp. 439–456.

[18] A. Navarra, *An Application of GMRES to Indefinite Linear Problems in Meteorology,* Comp. Phys. Comm., 53 (1989), pp. 321–327.

[19] Y. Saad & M. H. Schultz, *GMRES: A Generalized Minimum Residual Algorithm for Solving Nonsymmetric Linear Systems,* SIAM J. Sci. Stat. Comp., 7 (1986), pp. 856–869.

[20] F. Shakib, T. J. R. Hughes, Z. Johan, Element-by-Element Algorithms for Nonsymmetric Matrix Problems Arising in Fluids, J. H. Kane ed., *Symposium on the Solution of Super Large Problems in Computational Mechanics,* Plenum, New York, 1989.

[21] B. Swartz, *Courant-Like Conditions Limit Reasonable Mesh Refinement to Order h^2,* SIAM J. Sci. Stat. Comp., 8 (1987), pp. 924–933.

[22] O. B. Widlund, Iterative Substructuring Methods: Algorithms and Theory for Elliptic Problems in the Plane, R. Glowinski, G. H. Golub, G. A. Meurant & J. Periaux ed., *First International Symposium on Domain Decomposition Methods for Partial Differential Equations,* SIAM, Philadelphia, 1988, pp. 113–128.

[23] ————, Optimal Iterative Refinement Methods, T. F. Chan, R. Glowinski, J. Periaux & O. Widlund ed., *Second International Symposium on Domain Decomposition Methods,* SIAM, Philadelphia, 1989, pp. 114–125.

[24] L. B. Wigton, N. J. Yu & D. P. Young, GMRES Acceleration of Computational Fluid Dynamics Codes, *Proceedings of the AIAA 7th Computational Fluid Dynamics Conference, 85-1494,* AIAA, Washington, DC, 1985.

[25] H. Yserentant, *On the Multi-level Splitting of Finite Element Spaces for Indefinite Elliptic Boundary Value Problems,* SIAM J. Num. Anal., 23 (1986), pp. 581–595.