Computing and Visualization in Science manuscript No.
(will be inserted by the editor)

The influence of partitioning on Domain Decomposition convergence rates

Randolph E. Bank · **Chris Deotte**

Received: May 29, 2015 / Accepted: date

Abstract This paper discusses the effects that partitioning has on the convergence rate of Domain Decomposition. When Finite Elements are employed to solve a second order elliptic partial differential equation with strong convection and/or anisotropic diffusion, the shape and alignment of a partition's parts significantly affect the Domain Decomposition convergence rate. Given a PDE, if b is the direction of convection or the prominent direction of anisotropic diffusion, then if one considers traversing the domain in the direction of b, partitions having fewer parts to traverse in this direction converge faster while partitions having more converge slower.

Keywords Domain Decomposition, Bank-Holst Paradigm, Stiffness Matrix Weighting

Mathematics Subject Classification (2010) 65N55,65Y05

1 Introduction

Scientists are interesting in solving the second order elliptic boundary value problem

$$-\nabla\cdot(a\nabla u)+b\cdot\nabla u+cu-f=0 \qquad \text{in } \Omega \tag{1}$$

Bank: The work of this author was supported by the National Science Foundation under contract DMS-1345013.

Deotte: The work of this author was supported by the National Science Foundation under contract DMS-1345013.

Bank: Department of Mathematics, University of California, San Diego, La Jolla, California 92093-0112. Email:rbank@ucsd.edu
Deotte: Department of Mathematics, University of California, San Diego, La Jolla, California 92093-0112. Email:cdeotte@ucsd.edu

with boundary conditions

$$a\nabla u\cdot n=g_N \qquad \text{on } \partial\Omega_N \tag{2}$$
$$u=g_D \qquad \text{on } \partial\Omega_D \tag{3}$$

Here $\Omega\in\mathbb{R}^d$ is a bounded domain, n is the unit normal vector, a is a $d\times d$ spd matrix, b is a vector of length d, and $[a]_{i,j}$, $[b]_i$, c, f, g_N, and g_D are scalar functions on Ω.

A typical procedure for numerically solving (1)-(3) is to convert the PDE into its Weak Form, use a Galerkin Approximation with Finite Elements, partition the domain and solve with a Domain Decomposition method using multiple CPUs. When the underlying PDE is not self-adjoint, upwinding terms are added to improve stability.

Before DD is employed, the domain is partitioned into either overlapping or non-overlapping subdomains. Both types can be created from a partition of disjoint parts with the former having their subdomains enlarged to overlap their neighbors.

A common class of overlapping DD methods is the Schwarz methods [19] [20] [21]. In the case of two overlapping subdomains Ω_1 and Ω_2 where u_i is the solution on subdomain Ω_i and $Lu-f=0$ is our original PDE, the Schwarz methods solve the two local problems

$$Lu_1-f_1=0 \quad \text{on}\,\Omega_1 \tag{4}$$
$$Lu_2-f_2=0 \quad \text{on}\,\Omega_2 \tag{5}$$

with boundary conditions (2) and (3) and interface boundary conditions on one subdomain using information from the other subdomain.

A popular class of non-overlapping DD methods are the Lagrange Multiplier methods [19] [20] [21]. They solve the

saddle point problem

$$Lu_1 - f_1 + \lambda \frac{\partial G}{\partial u_1}(u_1,u_2) = 0 \quad \text{on } \Omega_1 \tag{6}$$

$$Lu_2 - f_2 + \lambda \frac{\partial G}{\partial u_2}(u_1,u_2) = 0 \quad \text{on } \Omega_2 \tag{7}$$

$$G(u_1,u_2) = 0 \tag{8}$$

with boundary conditions (2) and (3) and $G(u_1,u_2) = u_1|_\Gamma - u_2|_\Gamma = 0$ enforces continuity across the interface. After choosing initial guesses for u_1, u_2, and λ, an iterative DD procedure to solve (6)-(8) alternates between solving for $\{u_1,u_2\}$ and updating λ.

Assume T is the triangulation of our domain Ω. Let u_h be our finite element solution belonging to the discrete space S_h formed from a basis of the standard linear Lagrange nodal functions having degrees of freedom at triangle vertices. A standard partition $\{T_1,T_2,...,T_P\}$ of disjoint parts is one that both minimizes the number of degrees of freedom on the interface (triangle edges on interface) and balances the number of degrees of freedom per part (triangles per subdomain).

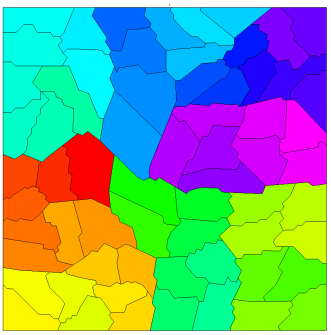


Fig. 1: METIS partitioning the unit square into 64 parts.

Domain Decomposition methods converge well using this standard partition displayed in Figure 1 because work is balanced between processors and dependence between processors is minimized [14] [19]. In 2D (3D), a typical partition consists of parts shaped like circles (spheres) which maintains subdomain area while minimizing subdomain boundary.

This paper is organized as follows. Sections 2, 3, and 4 discuss alternative partitioning schemes. Sections 5 and 6 describe an experimental design to test these schemes while Sections 7 and 8 report the results. Section 9 discusses computational resources. Section 10 concludes the work.

2 Alternate Domain Partitioning Schemes

Let $T = \{t_1,...t_n\}$ be a triangularization of domain Ω. Each triangle t_i and each edge shared by two triangles can be assigned a weight. Define $w(t_i)$ as the weight associated with triangle i and define $W(t_i,t_j) = W(t_j,t_i)$ as the weight associated with the edge shared by triangle i and j. If T_k is a subset of T, define

$$|T_k| \equiv \sum_{t_k \in T_k} w(t_k)$$

Given two subsets T_i and T_j, we define

$$\delta_t(T_i,T_j) \equiv \sum_{\{t_r \in T_i, t_s \in T_j\}} W(t_r,t_s)$$

and given three or more subsets, we define

$$\delta_t(T_1,...,T_P) \equiv \sum_{i=1}^{P-1} \sum_{j=i+1}^{P} \delta_t(T_i,T_j)$$

With this language, finding a partition $\{T_1,T_2,...,T_P\}$ of T can be formally written as

$$\text{Find a partition } \tilde{K}_\varepsilon \text{ such that}$$

$$(1-\varepsilon)\frac{|T|}{P} \le |T_i| \le (1+\varepsilon)\frac{|T|}{P} \quad \text{for } i = 1,...,P \tag{9}$$

$$\delta_t(T_1,...,T_P) = \min_{\bar{T}_1,...,\bar{T}_P} \delta_t(\bar{T}_1,...,\bar{T}_P)$$

where P is the desired number of parts. Since this is an NP hard discrete problem, it is solved approximately with heuristic algorithms.

Typically, the weights are set to value one, $w(t_i) = 1$ for all i and $W(t_j,t_k) = 1$ for all j,k, and algorithms produce a partition as shown in Figure 1 or theoretically as Figure 2.

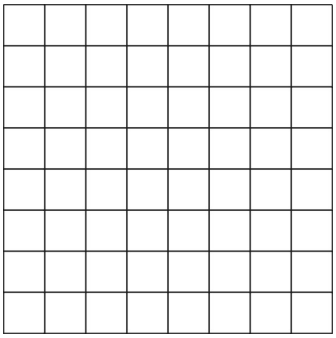
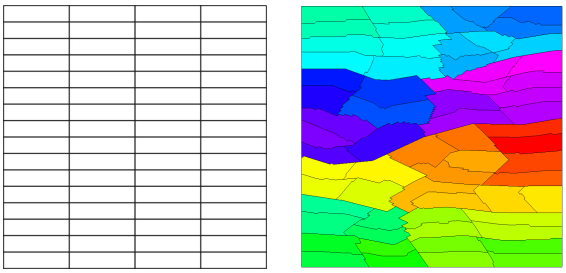


Fig. 2: Theoretical partition into 64 parts.

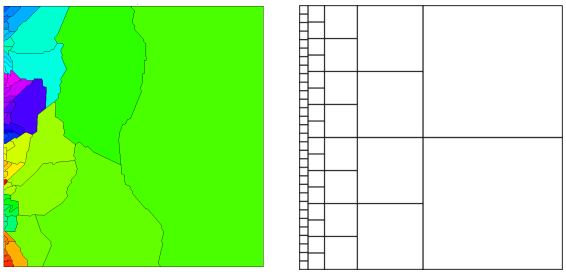
If the edge weights are set to different values, one can achieve partitions with rectangular parts as pictured in Figure 3. [15]. If the triangle weights are set to different values, one can achieve partitions having parts of varying sizes as pictured in Figure 4. [3] [4] [15].

Do these alternate partitions affect DD convergence rates? It has been shown in [15] that if (1)-(3) is solved with strong convection present, namely $a = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $b = \begin{bmatrix} \beta & 0 \end{bmatrix}^T$,



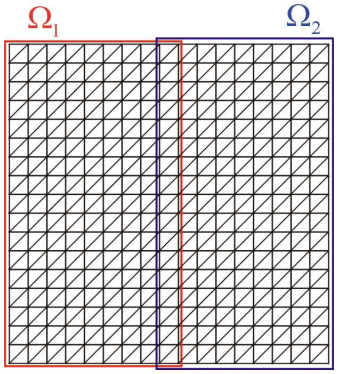
(a) Theoretical rectangles (b) METIS rectangles

Fig. 3: Unit square partitioned with edge weights into 64 parts.

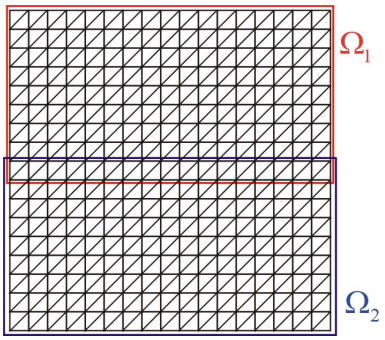


(a) METIS (b) Theoretical

Fig. 4: Unit square partitioned with triangle weights into approximately 64 parts.



(a) Partition $\bar{K}_0^{(1)}$



(b) Partition $\bar{K}_0^{(2)}$

Fig. 5: Different partitions of the unit square.

$|\beta h| \geq 1$ where h is the mesh size, $c = 0$, $f = 1$, and $u = 0$ on $\partial\Omega$, then the partition does affect the convergence rate.

When using Finite Differences and solving the above PDE with the Additive Schwarz DD method, consider the two different overlapping partitions of the unit square pictured in Figure 5. When using rectangle subdomains that align with the direction of convection, namely partition $\bar{K}_0^{(2)}$, DD has an asymptotic convergence rate of

$$\frac{U_{\frac{m}{2}-1}\left(1+\frac{1+\beta h-2\sqrt{\beta h}}{2}+\sqrt{\beta h}\left(1+\cos\left(\frac{m\pi}{m+1}\right)\right)\right)}{U_{\frac{m}{2}}\left(1+\frac{1+\beta h-2\sqrt{\beta h}}{2}+\sqrt{\beta h}\left(1+\cos\left(\frac{m\pi}{m+1}\right)\right)\right)} \tag{10}$$

and when using rectangle subdomains that are perpendicular to convection, namely partition $\bar{K}_0^{(1)}$, DD has an asymptotic convergence rate of

$$\frac{U_{\frac{m}{2}-1}\left(1+\frac{1+\beta h-2\sqrt{\beta h}}{2\sqrt{\beta h}}+\frac{1}{\sqrt{\beta h}}\left(1+\cos\left(\frac{m\pi}{m+1}\right)\right)\right)}{U_{\frac{m}{2}}\left(1+\frac{1+\beta h-2\sqrt{\beta h}}{2\sqrt{\beta h}}+\frac{1}{\sqrt{\beta h}}\left(1+\cos\left(\frac{m\pi}{m+1}\right)\right)\right)} \tag{11}$$

$U_k(\cdot)$ is the k^{th} Chebyshev polynomial of the second kind, h is the mesh size, $m = \frac{1}{h}$, and [15] shows that (10) $<$ (11). Thus DD converges faster when the subdomains align with strong convection.

Similarly, it is shown in [15] that this is true for anisotropic diffusion and Finite Elements. Solve (1)-(3) with $a = \begin{bmatrix} \alpha & 0 \\ 0 & 1 \end{bmatrix}$, $|\alpha| \geq 2$, $b = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$, $c = 0$, $f = 1$, and $u = 0$ on $\partial\Omega$, and compare partitions $\bar{K}_0^{(1)}$ and $\bar{K}_0^{(2)}$ again. And partitions are shown to affect DD convergence rates if Multiplicative Schwarz is used instead of Additive Schwarz [15].

Other techniques such as downwind numbering have also been shown to be helpful for improving the convergence of convection-diffusion problems [13].

3 Stiffness Matrix Weighting Scheme

When a PDE has strong convection and/or anisotropic diffusion, directional dependence between degrees of freedom exist. The Stiffness Matrix Weighting Scheme automatically detects this and applies weights to the partitioning process to encourage the creation of rectangular subdomains which align with this dependence direction.

The finite element solution of (1)-(3) is a discrete approximation with the form

$$u_h = u_D + \sum_{k=1}^{n} U_k \psi_k$$

where u_D is some function that satisfies the dirchelet boundary conditions and $\{\psi_k | k = 1,2,...,n\}$ are the C^0 piecewise linear Lagrangian basis functions. The degrees of freedom of u_h are $U \in \mathbb{R}^n$ and are computed by solving

$$AU - F = 0. \tag{12}$$

Here the *Stiffness Matrix* $A \in \mathbb{R}^{n\times n}$ is $[A]_{j,k} = a(\psi_k, \psi_j)$. The *Load Vector* $F \in \mathbb{R}^n$ is $[F]_i = b(\psi_i) - a(u_D, \psi_i)$. The functions $a(u,v)$ and $b(v)$ follow from the Weak Form of (1)-(3)

$$a(u,v) = \int_\Omega a\nabla u \cdot \nabla v + (b\cdot\nabla u)v + cuv\,dx\,dy.$$

$$b(v) = \int_\Omega fv\,dx\,dy + \int_{\partial\Omega_N} g_N v\,ds.$$

When (1)-(3) is not self-adjoint, the discretization needs some upwinding terms added to improve stability. Therefore $a(u,v)$ is replaced by $a_h(u,v)$ in (12)

$$a_h(u_h, v_h) = a(u_h, v_h) + \int_\Omega \nabla u_h \cdot (w_h \nabla v_h)\,dx\,dy$$

When solving the partitioning equation (9) for PDE (1)-(3), the Stiffness Matrix Weighting Scheme adds weights to the triangle edges of the domain triangularization T as follows

$$W(t_i,t_j) = \beta + \tag{13}$$
$$\alpha \max\left\{\frac{|A_{m+2,m}| + |A_{m+2,m+1}|}{2|A_{m+2,m+2}|}, \frac{|A_{m,m+2}|}{2|A_{m,m}|} + \frac{|A_{m+1,m+2}|}{2|A_{m+1,m+1}|}\right\} +$$
$$\alpha \max\left\{\frac{|A_{m+3,m}| + |A_{m+3,m+1}|}{2|A_{m+3,m+3}|}, \frac{|A_{m,m+3}|}{2|A_{m,m}|} + \frac{|A_{m+1,m+3}|}{2|A_{m+1,m+1}|}\right\}$$

$[A]_{r,s}$ is the stiffness matrix entry corresponding with the interaction between degrees of freedom U_s and U_r where U_s and U_r are defined by Figure 6 which displays triangle t_i and t_j. Scaling factors α and β are generally set to 100 and 1 respectively but can be changed to stretch the aspect ratio of the resultant rectangle subdomains.

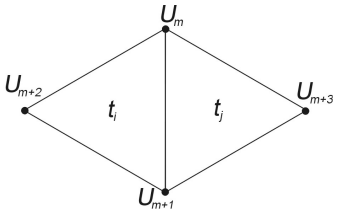
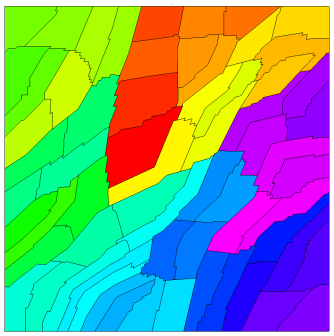
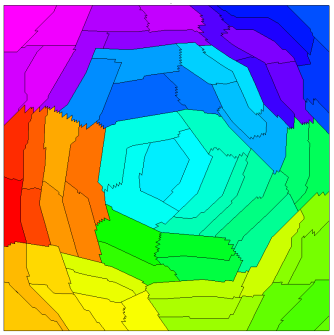


Fig. 6: Diagram for equation (13)

This algorithm is described in more detail in [15]. Figure 7 shows two examples of using Stiffness Matrix Weighting. Figure 7a is the result of the original PDE (1)-(3) having anisotropic diffusion with a diagonal prominent direction, namely $a = \frac{1}{2}\begin{bmatrix}1 & -1\\ 1 & 1\end{bmatrix}\begin{bmatrix}\alpha & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}1 & 1\\ -1 & 1\end{bmatrix}$, $|\alpha| \geq 2$, $b = 0$, $c = 0$, and $f = 1$. And Figure 7b is the result of having circular strong convection, namely $a = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$, $b = \beta\begin{bmatrix}0.5-y\\ x-0.5\end{bmatrix}$, $|\beta h| \geq 1$, $c = 0$, and $f = 1$.



(a) anisotropic diffusion with diagonal prominence



(b) circular strong convection

Fig. 7: METIS using the Stiffness Matrix Weighting Scheme.

4 Error Weighting Scheme

When a PDE has strong convection directed into a dirichlet boundary, an alternative to Stiffness Matrix Weighting is Error Weighting. The PDE solution has a boundary layer that Finite Elements using a uniform mesh with linear basis functions has difficulty approximating. See Figure 8.

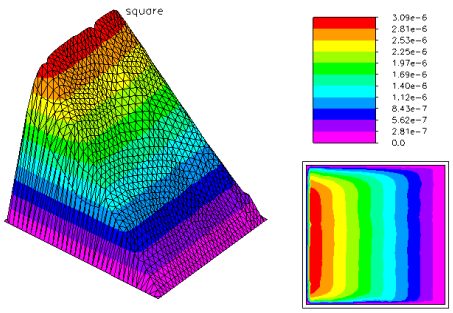


Fig. 8: PDE solution with boundary layer

The finite elements, $t \in T$, near the boundary layer have greater error in the form of $||\nabla(u-u_h)||_{L^2(t)}$ than other elements. The Error Weighting Scheme uses this information to add weights to the partitioning process.

Without knowing the true solution u of (1)-(3), we can not compute $||\nabla(u-u_h)||_{L^2(t)}$ directly for linear Finite Elements. Bank and Xu developed an asymptotically exact estimate [11] [12] which is based on a superconvergent approximation of the order 2 derivatives of u. [7] [11] [12]

$$||\nabla(u-u_h)||_{L^2(t)} \approx ||\nabla e_t||_{L^2(t)}$$

$$e_t(x,y) = \sum_{k=1}^{3} l_k^2 t_k^T \bar{M}_t t_k q_k(x,y)$$

$$\bar{M}_t = \frac{\alpha_t}{2}(\tilde{M}_t + \tilde{M}_t^T)$$

$$\tilde{M}_t = -\frac{1}{2}\begin{bmatrix} \partial_x S^m Q_h \partial_x u_h & \partial_x S^m Q_h \partial_y u_h \\ \partial_y S^m Q_h \partial_x u_h & \partial_y S^m Q_h \partial_y u_h \end{bmatrix}$$

Triangle $t \in T$ has 3 sides. l_k is the length of side k. t_k is the unit tangent of side k. $q_k(x,y)$ is a quadratic function equal to 1 at the midpoint of side k and equal to 0 at the three vertices and other two midpoints. Q_h is the L_2 projection from the space of discontinuous piecewise constant functions into the space of continuous piecewise linear polynomials. S_m is a smoothing operator based on the discrete Laplace operator.

When solving the partitioning equation (9) for PDE (1)-(3), the Error Weighting Scheme adds weights to the triangles of the domain triangularization $\forall t \in T$ as follows

$$w(t) = ||\nabla(u-u_h)||_{L^2(t)}$$

This encourages the creation of a partition with subdomains of varying sizes that decrease from large to small in the direction of convection. Since each subdomain originally contains a different number of degrees of freedom (triangles), each processor will refine its subdomain to an agreed upon number of degrees of freedom thus balancing the load and adding degrees of freedom to regions where they are needed to improve the final finite element solution's accuracy.

This algorithm is the main component of the Bank-Holst Paradigm and is described in more detail in [3] [4] [15]. Figure 9 shows an example of using Error Weighting to solve (1)-(3) with dirichlet boundary conditions, $a = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $b = \begin{bmatrix} \beta & 0 \end{bmatrix}^T$, $\beta h \geq 1$, $c = 0$, $f = 1$, and $u = 0$ on $\partial\Omega$.

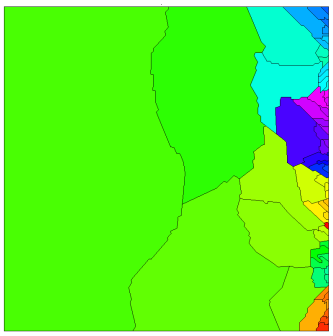


Fig. 9: METIS using the Error Weighting Scheme

5 Numerical Experiment Design

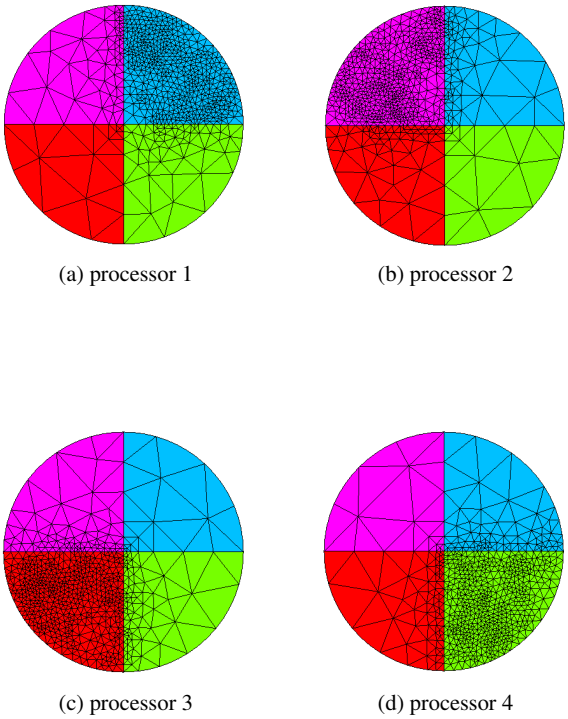
For very simple cases, Section 2 demonstrates mathematically that partition design affects the convergence rate of DD. In order to test the influence of more complicated partitions on DD convergence, we conduct numerical experiments.

Our numerical experiments are performed using PLTMG 11.0, METIS 5.1.0, and SG running on CCoM's computing resource BOOM. PLTMG 11.0 is a package for solving elliptic partial differential equations in general regions of the plane created by Randolph Bank [2]. BOOM is a resource of the Center for Computational Mathematics at the University of California San Diego. BOOM is a ROCKS-based 720-core/1440-GB (60 dual-cpu/six-core/24GB nodes) 64-bit Xeon Cluster (from Dell). METIS 5.1.0 is a serial software package for partitioning large irregular graphs, partitioning large meshes, and computing fill-reduced ordering of sparse matrices created by George Karypis [17]. SG is a visualization tool created by Michael Holst which provides most of the visualizations for this paper [16].

6 Bank-Holst Paradigm DD Solver

PLTMG 11.0 solves elliptic PDEs by the methods described in Section 1 and uses a unique Domain Decomposition solver that has properties of both overlapping and non-overlapping methods. Every processor maintains a mesh of the entire domain giving it the benefits of overlapping schemes. And, similar to non-overlapping methods, each processor owns a unique disjoint portion of the domain known as its subdomain. When a processor refines its mesh, it places the majority of the new degrees of freedom within its subdomain. Therefore on a specific processor, the domain has a fine mesh within its subdomain and a coarse mesh outside its subdomain. Figure 10 illustrates this for 4 processors. The entire domain is the unit circle. Processor 1's sub-

domain is quadrant 1 and processor k's subdomain is quadrant k.



(a) processor 1 (b) processor 2

(c) processor 3 (d) processor 4

Fig. 10: Local meshes of four processors.

The Bank-Holst paradigm DD solver enforces the transmission conditions between the disjoint subdomains with Lagrange Multipliers as in (6)- (8). But because each processor has information about the entire domain, the Bank-Holst paradigm DD solver method doesn't need to explicitly solve for the multipliers.

Each processor creates their own discrete saddle point problem from (6)-(8) and then solves only for the unknowns they need.

$$\begin{bmatrix} A_1^{II} & A_1^{IB} & 0 & 0 & 0 \\ A_1^{BI} & A_1^{BB} & 0 & 0 & I \\ 0 & 0 & \bar{A}_2^{II} & \bar{A}_2^{IB} & 0 \\ 0 & 0 & \bar{A}_2^{BI} & \bar{A}_2^{BB} & -I \\ 0 & I & 0 & -I & 0 \end{bmatrix} \begin{bmatrix} \delta U_1^I \\ \delta U_1^B \\ \delta \bar{U}_2^I \\ \delta \bar{U}_2^B \\ \lambda \end{bmatrix} = \begin{bmatrix} R_1^I \\ R_1^B \\ R_2^I \\ R_2^B \\ U_2^B - U_1^B \end{bmatrix} \tag{14}$$

In the case of two processors, Equation (14) is the saddle point problem that processor 1 forms. In place of $A_2^{II}, A_2^{IB}, A_2^{BI}$, and A_2^{BB} it substitutes its own stiffness matrices created from the coarse mesh that it has located in processor 2's subdomain and denoted above as $\bar{A}_2^{II}, \bar{A}_2^{IB}, \bar{A}_2^{BI}$, and $\bar{A}_2^{BB}$. Processor 1 receives R_2^B and U_2^B from processor 2 and it sets $R_2^I = 0$. Processor 2 creates a similar saddle point problem substituting its coarse stiffness matrices for A_1's and receives R_1^B and U_1^B from processor 1.

Since processor 1 doesn't need the quantities $\delta \bar{U}_2^B$ and λ, it reorders (14) and eliminates these sets of equations by block elimination

$$\left[\begin{array}{cc|ccc} 0 & -I & 0 & I & 0 \\ -I & \bar{A}_2^{BB} & 0 & 0 & \bar{A}_2^{BI} \\ \hline 0 & 0 & A_1^{II} & A_1^{IB} & 0 \\ I & 0 & A_1^{BI} & A_1^{BB} & 0 \\ 0 & \bar{A}_2^{IB} & 0 & 0 & \bar{A}_2^{II} \end{array}\right] \left[\begin{array}{c} \lambda \\ \delta \bar{U}_2^B \\ \hline \delta U_1^I \\ \delta U_1^B \\ \delta \bar{U}_2^I \end{array}\right] = \left[\begin{array}{c} U_2^B - U_1^B \\ R_2^B \\ \hline R_1^I \\ R_1^B \\ R_2^I \end{array}\right]$$

The 3×3 Schur complement system is

$$\begin{bmatrix} A_1^{II} & A_1^{IB} & 0 \\ A_1^{BI} & A_1^{BB} + \bar{A}_2^{BB} & \bar{A}_2^{BI} \\ 0 & \bar{A}_2^{IB} & \bar{A}_2^{II} \end{bmatrix} \begin{bmatrix} \delta U_1^I \\ \delta U_1^B \\ \delta \bar{U}_2^I \end{bmatrix} = \begin{bmatrix} R_1^I \\ R_1^B + R_2^B + \bar{A}_2^{BB}(U_2^B - U_1^B) \\ \bar{A}_2^{IB}(U_2^B - U_1^B) \end{bmatrix} \tag{15}$$

Algorithm 1 Bank-Holst paradigm DD solver

Initialize U_1 and U_2 to an initial guess
1: **for** $k = 0, 1, 2, \ldots$ until convergence **do**
2: Compute residuals R_1 and R_2
3: Communicate R_i^B and U_i^B from Γ to neighbor processors
4: Simultaneously for i=1,2
Solve (15) $A_i^* \delta U_i = R_i^*$ for δU_i.
5: **end for**

Each processor solves its system (15) using the Conjugate Gradient method with an incomplete LU preconditioner.

This algorithm is described in more detail in [8] [1] [18] [10]. It is motivated by and similar to domain decomposition algorithms described in [5] [6]. The Bank-Holst paradigm is described in [3] [4] with an additional contribution described in [9].

7 Stiffness Matrix Weighting Experiments

7.1 Isotropic Diffusion

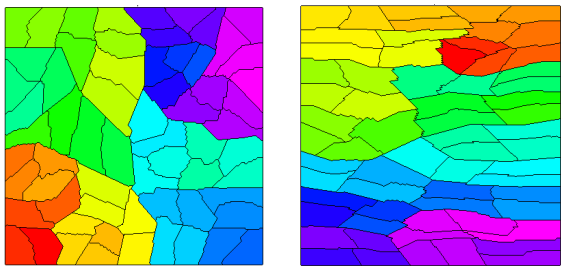
When PDE (1)-(3) is mostly isotropic diffusion, namely $\frac{\lambda_1(a)}{\lambda_2(a)} < 2$ and $||b||h < 1$, then the Stiffness Matrix Weighting Scheme detects no favored directional dependences. The resultant partition is the same as the standard partition created by setting all the weights equal to 1.

In this case, we can force the partitioner to create rectangle subdomains by applying a Directional Weighting Scheme [15]. If v is a given direction vector and $n_{i,j}$ is the unit normal vector to the edge between triangles i and j, then we can add weights to the triangle edges of the domain triangularization T as follows

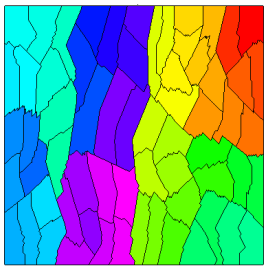
$$W(t_i, t_j) = 4(v \cdot n_{i,j})^2 + 1$$

This encourages the partitioner to create rectangle subdomains of aspect ratio 4 : 1 in the direction of v.

Now, using 64 processors, we solve (1)-(3) with $a = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $b = 0$, $c = 0$, $f = 1$, and $u = 0$ on $\partial\Omega$ for the three cases of $v = 0, \hat{i}, \hat{j}$. Note that when $v = 0$, then $W(t_i, t_j) = 1 \forall i, j$ and creates the standard partition. The three resultant partitions are displayed in Figure 11 and the details of the DD convergence are listed in Table 1.



(a) $v = 0$ (b) $v = \hat{i}$



(c) $v = \hat{j}$

Fig. 11: Directional Weighting with $v = 0, \hat{i}, \hat{j}$.

Table 1: DD convergence details

	$v = 0$	$v = \hat{i}$	$v = \hat{j}$
$\sqrt[k-1]{\frac{\|\|\delta u_k\|\|}{\|\|\delta u_1\|\|}}$	0.37	0.51	0.51
$\|\|u_0\|\|$	4.13×10^{-2}	4.13×10^{-2}	4.13×10^{-2}
$\sqrt[k-1]{\frac{\|\|r_k\|\|}{\|\|r_1\|\|}}$	0.43	0.57	0.60
$\|\|r_0\|\|$	5.01	4.96	5.12
$\|\|e_h\|\| \approx \|\|u - u_h\|\|$	1.71×10^{-8}	1.87×10^{-8}	1.83×10^{-8}
iterations	8	12	12

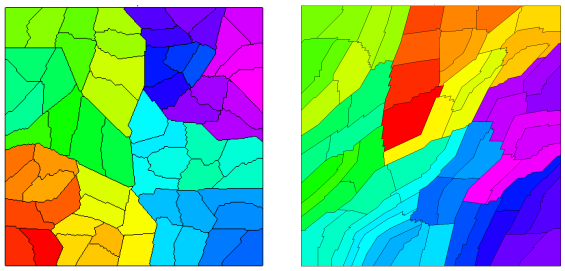
With regard to the $L_2(\Omega)$ norm, the first five rows of Table 1 show the convergence rate of the solution increment $||\delta u_k||$, the initial solution $||u_0||$, the convergence rate of the residual $||r_k||$, the initial residual, and the error of the finite element solution respectively. The last row lists the number of required DD iterations. DD iterations are terminated when

$$||\delta u_k|| = ||u_k - u_{k-1}|| < \frac{1}{10}||u - u_h|| \approx \frac{1}{10}||e_h|| \tag{16}$$

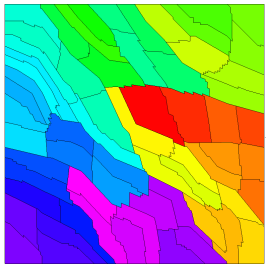
This experiment demonstrates that when no favored directional dependence exists, the standard partition which minimizes the number of degrees of freedom on the interface achieves fastest convergence.

7.2 Anisotropic Diffusion

Next, using 64 processors, we solve (1)-(3) with anisotropic diffusion, $a = \frac{1}{2}\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} \alpha & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$, $b = 0$, $c = 0$, $f = 1$, and $u = 0$ on $\partial\Omega$. We solve with $\alpha = 2, 10, 100$ and compare the three partitioning schemes, Standard Weighting, Stiffness Matrix Weighting, and Orthogonal Stiffness Matrix Weighting. The finite element solution for $\alpha = 100$ is pictured in Figure 13. The three resultant partitions are displayed in Figure 12. The choice of partitioning scheme does not significantly affect $||u_0||$, $||r_0||$, and $||e_h||$. The other DD convergence details are listed in Tables 2-4.



(a) Standard Weigting (b) Stiffness Matrix Weighting



(c) Orthogonal Stiffness Matrix Weighting. (Directional Weighting $v = [1, -1]$)

Fig. 12: Different partitioning schemes

Table 2: DD $\sqrt[k-1]{\frac{||\delta u_k||}{||\delta u_1||}}$

α	Standard	Stiffness	Orthogonal Stiffness
2	0.41	0.36	0.50
10	0.56	0.45	0.63
100	0.61	0.54	0.64

Table 3: DD $\sqrt[k-1]{\frac{||r_k||}{||r_1||}}$

α	Standard	Stiffness	Orthogonal Stiffness
2	0.45	0.41	0.57
10	0.58	0.47	0.69
100	0.67	0.58	0.76

Table 4: DD iterations

α	Standard	Stiffness	Orthogonal Stiffness
2	10	8	12
10	14	11	18
100	16	13	18

This experiment demonstrates that when a favored directional dependence exists because of anisotropic diffusion, a partition which aligns with this direction achieves fastest convergence with the same level of accuracy as compared to other partitions.

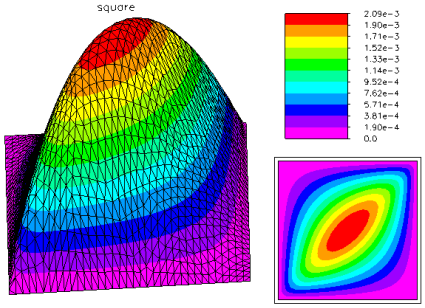


Fig. 13: PDE solution with anisotropic diffusion

7.3 Strong Convection

Next, using 64 processors, we solve (1)-(3) with strong convection, $a = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $b = \begin{bmatrix} \beta & 0 \end{bmatrix}^T$, $c = 0$, $f = 1$, and $u = 0$ on $\partial\Omega$. We solve with $\beta h = 1, 10, 100$ where h is the mesh size and compare the three partitioning schemes, Standard Weighting, Stiffness Matrix Weighting, and Orthogonal Stiffness Matrix Weighting. The finite element solution for $\beta h = 100$ is pictured in Figure 8. The three resultant partitions are the same as those displayed in Figure 11. The choice of partitioning scheme does not significantly affect $||u_0||$, $||r_0||$, and $||e_h||$. The other DD convergence details are listed in Tables 5-7.

Table 5: DD $\sqrt[k-1]{\frac{||\delta u_k||}{||\delta u_1||}}$

βh	Standard	Stiffness	Orthogonal Stiffness
1	0.23	0.14	0.28
10	0.33	0.21	0.44
100	0.34	0.24	0.44

Table 6: DD $\sqrt[k-1]{\frac{||r_k||}{||r_1||}}$

βh	Standard	Stiffness	Orthogonal Stiffness
1	0.23	0.16	0.30
10	0.36	0.23	0.46
100	0.37	0.25	0.47

Table 7: DD iterations

βh	Standard	Stiffness	Orthogonal Stiffness
1	6	4	6
10	7	4	8
100	7	4	8

This experiment demonstrates that when a favored directional dependence exists because of strong convection, a partition which aligns with this direction achieves fastest convergence with the same level of accuracy as compared to other partitions.

Additionally, it is shown in [15] that all the results in Sections 7.1–7.3 hold when varying the number of processors, the boundary conditions, the domain shape, and the forcing function f in the PDE.

8 Error Weighting Experiments

8.1 Isotropic Diffusion

When the finite element solution of (1)-(3) has equal error on each element of a uniform global mesh, $||\nabla(u-u_h)||_{L^2(t)}$, then the partition created from Error Weighting is the same

as the standard partition created by setting all weights equal to 1. Therefore if (1)- (3) is solved with $a = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $b = 0$, $c = 0$, $f = 1$ with and without Error Weighting, the partitions are basically the same.

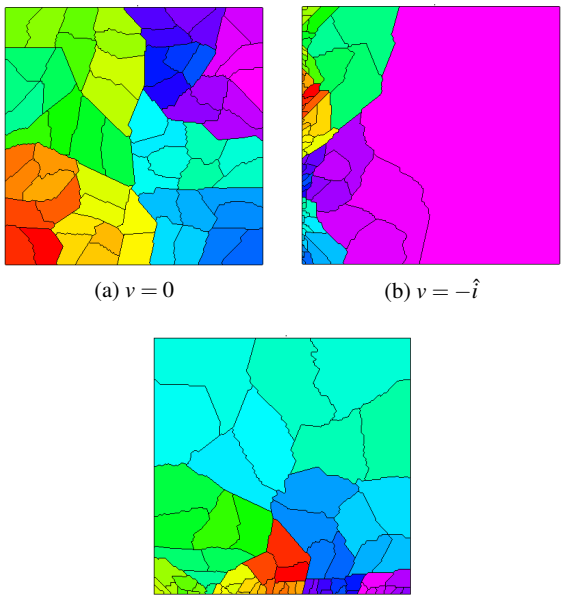
In this case, the partitioner can be forced to create subdomains of varying sizes that decrease from large to small in a direction v of choice by applying a Flow Weighting Scheme [15]. To apply Flow Weighting on the unit square in the direction $v = -\hat{i}$, we could add weights to the triangles of the domain triangularization $\forall t \in T$ as follows:

$$w(t) = \left(p_x(t) + 10^{-3}\right)^{-1.5}$$

where $p_x(t)$ is the x coordinate of the center of triangle t. To apply Flow Weighting on the unit square in the direction of $v = -\hat{j}$, we could use:

$$w(t) = \left(p_y(t) + 10^{-3}\right)^{-1.5}$$

Figure 14 shows the resultant partitions from using Flow Weighting and $v = 0, -\hat{i}, -\hat{j}$. When $v = 0$, we define $w(t) = 1 \forall t$.



(a) $v = 0$ (b) $v = -\hat{i}$

(c) $v = -\hat{j}$

Fig. 14: Flow Weighting with $v = 0, -\hat{i}, -\hat{j}$.

If (1)-(3) is solved with $a = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $b = 0$, $c = 0$, $f = 1$, and $u = 0$ on $\partial\Omega$ using the three partitions from Flow Weighting with $v = 0, -\hat{i}, -\hat{j}$, there is no significant difference between the DD convergence rates of each.

8.2 Strong Convection

Next, using 64 processors, we solve (1)-(3) with strong convection and a dirichlet boundary, $a = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $b = \begin{bmatrix} \beta & 0 \end{bmatrix}^T$, $c = 0$, $f = 1$, and $u = 0$ on $\partial\Omega$. We solve with $\beta h = 1, 10, 100$ where h is the mesh size and compare the three partitioning schemes, Standard Weighting, Error Weighting, and Orthogonal Error Weighting. The finite element solution for $\beta h = 100$ is pictured in Figure 8. The three resultant partitions are the same as those displayed in Figure 14. The DD convergence details are listed in Tables 8-11.

Table 11 lists the number of iterations before DD is terminated by the stopping criteria of (16). The numbers in parenthesis are how many iterations are required to obtain the same $||\delta u_k||$ accuracy as the standard partition. From Table 8, we see that using a partition created from Error Weighting allows the finite element solution to achieve greater accuracy than using the standard partition. And using an Orthogonal Error Weighted partition produces a less accurate solution.

Table 8: DD $||e_h|| \approx ||u - u_h||$

βh	Standard	Error	Orthogonal Error
1	3.1×10^{-7}	1.64×10^{-7}	1.25×10^{-6}
10	9.31×10^{-8}	2.78×10^{-8}	1.34×10^{-7}
100	9.33×10^{-9}	3.52×10^{-9}	1.38×10^{-8}

Table 9: DD $\sqrt[k-1]{\frac{||\delta u_k||}{||\delta u_1||}}$

βh	Standard	Error	Orthogonal Error
1	0.23	0.09	0.29
10	0.33	0.07	0.66
100	0.34	0.07	0.72

Table 10: DD $\sqrt[k-1]{\frac{||r_k||}{||r_1||}}$

βh	Standard	Error	Orthogonal Error
1	0.23	0.13	0.35
10	0.36	0.13	0.75
100	0.37	0.16	0.82

Table 11: DD iterations

βh	Standard	Error	Orthogonal Error
1	6	3 (3)	4 (5)
10	7	4 (3)	10 (12)
100	7	3 (3)	13 (14)

This experiment demonstrates that when a favored directional dependence exists because of strong convection, a partition which aligns with this direction achieves fastest convergence. Furthermore, Error Weighting produces a more accurate finite element solution when a boundary layer is present.

Additionally, it is shown in [15] that all the results in Sections 8.1-8.2 hold when varying the number of processors, the boundary conditions, the domain shape, and the forcing function f in the PDE.

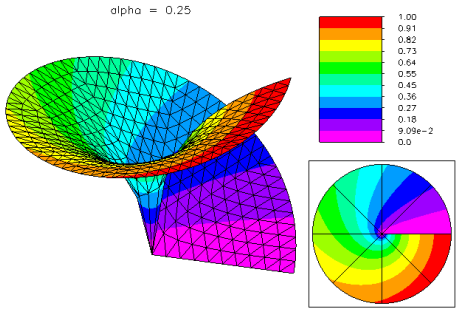
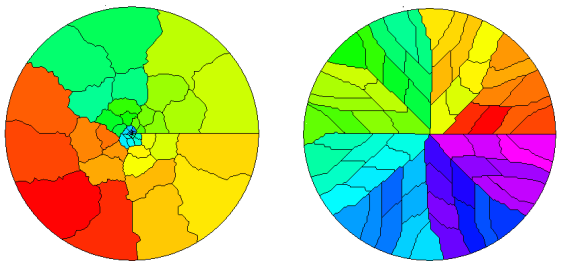


Fig. 15: PDE solution with singularity



(a) Error Weighting (b) Standard Weighting

Fig. 16: Partitions on the unit circle.

8.3 Singularities

The Error Weighting Scheme does not detect anisotropic diffusion in general, but Error Weighting has the additional advantage of detecting singularities. Singularities can be present in the solution of (1)-(3) when singularities are present in a, b, c, f, or the boundary conditions. When the solution u has a singularity, Error Weighting produces a more accurate finite element solution than Standard Weighting because it adds degrees of freedom where they are needed.

Furthermore, the partition created from Error Weighting maintains the same DD convergence rate as the standard partition. Using 64 processors, we solved (1)-(3) with $a = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $b = 0$, $c = 0$, $f = 0$. We created a singularity in the boundary conditions by solving over the unit circle with a crack down the x axis. Homogenous Dirichlet boundary conditions are imposed on the top of the crack, and homogenous Neumann boundary conditions are imposed below the crack with $g_D = 0$ and $g_N = 0$. The remainder of the boundary has $u = \sin\frac{1}{4}\theta$.

We compared the two partitioning schemes, Standard Weighting and Error Weighting. The finite element solution is pictured in Figure 15. The two resultant partitions are displayed in Figure 16 and the details of the DD convergence are listed in Table 12.

Even though Standard Weighting has a smaller initial residual, Error Weighting computed a better initial guess $||u - u_0||$. Both partitions have approximately the same $||\delta u_k||$ convergence rate and Error Weighting achieves $||\delta u_k|| < (\frac{1}{10})3.88 \times 10^{-5}$ in 7 iterations while Standard Weighting needs 11 iterations. The better global mesh from the Error Weighting partition allows its finite element solution to achieve greater accuracy. Error Weighting achieves $||\delta u_k|| < (\frac{1}{10})2.18 \times 10^{-7}$ after 14 iterations. DD iteration is terminated by the stopping criteria of (16).

Table 12: DD convergence details

	Standard	Error
$\sqrt[k-1]{\frac{\|\|\delta u_k\|\|}{\|\|\delta u_1\|\|}}$	0.46	0.49
$\|\|u_0\|\|$	1.12	1.12
$\sqrt[k-1]{\frac{\|\|r_k\|\|}{\|\|r_1\|\|}}$	0.49	0.48
$\|\|r_0\|\|$	9.72×10^3	1.9×10^7
$\|\|e_h\|\| \approx \|\|u - u_h\|\|$	3.88×10^{-5}	2.18×10^{-7}
iterations	11	14 (7)

This experiment demonstrates that when the solution u has a singularity caused by boundary conditions, using a par-

tition created from Error Weighting produces a more accurate finite element solution while maintaining the DD convergence rate. Additionally these results hold for the other types of singularity problems listed in the beginning of Section 8.3.

9 Computational Resources

In each of these experiments, we monitored the CPU, memory, and communication usage. We observed that the choice of partition did not significantly affect the time needed to complete one iteration.

Specifically, we found that the communication time required between processors was proportional to the length of the interface between subdomains. When using Stiffness Matrix Weighting with rectangle parts of aspect ratio 4:1, the interface was 25% longer than the standard unweighted partition. Error Weighting did not significantly affect the length of the interface.

The Bank-Holst paradigm DD solver spent approximately $1/500$ of the time needed for each iteration in communication. Therefore a small percentage change in communication did not significantly change the time needed to complete one iteration.

The majority of each iteration's time was spent in computation, specifically computing the ILU preconditioner for the Conjugate Gradient method. We found that the computation time needed to complete one iteration was not significantly affected by the choice of partition. And memory usage was not significantly affected either. These results are shown in [15].

Therefore, among these experiments, the overall solve time to complete DD was proportional to the number of iterations needed and improving the convergence rate lowered the overall solve time.

10 Conclusion

In conclusion, when Finite Elements are employed to solve a second order elliptic partial differential equation with strong convection and/or anisotropic diffusion, the shape and alignment of a partition's parts significantly affect the Domain Decomposition convergence rate. Given a PDE, if b is the direction of convection or the prominent direction of anisotropic diffusion, then if one considers traversing the domain in the direction of b, partitions having fewer parts to traverse in this direction converge faster while partitions having more converge slower.

Furthermore, partitioning with Stiffness Matrix Weighting (rectangular subdomains aligned in a specific direction) maintains the accuracy of the global finite element solution while improving the DD convergence rate and Error Weighting (subdomain sizes decrease in a specific direction) improves the accuracy of the global finite element solution while improving or maintaining the DD convergence rate.

References

1. Bank, R.E.: A domain decomposition solver for a parallel adaptive meshing paradigm. Domain Decomposition Methods in Science and Engineering **XVI**, 3–14 (2006)
2. Bank, R.E.: PLTMG: A software Package for Solving Elliptic Partial Differential Equations Users' Guide 11.0 (2012)
3. Bank, R.E., Holst, M.J.: A new paradigm for parallel adaptive meshing algorithms. SIAM J. on Scientific Computing **22**, 1411–1443 (2000)
4. Bank, R.E., Holst, M.J.: A new paradigm for parallel adaptive meshing algorithms. SIAM Review **45**, 292–323 (2003)
5. Bank, R.E., Jimack, P., Nadeem, S.A., Nepomnyaschikh, S.V.: A weakly overlapping domain decomposition preconditioner for the finite element solution of elliptic partial differential equations. SIAM J. on Scientific Computing **23**, 1817–1841 (2002)
6. Bank, R.E., Jimack, P.K.: A new parallel domain decomposition method for the adaptive finite element solution of elliptic partial differential equations. Concurrency and Computation: Practice and Experience **13**, 327–350 (2001)
7. Bank, R.E., Lu, J.: Asymptotically exact a posteriori error estimators, part i: Grids with superconvergence. SIAM J. Numerical Analysis **41**, 2294–2312 (2003)
8. Bank, R.E., Lu, S.: A domain decomposition solver for parallel adaptive meshing paradigm. SIAM J. on Scientific Computing **45**, 292–323 (2003)
9. Bank, R.E., Ovall, J.S.: Dual functions for a parallel adaptive method. SIAM J. on Scientific Computing **29**, 1511–1524 (2007)
10. Bank, R.E., Vassilevski, P.S.: Convergence analysis of a domain decomposition paradigm. Computing and Visualization in Science **11**, 333–350 (2008)
11. Bank, R.E., Xu, J.: Asymptotically exact a posteriori error estimators, part ii: General unstructer grids. SIAM J. Numerical Analysis **41**, 2313–2332 (2003)
12. Bank, R.E., Xu, J., Zheng, B.: Superconvergent derivative recovery for lagrange triangular elements of degree p on unstructured grids. SIAM J. Numerical Analysis **45**, 2032–2046 (2007)
13. Bey, J., Wittum, G.: Downwind numbering: robust multigrid for convection-diffusion problems. Appl. Numer. Math. **23**, 177–192 (1997)
14. Bichot, C.E., Siarry, P.: Graph Partitioning. Wiley (2011)
15. Deotte, C.: Domain partitioning methods for elliptic partial differential equations. Ph.D. thesis, University of California at San Diego (2014)
16. Holst, M.J.: Sg user's guide. http://www.fetk.org/codes/sg/index.html (2014)
17. Karypis, G.: METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reduced Orderings of Sparse Matrices Version 5.1.0 (2013)
18. Lu, S.: Parallel adaptive multigrid algorithms. PhD thesis, Dept of Math, UCSD (2004)
19. Mathew, T.P.: Domain decomposition methods for the numerical solution of partial differential equations. Springer (2008)
20. Smith, B.F., Bjørstad, P.E., Gropp, W.D.: Domain Decomposition: Parallel multilevel methods for elliptic partial differential equations. Cambridge University Press (1996)
21. Toselli, A., Widlund, O.: Domain Decomposition Methods. Springer (2005)