Mathematical Modelling and Numerical Analysis
Modélisation Mathématique et Analyse Numérique

ESAIM: M2AN
M2AN, Vol. 37, N° 1, 2003, pp. 133–142
DOI: 10.1051/m2an:2003016

NUMERICAL STUDY OF TWO SPARSE AMG-METHODS

Janne Martikainen[1]

Abstract. A sparse algebraic multigrid method is studied as a cheap and accurate way to compute approximations of Schur complements of matrices arising from the discretization of some symmetric and positive definite partial differential operators. The construction of such a multigrid is discussed and numerical experiments are used to verify the properties of the method.

Mathematics Subject Classification. 65F10, 65N22.

Received: November 30, 2001. Revised: October 9, 2002.

Introduction

Lagrange multipliers [1] are used frequently in fictitious domain methods to enforce boundary conditions [7,8], and in domain decomposition methods to connect the solutions of subdomains on the inner boundaries [2]. The use of boundary supported Lagrange multipliers with a finite element discretization of a symmetric, positive definite partial differential operator leads to a saddle point problem of the following form

$$\begin{pmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{p} \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \mathbf{g} \end{pmatrix}. \tag{0.1}$$

When such problems are solved iteratively, a preconditioner is often used to accelerate the convergence rate of the iterative method. It is a well known result, originally due to Kuznetsov (1990), that an efficient preconditioner $\mathbf{P}$ for the problem (0.1) is given by

$$\mathbf{P} = \begin{pmatrix} \tilde{\mathbf{A}} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{S}} \end{pmatrix}, \tag{0.2}$$

where the matrices $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{S}}$ are close to the matrices $\mathbf{A}$ and $\mathbf{S} = \mathbf{B}\mathbf{A}^{-1}\mathbf{B}^T$ [18]. The matrix $\tilde{\mathbf{A}}$ may be a fictitious domain or domain decomposition preconditioner. Since the matrix $\mathbf{B}$ corresponds to the boundary Lagrange multipliers, the operation $\mathbf{A}^{-1}$ has to be approximated only on a subset of the discretization nodes located in the neighborhood of the boundary. On the other hand, such an approximation should be cheap to compute, since preconditioning requires solving the system $\tilde{\mathbf{S}}\mathbf{u} = \mathbf{v}$ iteratively with varying $\mathbf{v}$. A few approximation schemes have been suggested for $\mathbf{A}^{-1}$ in a subspace such as $K^{1/2}$ family of preconditioners [3, 6, 9, 14], sparse BPX [13, 15, 19] and sparse algebraic multigrid methods [12]. The term "algebraic multigrid" in [12] refers to

Keywords and phrases. Algebraic multigrid, Schur complement, Lagrange multipliers.

[1] University of Jyväskylä, Department of Mathematical Information Technology, P.O. Box 35 (Agora), 40351 Jyväskylä, Finland. e-mail: Janne.Martikainen@mit.jyu.fi

© EDP Sciences, SMAI 2003

a multigrid method, which is based on successive block factorizations and uses the Chebyshev method as a smoother. In this article, the construction and the properties of sparse algebraic multigrid methods for Schur complement approximation are studied.

The rest of the paper is organized as follows: First, the algebraic multigrid methods, considered in this paper, are described shortly. Then, the sparse implementation is discussed and finally, the numerical examples are explained. The results are presented to give accurate qualitative information on the properties of the method and to compare the sparse AMG methods with the sparse BPX. In the following, the term "node" refers to a degree of freedom in an algebraic system, not to a discretization node.

1. Algebraic multigrid methods

It is known that stationary iterative methods, known in multigrid methods as smoothers, such as Jacobi and Gauss–Seidel, reduce effectively rapidly oscillating error components, while the reduction of slowly oscillating error components is poor. This observation led to the invention of multigrid. In two-grid methods the slowly converging error components are filtered out by projecting the residual on a coarser grid where the residual equation $\mathbf{Ae} = \mathbf{r}$ is solved. An update is interpolated back to the original grid and added to the approximate solution. This procedure is known as coarse grid correction. In true multigrid methods the coarse grid correction is used recursively until on the coarsest grid the problem is so small that one can afford to solve it accurately. Therefore, in any multigrid method one has a series of linear problems, decreasing in size, and two operators which can transfer the data between the problems, restriction transfers the residual to a coarser level and prolongation transfers the coarse grid correction to a finer level. A typical structure for multigrid algorithm is the following:

Algorithm 1.1. *Multigrid cycle*

1. *If the current level is the coarsest level*
2. *Solve* $\mathbf{A}_{\mathrm{L}}\mathbf{x}_{\mathrm{L}} = \mathbf{f}_{\mathrm{L}}$
3. *End if*
4. *If the current level i is not the coarsest level*
5. $\mathbf{x}_i \leftarrow \mathbf{x}_i + \mathbf{S}_i(\mathbf{f}_i - \mathbf{A}_i\mathbf{x}_i)$
6. $\mathbf{f}_{i+1} = \mathbf{R}_i(\mathbf{f}_i - \mathbf{A}_i\mathbf{x}_i)$
7. $\mathbf{x}_{i+1} = \mathbf{0}$
8. *Call this algorithm ν times for the level $i+1$*
9. $\mathbf{x}_i \leftarrow \mathbf{x}_i + \mathbf{P}\mathbf{x}_{i+1}$
10. $\mathbf{x}_i \leftarrow \mathbf{x}_i + \mathbf{S}_i(\mathbf{f}_i - \mathbf{A}_i\mathbf{x}_i)$
11. *End if.*

Above, the level index i ranges from 1 to L such that level 1 is the finest and L is the coarsest level. The system matrices are denoted by $\mathbf{A}_i$, the smoothers (assumed to be stationary) by $\mathbf{S}_i$, the restrictions by $\mathbf{R}_i$ and the prolongations by $\mathbf{P}_i$. This algorithm is called V-cycle, when $\nu = 1$ and W-cycle, when $\nu = 2$.

In geometric multigrid methods, a hierarchy of discretization meshes is required for obtaining the linear systems and transfer operators, whereas in algebraic multigrid methods they are created by the method itself. Algebraic multigrids use often the transpose of the restriction as a prolongation, and the coarse systems matrices are calculated using the Galerkin formula $\mathbf{A}_{i+1} = \mathbf{R}_i\mathbf{A}_i\mathbf{P}_i$.

It has been proved in [16] that the Algorithm 1.1 defines a symmetric and positive definite operator, if $\mathbf{A}_1$ and $\mathbf{S}_i$ are symmetric and positive definite, the Galerkin formula is used to define the coarse system matrices and the following condition is satisfied on each multigrid level

$$\lambda_{\min}(\mathbf{I} - \mathbf{A}_i\mathbf{S}_i) > -1, \tag{1.1}$$

where $\lambda_{\min}(\cdot)$ denotes the smallest eigenvalue of a given matrix. It should be noted that the inequality (1.1) can be easily ensured by underrelaxing the smoothers $\mathbf{S}_i$ properly.

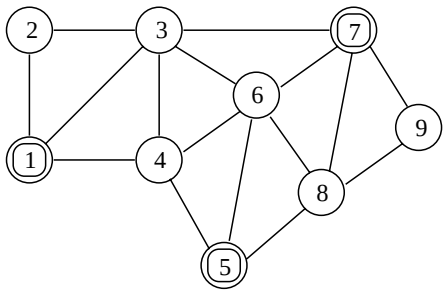

FIGURE 1. An example graph with selected coarse nodes.

Nowadays, there is a vast literature on algebraic multigrid methods starting from [17]. Since, the method of Ruge and Stüben is well documented and readily available, it will be the first algebraic multigrid to be tested here. Its coarsening strategy is based on the notion of a strong connection. Two nodes i and j in a linear system are strongly connected if the system matrix entry A_{ij} satisfies $A_{ij} \leq \theta \min_{k \neq i} A_{ik}$, where $\theta \in]0, 1[$ is a parameter given by the user. Note, that the codiagonals of the matrix $\mathbf{A}$ are assumed to be negative. In the coarsening process, the aim is to select nodes into the coarse system in such a way that the system does not contain any strongly connected pairs and that every eliminated node has a strong connection to at least one node that has been selected to the coarse system. The restriction operator is constructed using the values of the system matrix and the coarse system matrices are calculated using the Galerkin formula. The coarsening process is continued until the size of the coarse linear system is smaller than a given threshold. In the current implementation, the definition of the strong connection has been modified to include positive elements of the system matrix. This has been proposed, for example, in [5]. This algebraic multigrid will be later referred as AMG 1.

The second algebraic multigrid method which is tested here was proposed in [11]. It was chosen because it has many properties in common with geometric multigrid methods and thus, it is expected to give a computationally cheap sparse implementation. Instead of using the actual values stored in the system matrix, this multigrid uses the graph related to the system matrix. The method can be easily implemented to use any given graph related to the problem. The coarsening process operates in a geometric fashion by sequentially choosing a coarse node and eliminating the neighboring nodes of the graph. The criterion for selecting the next coarse grid node is to follow the original numbering.

The selection process for the graph presented in Figure 1 proceeds as follows: First, node number one is selected as a coarse node and nodes two, three and four are eliminated. Next, the node five is selected and nodes six and eight are eliminated. Finally, the node seven is selected and node nine eliminated. The graphs can be constructed for all the levels prior to calculating the coarse system matrices.

The restriction operator of the second method is generated also using the information of the given graph as follows:

$$(R)_{ij} = \begin{cases} 1 & \text{for a fine grid point } j \text{ which is a coarse grid point } i, \\ \dfrac{1}{k} & \text{for a fine grid point } j \text{ which is a neighbor of coarse grid point } i \\ & \text{and has } k \text{ neighboring coarse grid points,} \\ 0 & \text{otherwise.} \end{cases}$$

Again, Galerkin formula is used in order to get the coarse system matrices. For a finite element discretization it can be equivalently defined that a coarse grid basis function ϕ^i_{C} is $\phi^i_{\mathrm{C}} = \sum_{j=1}^{t} (R)_{ij} \phi^j_{\mathrm{F}}$, where ϕ^j_{F} is a fine grid basis function, $j = 1, \ldots, t$. In order to achieve exactly the geometric multigrid the initial graph must be hierarchical, the coarse grid points must be picked up systematically and the Dirichlet boundary conditions must be eliminated after the coarsening process. This algebraic multigrid can be used directly for some PDE systems.

When the initial graph has disjoint subgraphs corresponding to different type of unknowns, the structure of the original system is preserved on coarse levels. This algebraic multigrid will be later called AMG 2.

The BPX method [4], also known as the nodal basis method, is also a multilevel method, but it does not require a smoother or system matrices. It gives a spectrally optimal approximation for the Laplace operator. The method follows the Algorithm 1.2 given below.

Algorithm 1.2. *The BPX method*

1. *If the current level i is not the coarsest level*
   2. $\mathbf{f}_{i+1} = \mathbf{R}_i \mathbf{f}_i$
   3. *Call this algorithm for the level* $i+1$
   4. $\mathbf{f}_i \leftarrow \mathbf{f}_i + \mathbf{P}_i \mathbf{f}_{i+1}$
5. *End if.*

Algorithm 1.2 replaces the vector $\mathbf{f}_1$ with an approximation for $\mathbf{L}^{-1}\mathbf{f}_1$, where $\mathbf{L}$ is the discretized Laplace operator.

2. Sparse implementation of the AMG

The matrix vector product with the inverse of the Schur complement has to be approximated in order to construct the preconditioner (0.2). We are going to do this by iteratively solving systems of the form $\hat{\mathbf{S}}\mathbf{v} = \mathbf{y}$, where $\hat{\mathbf{S}}$ is an approximation of $\mathbf{S}$. The multiplication of a vector $\mathbf{v}$ by the Schur complement $\mathbf{S}$ can be evaluated in three steps: $\mathbf{u}^1 = \mathbf{B}\mathbf{v}$, $\mathbf{A}\mathbf{u}^2 = \mathbf{u}^1$ and $\mathbf{u}^3 = \mathbf{B}^T\mathbf{u}^2$, where vector $\mathbf{u}^3$ contains the result of the evaluation. Note that the middle step corresponds to a system solve with the matrix $\mathbf{A}$. The actual solution process can be approximated by, for example, a few cycles of multigrid method.

Let us study more closely the linear system (0.1). After a rearrangement of the variables, the system matrix has the block representation

$$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{B}_1 \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \mathbf{0} \\ \mathbf{B}_1^T & \mathbf{0} & \mathbf{0} \end{pmatrix}, \tag{2.1}$$

where the variables restricted by the Lagrange multipliers are numbered first. Thus, in the first evaluation phase $\mathbf{u}^1 = (\mathbf{u}_1^1, \mathbf{0})$ and in the middle step we actually solve

$$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1^2 \\ \mathbf{u}_2^2 \end{pmatrix} = \begin{pmatrix} \mathbf{u}_1^1 \\ \mathbf{0} \end{pmatrix}.$$

The final result $\mathbf{u}^3 = \mathbf{B}_1^T\mathbf{u}_1^2 + \mathbf{0}\mathbf{u}_2^2$ is independent of the vector block $\mathbf{u}_2^2$. Now, we want to create a special implementation of the algebraic multigrid method cycle in which the vector blocks $\mathbf{u}_2^1$ and $\mathbf{u}_2^2$ are not used since the first of them is zero vector and the second one is irrelevant for the evaluation. We call this special implementation a sparse algebraic multigrid.

For our model problem, we assume that the matrix $\mathbf{A}$ corresponds to a quasiuniform discretization mesh and the degrees of freedom corresponding to the first block row of (2.1) represent a lower dimensional plane with sufficient smoothness inside or on the boundary of a computational domain. Then, the number of components to be approximated is at most $\mathcal{O}(n^{(d-1)/d})$, where n is the total number of discretization nodes and d is the space dimension of the continuous problem.

Since the optimal computational complexity $\mathcal{O}(n)$ of the preconditioner for the problem (0.1) is wanted, the application of the sparse AMG should require at most $\mathcal{O}(n^{(d-1)/d})$ floating point operations. This is due to the fact that solving iteratively systems in which sparse AMG is used to perform matrix-vector multiplication, takes at most $\mathcal{O}(\log \frac{1}{\epsilon} n^{1/d})$ iterations to the relative accuracy ϵ using, for example, conjugate gradient or Chebyshev

method [15]. For this reason the multigrid cycle has to be performed in a subspace, whose dimension does not exceed $\mathcal{O}(n^{(d-1)/d})$.

In order to perform the multigrid cycle in a subspace, the iterative method for the sparse AMG must be such that the approximate solution is updated only in the neighborhood of the nodes, where the residual before smoothing was nonzero. This requirement excludes, for example, Gauss–Seidel smoothing. Here, the Jacobi iteration has been chosen. Moreover, if we study an ordinary geometric multigrid with linear interpolation for a discretized one-dimensional boundary value problem, we notice that the number of nodes on each coarser level is approximately halved. When W-cycle is used, level i of the multigrid is visited 2^{i-1} times. Such a multigrid cycle requires $\mathcal{O}(n \log n)$ floating point operations, where n is the size of the original (level 1) linear system. This is similar to approximating a Schur complement of a two-dimensional problem on a one-dimensional curve. Therefore, for the sparse algebraic multigrid, only V-cycle is allowed in order to meet the complexity requirements.

If conjugate gradient method is used to solve problems, where sparse AMG acts as a matrix vector product, we must ensure that the sparse AMG defines a symmetric and positive definite operator. For the choices made here this turns out to be simple. The (diagonal) iteration matrix of the Jacobi method is always symmetric and positive definite for symmetric and positive definite problems. All that is required, is to choose suitable relaxation parameters for Jacobi iteration on each multigrid level.

The initialization of the sparse AMG may take as much as $\mathcal{O}(n)$ operations. Therefore, it is possible to construct the sparse AMG after the full AMG is initialized. The Algorithm 2.1 constructs the sparse AMG out of the full AMG.

Algorithm 2.1. *Sparse AMG initialization*

1. *Give a subset T_1 of the nodes, where the approximate solution is required*
2. *Define a set S_1 which includes T_1 and all the nodes the smoother may update, when the residual is nonzero for nodes in the set T_1.*
3. *For all the levels $i = 1, 2, \ldots, \mathrm{L}$ do*
   4. *Pick the rows and columns of $\mathbf{A}_i$, indicated by set S_i, to the sparse system matrix $\mathbf{A}_i^{\mathrm{s}}$*
   5. *Define the set T_{i+1} such that $j \in T_{i+1}$ if $(\mathbf{R}_i)_{jk} \neq 0$ for some $k \in S_i$.*
   6. *Define a set S_{i+1} which includes T_{i+1} and all the nodes the smoother may update, when the residual is nonzero for nodes in the set T_{i+1}.*
   7. *Pick the columns of $\mathbf{R}_i$, indicated by S_i, and rows of $\mathbf{R}_i$, indicated by S_{i+1} to the sparse restriction matrix $\mathbf{R}_i^{\mathrm{s}}$.*
8. *End for.*

After the initialization, the sparse system matrices $\mathbf{A}_i^{\mathrm{s}}$ and the sparse restrictions $\mathbf{R}_i^{\mathrm{s}}$ are used to perform a single V-cycle of the multigrid with zero initial guess for a given vector.

Theorem 2.1. *Let there be a hierarchy of uniformly coarsened meshes M_i, $i = 1, \ldots, \mathrm{L}$ and a discrete partial differential operator $\mathbf{A}$ discretized on the mesh M_1, where the degrees of freedom corresponding to mesh M_i are numbered before the degrees of freedom corresponding to mesh M_{i-1}. Moreover, let the number of degrees of freedom for the mesh M_{L} be at most the threshold, which is given as a stopping criterion for coarsening in AMG 2. Then, the sparse cycle of AMG 2 corresponding to a lower dimensional plane with $\mathcal{O}(n^{(d-1)/d})$ degrees of freedom requires at most $\mathcal{O}(n^{(d-1)/d})$ arithmetic operations and memory locations for $d \geq 2$ and $\mathcal{O}(\log n)$ operations and memory locations for $d = 1$.*

Proof. Due to the coarsening strategy, the number of nodes corresponding to the lower dimensional plane on level i is $\mathcal{O}(n^{(d-1)/d}/2^{(d-1)i})$. They are a subset of the nodes for which the solution was required on level 1. For each node on each level, the number of neighboring nodes is at most the number of neighboring nodes for the corresponding node on level 1, which is bounded. Thus, the required number arithmetic operations and memory locations on level i is also $\mathcal{O}(n^{(d-1)/d}/2^{(d-1)i})$. Summing these up over the levels gives us the estimate presented above. □

Note, that while the previous theorem assures the complexity of sparse AMG 2 in certain cases, the same estimate holds for many other cases also, for example, for all the numerical tests presented in this paper.

For the initialization of the sparse BPX, the sets S_i can be defined as $S_i = T_i$ and the step 4 in the Algorithm 2.1 can be omitted.

3. Model operators for numerical tests

For numerical testing, two positive definite partial differential operators are used: the diffusion operator

$$\mathcal{L}(u) = -\alpha I + \nu \Delta u \tag{3.1}$$

and the linear elasticity operator for an isotropic material

$$\mathcal{E}(\mathbf{u}) = -2\mu \nabla \cdot \varepsilon(\mathbf{u}) - \lambda \nabla(\nabla \cdot \mathbf{u}). \tag{3.2}$$

The operators are discretized with the finite element method and therefore, corresponding weak formulations of the operators are needed. Let there be a domain $\Omega \subset \mathbb{R}^d$, $d = 2, 3$ such that the boundary $\partial\Omega = \Gamma_0 \cup \Gamma_1$ and that $\text{meas}(\Gamma_0) > \gamma > 0$. The function space V is defined by

$$V = \{\varphi \ : \ \varphi \in H^1(\Omega), \ \varphi = 0 \text{ on } \Gamma_0\} \cdot \tag{3.3}$$

Now, the weak bilinear form $L : V \times V \to \mathbb{R}$ corresponding to the diffusion operator (3.1) reads:

$$L(u, v) = \int_\Omega \alpha uv + \nu \nabla u \cdot \nabla v \, \mathrm{d}\Omega, \tag{3.4}$$

and the weak bilinear form $E : V^d \times V^d \to \mathbb{R}$ corresponding to the linear elasticity operators (3.2) reads:

$$E(u, v) = \int_\Omega \left(\sum_{i=1}^{d} \sum_{j=1}^{d} 2\mu \varepsilon_{ij}(u) : \varepsilon_{ij}(v) - \lambda \frac{\partial u_i}{\partial x_i} \frac{\partial v_j}{\partial x_j} \right) \mathrm{d}\Omega. \tag{3.5}$$

The finite element spaces of the discrete problems are piecewise linear and bilinear in two dimensions with triangular and quadrilateral elements, respectively and bilinear and trilinear in three dimensions with prismatic and brick elements, respectively. These spaces are subspaces of V. The components of the matrix block $\mathbf{A}$ in (0.1) are obtained by inserting the corresponding finite elements basis functions to the bilinear forms L and E. For the following numerical examples, the matrix $\mathbf{B} \in \mathbb{R}^{m \times n}$ has one nonzero element on each row corresponding to each node on the lower dimensional plane, on which the Schur complement $\mathbf{S} = \mathbf{B}\mathbf{A}^{-1}\mathbf{B}^T$ is approximated, and the value of the nonzero elements is always 1.

4. Numerical results

For measuring the accuracy of a Schur complement approximation, the smallest $\lambda_{\min}$ and the largest eigenvalue $\lambda_{\max}$ of the generalized eigenvalue problem $\mathbf{S}\mathbf{u} = \lambda \tilde{\mathbf{S}}\mathbf{u}$ are computed. The quality of the approximation is measured by the ratio of the eigenvalues, which is also the spectral condition number of the product matrix $\tilde{\mathbf{S}}^{-1}\mathbf{S}$. Since the complexity of the algebraic multigrid is not known exactly prior to its construction, it is useful to measure it also. For the three compared methods, including the sparse BPX, all the required data structures are packed into sparse matrices. Therefore, a common measure for the method complexity is the number of sparse matrix entries used in the method. To present more illustrative information, a solution method for the corresponding problem with Lagrange multipliers of the form (0.1) is constructed and the numbers of iterations and CPU-times in seconds are reported. The solution method is the preconditioned MINRES [10] with the

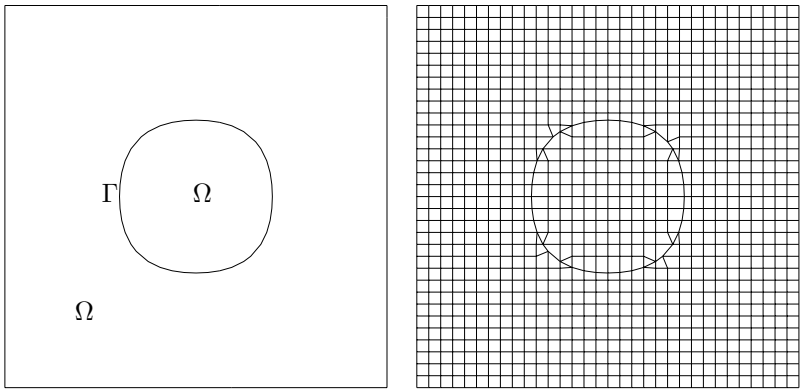

FIGURE 2. Computational domain and 32×32 mesh for the test problem 1.

TABLE 1. Condition numbers with respect to the mesh step size for the test problem 1.

	Sparse AMG 1				Sparse AMG 2				Sparse BPX			
h	cond	size	iter	time	cond	size	iter	time	cond	size	iter	time
1/16	1.13	1201	12	0.01	1.10	1054	14	0.01	1.47	125	16	0.01
1/32	1.19	4132	16	0.10	1.14	2818	14	0.07	2.06	225	22	0.04
1/64	1.24	8740	15	0.38	1.15	6452	14	0.29	2.35	541	22	0.29
1/128	1.37	27682	18	2.33	1.21	14628	16	1.67	2.25	1113	22	1.39
1/256	1.29	50119	16	8.47	1.21	29564	14	6.04	2.48	2205	22	6.17
1/512	1.41	140627	16	36.78	1.22	61524	14	27.23	2.28	4145	22	26.13

block diagonal preconditioner $\mathbf{P} = \mathrm{diag}(\mathbf{P}_\mathrm{p}, \mathbf{P}_\mathrm{l})$, where the primary variable preconditioner $\mathbf{P}_\mathrm{p}$ corresponds to one symmetric cycle of AMG 2 and the Lagrange multiplier preconditioner $\mathbf{P}_\mathrm{l}$ corresponds to the method under consideration. The initial guess is chosen to be zero and the right hand side vectors are $\mathbf{f} = \mathbf{0}$ and each component of vector $\mathbf{g}$ is one. The problem has a nontrivial solution.

As the first test problem the Laplace operator ((3.4) with $\alpha = 0$, $\nu = 1$) is discretized with topologically hierarchical finite element mesh in the unit rectangle Ω. Here, $\Gamma_0 = \partial\Omega$. The Schur complement is approximated on a closed curve Γ located inside the domain as in Figure 2. The curve is composed of four pieces of parabola, the first of which goes through points $(0.3, 0.5)$, $(0.35, 0.65)$ and $(0.5, 0.7)$. The other pieces are obtained by rotating the first around the midpoint $(0.5, 0.5)$. Nodes have been fitted locally on the curve to achieve an accurate representation of the curve. A mesh with 32×32 rectangles is also shown in Figure 2. The rectangles which have been cut by the curve have been split to two triangles. The mesh step sizes, condition numbers and the total number of sparse matrix entries of sparse AMGs and sparse BPX are presented in Table 1 as well as the numbers of iterations and CPU-times of the corresponding solution methods. The BPX is based on a hierarchical mesh, in which no local fitting is used. For the finest discretization, $h = 1/512$, of the first test problem, one V-cycle of the full AMG 1 took 0.661 s and had 3.26e6 sparse matrix entries, while one V-cycle of the full AMG 2 took 0.580 s of CPU-time and had 3.15e6 sparse matrix entries. In comparison, the sparse AMG 1 V-cycle took 1.88e-2 s, the sparse AMG 2 V-cycle 1.38e-2 s and the sparse BPX V-cycle 1.76e-4 s of CPU-time. The data structures of the sparse BPX fit into the cache memory. This shows that for simple problems the sparse BPX gives a good preconditioner with very quick computation. Only for some problem sizes the sparse AMG 2 is slightly more efficient.

As the next test, the Schur complement of Laplace operator is approximated on two opposite sides, numbered by 3 and 6, of a hexagon, while Γ_0 is composed of the other four sides, as in Figure 3. The discretization mesh is not hierarchical. In Table 2, the results are presented in a form which is similar to the first test problem.

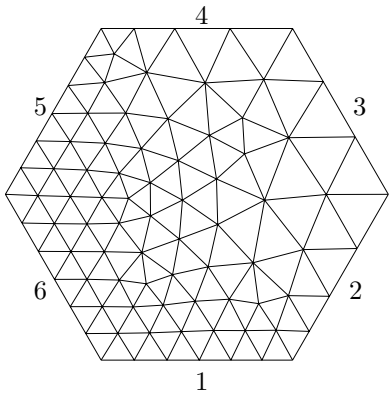

FIGURE 3. The coarsest finite element mesh for the test problem 2.

TABLE 2. Condition numbers with respect to number of nodes for the test problem 2.

	Sparse AMG 1				Sparse AMG 2			
nodes	cond	size	iter	time	cond	size	iter	time
86	1.11	693	14	0.01	1.06	468	14	0.01
301	1.15	1343	15	0.01	1.27	909	15	0.01
1156	1.20	5182	17	0.11	1.43	2516	18	0.09
4197	1.25	14763	17	0.58	1.95	5360	19	0.45
16452	1.29	44208	18	4.43	2.49	12007	25	2.84

TABLE 3. Condition numbers with respect to the mesh step size for the test problem 3.

	Sparse AMG 1				Sparse AMG 2				Sparse BPX			
h	cond	size	iter	time	cond	size	iter	time	cond	size	iter	time
1/16	6.71	6737	38	0.40	2.18	6641	23	0.16	14.9	125	47	0.07
1/32	9.00	18536	41	1.74	2.35	14165	22	0.71	16.5	225	54	0.53
1/64	27.0	76963	51	11.5	2.56	31894	24	3.41	17.6	541	55	2.67
1/128	92.0	233328	62	58.0	2.53	64261	23	15.1	16.6	1113	53	11.3
1/256	220.	965621	71	250.	2.61	127682	23	57.6	16.6	2205	52	46.4

For these unstructured meshes, the measured condition number is much better for AMG 1 than AMG 2, but the number of sparse matrix entries grows at greater rate in AMG 1. The condition number for AMG 2 clearly depends on the mesh step size h. This is due to the deteriorating quality of the interpolations. Nevertheless, the sparse AMG can be well used on unstructured meshes.

In the third test problem, the computational domain is the same as in the first test problem, but Γ_0 is only the bottom of the domain. The Schur complement of the linear elasticity operator (3.5) is approximated on the same curve Γ as in test problem 1. The Young modulus of the elastic material is 100 and the Poisson ratio 0.3 giving $\lambda \approx 57.7$ and $\mu \approx 38.5$. The sparse AMG 2 is compared to the sparse BPX, which is used for the two displacement components separately. The results are presented in Table 3. Also AMG 1 method was tested, using the unknown-approach described in [17], but the local fitting of the discretization mesh and underrelaxation needed to obtain the positive definiteness of the V-cycle greatly reduced the quality of the approximation. This experiment shows that sparse AMG 2, which is suited to the original problem directly, can give much better approximations to the Schur complement than sparse BPX, although the spectral properties of the linear elasticity operator are comparable to those of the vector Laplacian. However, when studying the

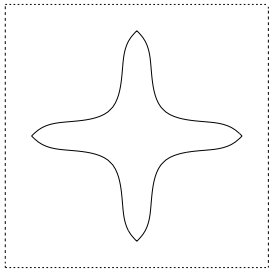

FIGURE 4. The profile of the cylinder for test problem 4.

TABLE 4. Condition numbers with respect to the mesh step size for the test problem 4.

	Sparse AMG 1				Sparse AMG 2				Sparse BPX			
h	cond	size	iter	time	cond	size	iter	time	cond	size	iter	time
1/16	2.44	41734	25	2.76	2.01	34768	22	2.84	14.0	2268	32	0.87
1/32	6.81	223586	40	33.8	2.01	161198	22	22.6	17.4	5925	30	7.03
1/64	10.0	998752	47	280.	1.93	643086	20	131.	28.0	32684	30	77.3

TABLE 5. Condition numbers with respect to a coefficient value for the test problem 5.

	Sparse AMG 1				Sparse AMG 2				Sparse BPX			
s	cond	size	iter	time	cond	size	iter	time	cond	size	iter	time
10^0	1.37	20834	17	2.41	1.29	14628	16	1.94	5.97	1113	25	1.80
10^1	1.40	31980	19	3.53	1.49	14628	19	2.39	8.10	1113	36	2.58
10^2	1.31	54778	20	5.55	1.58	14628	20	2.56	10.2	1113	52	3.73
10^3	1.40	63831	20	6.04	1.60	14628	20	2.56	10.6	1113	67	4.80

execution times of the corresponding linear solver we notice that the computational time of the preconditioner outweighs the accuracy. Using the sparse AMG to both components of the displacement separately, the efficiency compared to the sparse BPX behaves as in the test problem 1.

In the fourth test problem, the computational domain is the unit cube $]0,1[^3$. The Schur complement of the diffusion operator, (3.4) with $\alpha = 1$ and $\nu = 1$, is approximated on the surface of a cylinder with a star-shaped cross section profile as in Figure 4. The profile is composed of four pieces of natural cubic splines, first of which passes through points $(0.5, 0.1)$, $(0.45, 0.2)$, $(0.4, 0.4)$, $(0.2, 0.45)$ and $(0.1, 0.5)$. The other pieces are obtained by rotating the first around the midpoint $(0.5, 0.5)$ in $90°$ steps. The cylinder starts at $z = 0.3$ and ends at $z = 0.7$. The boundary Γ_0 is the bottom of the cube. The finite element mesh is a Cartesian and locally fitted one with cubic and prismatic first order elements. Some problems were encountered with AMG 1. In order to obtain a positive definite operator, some underrelaxation was needed and this is partly responsible for the growth of the condition number with respect to the mesh step size h.

In the fifth test problem, the approximation of the Schur complement is studied with respect to a jump in a coefficient. The setting of the problem is the same as for the first test problem, except that the discretization mesh is always the same, $h = 1/128$, and the coefficient ν of the weak diffusion operator (3.4) is $\nu = 1 + r$, where r is a random number of the interval $[0, s]$ chosen elementwise. Since the random numbers are accessed from a file, the problem for each method is exactly the same. The performance of the methods with respect to the varying scale factor s are given in Table 5. The results show that the sparse AMG 1 can maintain a high quality for the Schur complement approximation, but with growing expenses. This, of course, depends on the initialization parameter which is used to define a strong connection. If the parameter is small enough,

the sparse AMG 1 gives the same kind of slowly degrading approximation with almost constant computational cost as sparse AMG 2. The sparse BPX, which is not suited to this kind of problems, gives clearly the worst approximation for the Schur complement and large number of iterations to the system solver. The solver using the sparse BPX is slower than the one using sparse AMG 2, but due to its simple structure it is still faster than the method using the accurate sparse AMG 1 preconditioner.

5. Conclusions

Two sparse algebraic multigrid methods that were based on algorithms presented in [11] and [17] were tested for Schur complement approximation. They can be easily initialized if the full AMG is initialized first. For the tested problems, one or both of the methods give very good approximations for Schur complements with respect to condition numbers. The complexity of the sparse algebraic multigrids is considerably larger than that of the sparse BPX, and, therefore, the sparse BPX is more efficient for Laplace-like problems on structured meshes. For some more complex operators and problems on unstructured grids the sparse AMG method can be used for Schur complement approximation and for boundary Lagrange multiplier preconditioning.

Acknowledgements. The author wishes to thank Erkki Heikkola, Tuomo Rossi and Jari Toivanen for comments and suggestions concerning improvements of this article. The finite elements meshes for the test problem 2 were generated using EasyMesh (homepage http://www-dinma.univ.trieste.it/~nirftc/research/easymesh/easymesh.html), written by Bojan Niceno. This work was financially supported by The Academy of Finland, contract/grant number #53588.

References

[1] I. Babuška, The finite element method with Lagrangian multipliers. *Numer. Math.* **20** (1972/73) 179–192.
[2] C. Bernardi, Y. Maday and A.T. Patera, A new nonconforming approach to domain decomposition: the mortar element method, in *Nonlinear partial differential equations and their applications*. Collège de France Seminar, Vol. XI, Paris (1989–1991) 13–51. Longman Sci. Tech., Harlow (1994).
[3] J.H. Bramble, J.E. Pasciak and A.H. Schatz, The construction of preconditioners for elliptic problems by substructuring. I. *Math. Comp.* **47** (1986) 103–134.
[4] J.H. Bramble, J.E. Pasciak and Jinchao Xu, Parallel multilevel preconditioners. *Math. Comp.* **55** (1990) 1–22.
[5] Qianshun Chang, Yau Shu Wong and Hanqing Fu, On the algebraic multigrid method. *J. Comput. Phys.* **125** (1996) 279–292.
[6] M. Dryja, A capacitance matrix method for Dirichlet problem on polygon region. *Numer. Math.* **39** (1982) 51–64.
[7] R. Glowinski, T. Hesla, D.D. Joseph, T.-W. Pan and J. Periaux, Distributed Lagrange multiplier methods for particulate flows, in *Computational Science for the 21st Century*, M.-O. Bristeau, G. Etgen, W. Fitzgibbon, J.L. Lions, J. Periaux and M.F. Wheeler Eds., Wiley (1997) 270–279.
[8] R. Glowinski, Tsorng-Whay Pan and J. Périaux, A fictitious domain method for Dirichlet problem and applications. *Comput. Methods Appl. Mech. Engrg.* **111** (1994) 283–303.
[9] G.H. Golub and D. Mayers, The use of preconditioning over irregular regions, in *Computing methods in applied sciences and engineering VI*, Versailles (1983) 3–14. North-Holland, Amsterdam (1984).
[10] A. Greenbaum, *Iterative methods for solving linear systems*. SIAM, Philadelphia, PA (1997).
[11] F. Kickinger, Algebraic multi-grid for discrete elliptic second-order problems, in *Multigrid methods V*, Stuttgart (1996) 157–172. Springer, Berlin (1998).
[12] Yu.A. Kuznetsov, Efficient iterative solvers for elliptic finite element problems on nonmatching grids. *Russian J. Numer. Anal. Math. Modelling* **10** (1995) 187–211.
[13] Yu.A. Kuznetsov, Overlapping domain decomposition with non-matching grids. *East-West J. Numer. Math.* **6** (1998) 299–308.
[14] R.A.E. Mäkinen, T. Rossi and J. Toivanen, A moving mesh fictitious domain approach for shape optimization problems. *ESAIM: M2AN* **34** (2000) 31–45.
[15] J. Martikainen, T. Rossi and J. Toivanen, Multilevel preconditioners for Lagrange multipliers in domain imbedding. *Electron. Trans. Numer. Anal.* (to appear).
[16] G. Meurant, A multilevel AINV preconditioner. *Numer. Algorithms* **29** (2002) 107–129.
[17] J.W. Ruge and K. Stüben, Algebraic multigrid. SIAM, Philadelphia, PA, *Multigrid methods* (1987) 73–130.
[18] D. Silvester and A. Wathen, Fast iterative solution of stabilised Stokes systems. II. Using general block preconditioners. *SIAM J. Numer. Anal.* **31** (1994) 1352–1367.
[19] C.H. Tong, T.F. Chan, and C.-C. Jay Kuo, A domain decomposition preconditioner based on a change to a multilevel nodal basis. *SIAM J. Sci. Statist. Comput.* **12** (1991) 1486–1495.