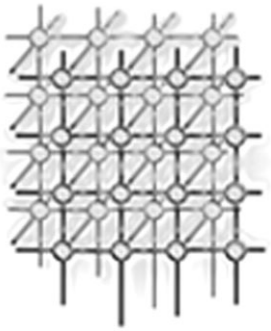
CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2006; **18**:501–518
Published online 13 September 2005 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe.940

Parallel iterative multilevel solution of mixed finite element systems for scalar equations



V. Chugunov, D. Svyatski, E. Tyrtyshnikov and Yu. Vassilevski*,†

Institute of Numerical Mathematics, Russian Academy of Sciences, 8 Gubkina Street, 119991, GSP-1, Moscow, Russia

SUMMARY

A combination of several contemporary techniques is used for the efficient parallel solution of the mixed finite element systems on locally refined Grids. Implementation experience and numerical results are reported. Copyright © 2005 John Wiley & Sons, Ltd.

KEY WORDS: mixed finite elements; multilevel preconditioner; locally refined meshes; parallel computations

1. INTRODUCTION

In the last decade, the mixed finite element method (MFEM) [1] has become a very popular technique for conservative approximations. It has the advantages of a solid mathematical foundation, simplicity of implementation, applicability to unstructured meshes and full dispersion tensors. The appealing features of the method led to extensive use in computational practice. This poses new challenges, such as the efficient solution of the sparse linear systems generated by MFEM. Of particular interest is the demand of the robust efficient solvers of MFEM systems associated with unstructured locally refined meshes. The conservative approximations on unstructured meshes is a very powerful tool for tackling many engineering applications.

The efficient solution of unstructured systems has been a subject of research in different fields of numerical analysis. Recent developments in *LU*-factorizations [2,3] for sparse matrices extend their applicability (on modern PCs) to systems with up to 100 000 unknowns. Of course, in the limit case the factorization time is very large and the direct methods are not competitive to iterative solvers.

*Correspondence to: Yu. Vassilevski, Institute of Numerical Mathematics, Russian Academy of Sciences, 8 Gubkina Street, 119991, GSP-1, Moscow, Russia.
†E-mail: vasilevs@dodo.inm.ras.ru

Contract/grant sponsor: Upstream Research Center, ExxonMobil Corporation

Received 26 November 2003
Revised 16 June 2004
Accepted 17 October 2004

Another class of *black box* solvers are Krylov subspace iterations (PCG, GMRES, BiCGstab) with incomplete *LU* preconditioners [4]. Being very fast in implementation, they suffer from slow convergence which deteriorates with subsequent refinement of the mesh. This is particularly important for locally refined meshes where the refinement may be very deep. The fictitious space technique provides an elegant method for efficient preconditioning of the unstructured systems by interpolating data onto a hierarchical locally refined Grid [5,6]. The drawback of this approach is the strong assumptions on the problem data (coefficients). An alternative approach to black box preconditioning unstructured systems is the *algebraic multigrid* (AMG) technology [7,8]. Being more expensive at the initialization stage, AMG methods provide (upon successful initialization) a better convergence rate which is often independent of the mesh size. The key issues for any AMG method are separation into 'fine' and 'coarse' degrees of freedom and the choice of the prolongation operator. Typically, the 'coarse' level matrix is the Galerkin projection of the matrix associated with the 'fine' level. The separation of degrees of freedom is based on the graph technologies applied to the matrix graph, which exploit [9] or do not exploit [10] matrix entries. The choice of the prolongation operator in AMG methods is the crucial issue involving the basic assumptions for the method, its type and its features. Being true black box solvers, the first AMG methods (e.g. [9]) have used the matrix entries only for the choice of prolongation operator. The price to be paid for the limited input information is a strong assumption on the input matrices (e.g. M-matrices). The modern trend in the development of AMG technologies is to relax assumptions on the input matrices requiring more input data. For instance, the input matrix may be given in a disassembled form that is very typical for the finite element applications. The methods (AMGe) [11,12] exploiting element matrices are more robust and less independent of problem data. Parallel implementations of the considered methods are also presented in the literature. We mention [2,13] for parallel *LU* factorization and [4,14] for parallel *ILU* factorization. The parallelization of AMG methods is not technically simple. However, successful implementations are known [15].

In this paper we report our experience in using MFEM for the approximate solution of scalar elliptic equations on unstructured locally refined meshes. Our implementation consists of several modern technologies used in applied numerical mathematics. The discretization is based on conformal triangular or tetrahedral hierarchical meshes. Although the meshes may be refined locally, their shape regularity is guaranteed. The hierarchy and local refinement are provided by the marked edge bisection algorithm [16,17]. The MFEM problem is reformulated in the equivalent hybrid counterpart [18,19]. The only data required by this cost-effective procedure are the global matrix in the disassembled form (local MFEM matrices) and Grid node coordinates together with the connectivity table. These data are available in all finite element method (FEM) applications.

The associated saddle-point system is reduced by a sequence of local modifications to a smaller system with a sparse symmetric positive definite matrix with at most seven (five in two dimensions) non-zero entries per row. Therefore, the reduction of MFEM saddle-point systems to more compact systems with symmetric positive definite matrices may be performed element-wise with a simple inexpensive technology. The reduced system is solved by the preconditioned conjugate gradient (PCG) method with a multilevel preconditioner suggested in [20].

The method has a solid theoretical background and its V-cycle formulation with one smoothing per level is shown to be robust with respect to the problem data. The latter property implies efficient (with optimal order of arithmetical complexity) preconditioning on locally refined meshes. We mention another work [21], where the V-cycle is analyzed in the case of an interface operator

arising in the domain decomposition method applied to MFEM systems on uniformly refined meshes. The preconditioner [20] is based on the equivalence [18,19,22] between the reduced system and the P_1 non-conforming finite element approximation. Therefore, a multigrid preconditioner for the non-conforming FE matrices may be used in the solution of the reduced MFEM system. Similarly to the AMG methods, the Grid-level matrices are built recursively as Galerkin projections of the fine Grid matrix at the initialization stage. The simple prolongation operator is constructed on the basis of Grid hierarchy data and is natural for the considered non-conforming finite element spaces. No assumptions on smooth error characterization and no matrix-entries analysis are needed for the prolongation operator derivation, in contrast to the AMG techniques. Thus the multigrid method may be considered as a *gray box* using the MFEM reduced system and Grid data on input (logical hierarchy for defining the prolongation operators). In comparison with the *LU* and *ILU* factorization techniques, the presented method features a very good convergence rate and small initialization time; in comparison with the AMG methods, it avoids the basic issues of AMG initialization, the separation of unknowns and the construction of the prolongation, due to the input of logical hierarchy of the Grid. Similarly to AMGe methods, it inputs the element matrices in the disassembled form; however, it uses them for the reduction to the system with smaller matrix and simpler structure. An important feature of the method is that the sparsity patterns of level matrices differ: the matrix on a coarser Grid level is more dense than that on a finer one. This may result in higher than linear (with respect to the number of unknowns) storage requirements and the arithmetical complexity. We consider both the original preconditioner [20] and its economical modification filtering-off some non-zero entries of the level matrices. The latter technique has much in common with the *ILU* factorization methods. Our approach to method parallelization is based on a simple partition strategy [23,24] and a particular data structure [25]. These choices provide a very simple parallel implementation on computers with moderate number of processors. The approach enables us to obtain feasible speed-ups on computers with distributed memory even on highly irregular Grids despite the deterioration of sparsity of the level matrices. We note that the presence of deep local refinements aggravates the arithmetical load balance on coarse levels and the sparsity deterioration increases the weight of interprocessor communications. Both factors complicate an efficient parallel realization. The novelty of our approach is its *gray box* character: it adopts the MFEM systems in the disassembled form and uses the Grid data (node coordinates for the reduction to symmetric positive definite (SPD) systems and logical hierarchy for defining the prolongation operators). This distinguishes it from the black box technologies (*AMG*, *LU*, *ILU*) and the integrated systems [26–28].

We note that deterioration of the sparsity patterns of level matrices is the new observation in MFEM multilevel technologies, and filtering-off some non-zero entries of the level matrices is the important modification of the original method [20]. Previous numerical examinations of the multigrid method [20] on unstructured meshes are not known to the authors. Regarding our parallel implementation, our simple (slice-wise) partition strategy and data structure allowed us a simple realization and feasible efficiency for moderate numbers of processors. It is pertinent to remark that clusters with a moderate (up to 20) number of processors are very popular in the computational community and our technology may be useful in many cases.

This paper is outlined as follows. After a brief formulation of the model problem and its mixed finite element discretization in Section 2, we give a formal description of the multilevel method and its possible modification in Section 3. In Section 4, we discuss issues of parallelization: our principles of parallelization, partitioning strategy and basic data structures. Numerical experiments are presented in Section 5.

2. MODEL PROBLEM AND MIXED FINITE ELEMENT APPROXIMATION

We consider the elliptic problem

$$\begin{aligned} -\nabla\cdot(\mathcal{A}\nabla u)+u &= g \text{ in } \Omega \\ u &= 0 \text{ on } \Gamma_0 \\ \frac{\partial u}{\partial \nu_{\Gamma_1}} &= 0 \text{ on } \Gamma_1 \end{aligned} \tag{1}$$

where Ω is a bounded polygonal (polyhedral) domain in $\mathbb{R}^d$, $d=2,3$, with the boundary $\partial\Omega$, Γ_0 is a closed subset of $\partial\Omega$, $\Gamma_1=\partial\Omega\setminus\Gamma_0$, $g(x)\in L_2(\Omega)$ and $\mathcal{A}(x)$ is a uniformly positive definite symmetric tensor:

$$\xi^T\mathcal{A}(x)\xi\geq a_0\xi^T\xi,\quad x\in\Omega,\ \xi\in\mathbb{R}^d,\ a_0>0$$

Problem (1) may be reformulated in the dual mixed form as follows. Find the pair $(\sigma,u)\in V\times W$:

$$\begin{aligned} (\mathcal{A}^{-1}\sigma,v)-(u,\nabla\cdot v) &= 0 \quad \forall v\in V \\ -(\nabla\cdot\sigma,w)-(u,w) &= -(g,w) \quad \forall w\in W \end{aligned} \tag{2}$$

where

$$V=\{v\in(L_2(\Omega))^d:\ \nabla\cdot v\in L_2(\Omega),\ (v,\nu_{\Gamma_1})=0 \text{ on } \Gamma_1\},\quad W=L_2(\Omega)$$

Formulation (2) is the basis for the mixed finite element approximation of (1). Given a conformal simplicial mesh $\mathcal{E}$ of Ω, we approximate (1) by the mixed finite method [1] as follows. Find $(\sigma^h,u^h)\in V^h\times W^h$:

$$(\mathcal{B}^h\sigma^h,v)-(u^h,\nabla\cdot v)=0\quad\forall v\in V^h \tag{3}$$

$$-(\nabla\cdot\sigma^h,w)-(u^h,w)=-(g,w)\quad\forall w\in W^h \tag{4}$$

where $\mathcal{B}^h=\mathcal{P}^h\mathcal{A}^{-1}$ (component by component) and $\mathcal{P}^h$ is the L_2-projection onto W^h. The spaces V^h, W^h are the lowest order Raviart–Thomas space defined via restrictions of V^h, W^h on a simplex E:

$$\begin{aligned} V^h(E) &= J_{\hat{E}\to E}(V^h(\hat{E})) \\ W^h(E) &= P_0(E) \\ V^h(\hat{E}) &= (P_0(\hat{E}))^d\oplus((x,y,z)P_0(\hat{E})) \end{aligned} \tag{5}$$

Here, $J_{\hat{E}\to E}(V^h(\hat{E}))$ is a linear mapping of the Raviart–Thomas space $V^h(\hat{E})$ defined on a reference simplex $\hat{E}$.

The problem (3)–(4) is reformulated in a hybrid form. To this end, we define the spaces

$$\tilde{V}^h=\{v\in(L_2(\Omega))^d:\ v|_E\in V^h(E),\ \text{for each } E\in\mathcal{E}\}$$

$$L^h=\left\{\mu\in L_2\left(\bigcup_{f\in\partial\mathcal{E}}f\right):\mu|_f\in V^h\cdot\nu|_f,\ \text{for each } f\in\partial\mathcal{E}\right\}$$

Here $\partial\mathcal{E}$ denotes the set of all interior faces and $\nu|_f$ denotes a unit normal to f.

The hybrid form of the mixed method is as follows. Find the triple $(\sigma^h, u^h, \lambda^h) \in \tilde{V}^h \times W^h \times L^h$:

$$(\mathcal{B}^h\sigma^h, v) - \sum_{E\in\mathcal{E}}[(u^h, \nabla\cdot v)_E - (\lambda^h, v\cdot\nu_E)_{\partial E\setminus\partial\Omega}] = 0 \quad \forall v \in \tilde{V}^h \tag{6}$$

$$-\sum_{E\in\mathcal{E}}(\nabla\cdot\sigma^h, w)_E - (u^h, w) = -(g, w) \quad \forall w \in W^h \tag{7}$$

$$\sum_{E\in\mathcal{E}}(\sigma^h\cdot\nu_E, \mu)_{\partial E\setminus\partial\Omega} = 0 \quad \forall \mu \in L^h \tag{8}$$

where $\nu|_E$ denotes the outer unit normal to the faces of E.

The matrix form of (6)–(8) is a system

$$\begin{pmatrix} A & B & C \\ B^{\mathrm{T}} & -D & O \\ C^{\mathrm{T}} & O & O \end{pmatrix}\begin{pmatrix}\sigma\\ u\\ \lambda\end{pmatrix} = \begin{pmatrix} O\\ G\\ O\end{pmatrix} \tag{9}$$

The block structure of matrices A and $B^{\mathrm{T}}A^{-1}B + D$ allows us to eliminate u and σ from (9) and reduce (9) to a smaller system [18,19,22]

$$[-C^{\mathrm{T}}A^{-1}B(B^{\mathrm{T}}A^{-1}B + D)^{-1}B^{\mathrm{T}}A^{-1}C + C^{\mathrm{T}}A^{-1}C]\lambda = C^{\mathrm{T}}A^{-1}B(B^{\mathrm{T}}A^{-1}B + D)^{-1}G$$

or

$$M\lambda = G' \tag{10}$$

The matrix M is sparse (there are at most seven (five in two dimensions) non-zero entries per row), symmetric and positive definite. Its order is equal to the number of faces in $\partial\mathcal{E}$.

3. MULTILEVEL PRECONDITIONER

As already mentioned, simplicial meshes (conformal triangular or tetrahedral partitionings) are used in the MFEM approximation of differential equations. The presence of solution peculiarities may require local mesh refinement. MFEM theory is based on the shape regularity of the mesh. Therefore, regular meshes providing local refinement should be used. On the other hand, the resulting algebraic systems have to be solved very efficiently. Multilevel techniques seem to be the best methods for solving many boundary value problems. They are based on a hierarchy of the Grids and regular hierarchical unstructured meshes are the best candidates for the announced objectives.

Such meshes may be obtained, for example, by the marked edge bisection algorithm [16,17,29]. The algorithm possesses several features that make it very attractive in practical applications. It provides a moderate growth of the number of elements due to one-level refinement: the number of elements is at most doubled. It is adimensional: both two- and three-dimensional meshes are produced by the same topological splitting, bisection of a coarse simplex into two finer ones. It preserves the shape regularity: if the coarsest mesh consists of shape regular simplexes, all of the fine Grids will be shape regular as well. It provides coarsening opportunities: the refined Grids may be coarsened back [29].

In addition to standard simplicial mesh data, the marked edge bisection algorithm requires a specification how to partition the mesh in the subsequent $d-1$ refinements. The consistent partitioning makes the initial mesh subject to certain restrictions. A possible class of appropriate meshes is determined by a consistency condition. A mesh satisfies the consistency condition if the set of its simplexes $\mathcal{L}$ can be split into non-intersecting subsets $\mathcal{L}_i$ such that: (a) all simplexes from $\mathcal{L}_i$ share a common edge r_i that belongs to simplexes from $\mathcal{L}_i$ only; (b) in any subset $\mathcal{L}_i$ whose associated edge r_i is not a boundary edge, the number of simplexes is even[‡]. We remark that the meshes resulting from logically rectangular Grids by partitioning their cells into two (two dimensions) or six (three dimensions) simplexes, do satisfy the consistency condition.

Let a sequence of nested simplicial meshes be generated by the marked edge bisection algorithm:

$$\mathcal{E}_0, \mathcal{E}_1, \ldots, \mathcal{E}_K \equiv \mathcal{E} \tag{11}$$

We associate with each mesh $\mathcal{E}_l$, $l = 0, 1, \ldots, K$, the discrete space

$$N_l = \{v \subset L_2(\Omega),\ v|_E \in P_1(E), \forall E \in \mathcal{E}_l,\ v \text{ is continuous at the baricenters of interior faces and vanishes at the baricenters of faces on } \Gamma_0\}$$

Note that the spaces N_l are non-nested, i.e. $N_{l-1} \not\subset N_l, l = 1, \ldots, K$. We define a coarse-to-fine inter-Grid transfer operator $I_l : N_{l-1} \to N_l, l = 1, \ldots, K$ as follows. Let $v \in N_{l-1}$ and q be the baricenters of a face of a simplex in $\mathcal{E}_l$, then $I_l v \in N_l$ is defined by

$$(I_l v)(q) = \begin{cases} 0, & \text{if } q \in \Gamma_0 \\ v(q), & \text{if } q \in \Gamma_1 \\ v(q), & \text{if } q \notin \partial E \text{ for any } E \in \mathcal{E}_{l-1} \\ \frac{1}{2}\{v|_{E_1}(q) + v|_{E_2}(q)\}, & \text{if } q \in \partial E_1 \cap \partial E_2 \text{ for some } E_1, E_2 \in \mathcal{E}_{l-1} \end{cases} \tag{12}$$

The fine-to-coarse inter-Grid transfer operator $P_l : N_l \to N_{l-1}, l = 1, \ldots, K$, is defined by I_l:

$$(P_l v, w)_{l-1} = (v, I_l w)_l \quad \forall w \in N_{l-1},\ v \in N_l,\ l = 1, \ldots, K \tag{13}$$

where

$$(v, w)_l := \sum_q v(q) w(q) \quad v, w \in N_l,\ l = 0, \ldots, K$$

Next, we define discrete operators on $N_l, l = 0, \ldots, K$, by use of induction:

$$M_K := M; \quad M_l = P_{l+1} M_{l+1} I_{l+1}, \quad l = K-1, \ldots, 0 \tag{14}$$

Finally, we denote the halved diagonal reciprocal part of M_l by R_l:

$$R_l = \tfrac{1}{2}(\operatorname{diag} M_l)^{-1}, \quad l = 0, \ldots, K$$

which corresponds to the Jacobi smoother in the multilevel preconditioner to be used.

The simplest version of the algorithm [20] is presented in the form of a pseudo-code. Let $v \in N_K$ be given as the action on v of the preconditioner B_K whose action on v is as follows:

[‡]We note that in the two-dimensional case this requirement is satisfied for any mesh since only two triangles can share a common interior edge.

Pseudo-code 1.

1. Set $u_K = v$.
2. For $l = K, K-1, \ldots, 1$: $x_l = R_l u_l$, $u_{l-1} = P_l(u_l - M_l x_l)$.
3. $w_0 = M_0^{-1} u_0$.
4. For $l = 1, 2, \ldots, K$: $y_l = x_l + I_l w_{l-1}$, $w_l = y_l + R_l(u_l - M_l y_l)$.
5. Set $B_K v = w_K$.

This is a V-cycle with one pre- and one post-smoothing Jacobi iteration. We use the preconditioner B_K in the preconditioned conjugate gradient method applied to (10).

The general framework of the convergence analysis [20] of the multilevel method is done in terms of the inequality

$$a_h((I - B_K M_K)v, v) \leq \delta_K a_h(v, v) \quad \forall v \in N_K$$

where $a_h(\cdot, \cdot)$ denotes the bilinear form associated with (1). The estimate of δ_K yields the convergence rate of the multilevel method. Although only two-dimensional applications have been considered in [20], the analysis may be extended to the three-dimensional case as well. In the two-dimensional case of the full elliptic regularity [30] of problem (1), δ_K is proven to be bounded from above by a constant $\delta < 1$. The convergence result without any regularity assumption is weaker: δ_K is proven to be bounded by $1 - 1/CK$ where C is independent of K.

From a practical standpoint, the recurrency (14) results in an undesirable effect: the sparsity of matrices M_l deteriorates as l reduces. The number of non-zero entries in a row may be as large as several hundreds in the three-dimensional case, while M_K has at most seven non-zero entries in a row. This results in large memory requests for the storage of matrices M_l. We have found that some non-zero entries may be neglected for the storage in many (not all) cases. Actually, they contribute important information to the recurrent generation of the matrix at the coarser level. However, every so often their influence is negligible after the initialization step. Taking into account the above considerations, it is natural to introduce the threshold parameter τ into the algorithm. The non-zero value of the threshold shows some off-diagonal entries of the level matrices (except M_K and M_0) to be filtered off. The thresholding essentially reduces memory requirements but can affect the convergence of the iterative solver. The optimal value of the threshold seems to be a compromise between the memory requests and the iteration convergence rate. The choice $\tau = 0$ preserves all non-zero off-diagonal entries and provides fast convergence, albeit requiring quite a large area to store all of the level matrices. The non-zero value of τ filters off those off-diagonal entries ${M_l}^{ij}$ of the intermediate level matrices M_l: (a) whose associated faces belong to the same coarsest simplex; and (b) $|{M_l}^{ij}| < \tau |{M_l}^{ii}|$. Any filtered entry is added to the diagonal entry from the same row, in order to preserve the row sum. This is very important in the absence of a mass term in M. The condition (a) makes the filtering technology robust in the case of jumping coefficients. We note that due to the recurrent definition (14) of the level matrices, the filtering should be applied to M_{l+1} after M_l has been generated.

4. PARALLELIZATION OF THE MULTILEVEL METHOD

The parallel algorithm we want to discuss complies with the following principles. The parallel algorithm is identical mathematically to its sequential counterpart (Pseudo-code 1). On parallel computers

with a moderate number of processors (less than 20), it provides an essential speed-up for both uniform and locally refined meshes. It is implemented identically for two- and three-dimensional meshes. The time of separation of mesh subdomains associated with processors is proportional to the number of faces in the finest mesh. The number of interior faces is far more than the number of interfaces. The number of unknowns (faces) is uniformly balanced over the processors. A work memory of the algorithm is proportional to the number of unknowns.

The fine Grid $\mathcal{E}$ is partitioned in a set of non-intersecting subdomains that are topological strips (slices in three dimensions):

$$\mathcal{E}^1,\ \mathcal{E}^2, \ldots, \mathcal{E}^{n_p}$$

where n_p stands for the number of processors. Each subdomain is associated with a processor. To this end, we take advantage of the inertial bisection algorithm [23,24], which may be described as follows. Given a mesh with n_s simplexes, we associate to each simplex a unit mass located at its baricenter x^k and define the inertia tensor by

$$T_{ij} = \sum_{k=1}^{n_s} \delta_{ij} |\mathbf{y}^k|^2 - y_i^k y_j^k, \quad i, j = 1, 2, \quad \mathbf{y}^k = \mathbf{x}^k - \sum_{l=1}^{n_s} \mathbf{x}^l / n_s$$

The partitioning axis is computed as the eigenvector $\mathbf{v}$ associated to the smallest eigenvalue of T, i.e. the principle direction with the smallest inertia moment. Then the simplexes are colored for partitioning as follows: sort the points in ascending order of $(\mathbf{y}^k, \mathbf{v})$; define the mass stride $\Delta m = n_s/n_p$; color the point by the integer part of $k/\Delta m$. The resulted subdomains are topological strips since the above partitioning may be thought of as cutting $\mathcal{E}$ by $n_p - 1$ planes orthogonal to the partitioning axis.

Such a decomposition complies with our requirements but, more importantly, allows us to easily define local interpolation and restriction operators associated with the Grid subdomains (the importance of the strip-wise partitioning for the implementation stage is discussed at the end of this section). The role of the intergrid transfer operators (interpolation and restriction) is two-fold. On the one hand, at the initialization stage, they are used in the computation of the Galerkin projections (14) onto coarse Grids. On the other hand, they are used in the evaluation of the preconditioner. In order to provide the mathematical equivalence between the parallel and sequential algorithms, the parallel intergrid transfer operators must be identical to their sequential counterparts. Their evaluation on a processor should be as independent of data in the other processors as possible, for the sake of high parallel efficiency. To this end, we first arbitrarily enumerate the faces of the coarse mesh $\mathcal{E}_0$. When the fine mesh $\mathcal{E}_l$ is generated by the bisection refinement of some simplexes from $\mathcal{E}_{l-1}$, $l = 1, \ldots, K$, each newly appearing face is assigned an incremented index. For any face r of mesh $\mathcal{E}_i$ we define a set $\mathcal{K}_r$ as a minimum subset of faces $\{1, \ldots, r-1\}$ yielding a completely defined right-hand side of (12) for computing $I_l v$ at the face r. Secondly, for any subgrid $\mathcal{E}^i$, $i = 1, \ldots, n_p$, we associate such a sequence of coarsening Grids $\mathcal{E}_l^i$, $l = K-1, \ldots, 0$, so that $\mathcal{E}_l^i$ contains the sets $\mathcal{K}_r$ for any face $r \in \mathcal{E}_{l+1}^i \setminus \mathcal{E}_l^i$, $l = K-1, \ldots, 0$. The subgrids $\mathcal{E}_l^i$ may be constructed as follows. The subgrid $\mathcal{E}_K^i$ coincides with $\mathcal{E}^i$, and the subgrid $\mathcal{E}_l^i$, $l = K-1, \ldots, 0$, is obtained from $\mathcal{E}_{l+1}^i$ by the elimination of faces belonging to $\mathcal{E}_{l+1} \setminus \mathcal{E}_l$ and the addition of the sets $\mathcal{K}_r$ associated with the eliminated faces. Subgrids of a coarse level $l < K$ may overlap due to a dilation of coarser subgrids.

The sequence $\mathcal{E}_l^i$, $l = 1, \ldots, K$, available at processor i, $i = 1, \ldots, n_p$ provides data for local evaluation of the interpolation operator I_l^i, according to (12), and almost local evaluation of the restriction operator P_l^i. Indeed, for any face $r \in \mathcal{E}_l^i$, the interpolation operator depends on values

associated with $\mathcal{K}_r$, whose elements belong to processor i by definition of $\mathcal{E}^i_{l-1}$. It allows us to evaluate the interpolation operator I^i_l on processor i without communications with other processors. The evaluation of the restriction operator P^i_l implies the accumulation of contributions of values at faces r into the elements of $\mathcal{K}_r$. It results in communications between processors whose sets $\mathcal{K}_r$ overlap [25].

The level matrices M_l occupy the largest part of data storage. Therefore, they are distributed among processors according to the partition into the subgrids. The fine Grid matrix M_K is represented as the sum of subdomain matrices M^i_l, $i = 1, \ldots, n_p$. They are assembled from the local matrices computed on simplexes from the respective subgrid $\mathcal{E}^i_K$.

The matrices of other levels M_l, $l = K-1, \ldots, 1$, are computed as the Galerkin projections (14) onto the finite element spaces for the associated Grid $\mathcal{E}_l$. The restrictions M^i_l of matrices M_l onto subgrids $\mathcal{E}^i_l$ are computed via the restrictions M^i_{l+1} of M_{l+1} and the local inter-Grid transfer operators P^i_{l+1}, I^i_{l+1}: $M^i_l = P^i_{l+1} M^i_{l+1} I^i_{l+1}$, $l = K-1, \ldots, 1$. Of course, the multiplication is performed on each processor independently of others, due to the local definition of M^i_l. We note that the set of faces contributing to the matrix M^i_{l-1} deviates from the same set for M^i_l no more than the mesh size in $\mathcal{E}_{l-1}$.

In order to describe the multiplication of matrix M^i_l with a vector, we introduce types of mesh faces. A face of level l is *interior* if it belongs to a single subgrid $\mathcal{E}^i_l$; it is a *local interface* if it is shared by two subgrids $\mathcal{E}^i_l$, $\mathcal{E}^j_l$, with $|i-j| = 1$; otherwise, it is *global*. A global face associated with a pair of subgrids is considered to be global for *all* subgrids of the current level. The M^i_l matrix–vector product is computed differently for different types of faces. For the interior faces, no communications are needed since only local data contribute to the matrix–vector product. For the local interface degrees of freedom, communications of the host processor with the physical neighbors $i-1 \leftarrow i \rightarrow i+1$ occur. The length of the messages is bounded by the number of local interface faces. In order to compute the matrix–vector product in the global faces, the global (all-to-all) communications are required. The parallel efficiency depends on the number of the global faces: the smaller it is, the more scalable matrix–vector multiplication becomes. The consistent classification of the face types may be performed on the basis of the sequence $\mathcal{E}^i_l$, $l = K, \ldots, 1$ and a minimal number of local interface and global communications. The detailed presentation of the algorithm is rather tedious [25] and is omitted here. It is pertinent to note, however, that the strip-wise partitioning of the fine Grid provides the consistent and efficient parallel classification of face types. We do not know how to extend the parallel classification algorithm to general partitions [31].

5. NUMERICAL RESULTS

The numerical experiments have been performed on a Beowulf system consisting of 16 dual AMD Athlon (1.6 GHz) processors. Interprocessor communications are based on MPI library calls. First, we examine two-dimensional cases. We consider problem (1) with $\Gamma_1 = \emptyset$ (homogeneous Dirichlet boundary condition), identity matrix $\mathcal{A}$, $f = 1$, and domain $\Omega = (0, 1)^2$. The coarsest mesh $\mathcal{E}_0$ consists of two triangles separated by the diagonal containing the point $(1; 0)$. The abbreviations #*it* and *cnd* stand for the number of iterations and the condition number of the preconditioned matrix, respectively, n_K stands for the number of faces (the order of the matrix M_K) and t denotes the time (in seconds) of PCG iterations.

Table I. Two-dimensional case: performance of PCG iterations for meshes with different types of local refinement.

	K											
	13				15				17			
Mesh	n_K	*#it*	*cnd*	*t*	n_K	*#it*	*cnd*	*t*	n_K	*#it*	*cnd*	*t*
Uniform	24 704	12	2.9	0.7	98 560	14	3.4	3.6	393 728	16	3.9	18.3
Ref. in a sbdm	6585	12	2.5	0.2	25 440	13	3.0	0.9	100 007	15	3.5	4.0
Ref. to a curve	1460	9	1.8	0.1	2986	9	1.8	0.1	6048	10	1.9	0.2
Ref. to a point	608	8	1.7	<0.01	1172	8	1.7	<0.01	2432	9	1.9	0.1

We begin with single processor experiments. In Table I we present the performance features[§] on various meshes, see Figure 1. The number of iterations increases slowly as the number of unknowns grows. However, the condition number saturates for locally refined meshes. For uniform refinements in (sub)domain we observe proportionality of the condition number to the number of levels. However, the condition numbers are very small (less than four) and four-fold increase of n_K causes the increment of the condition number by 0.5. However, it is not clear whether the saturation of the condition number will occur on meshes with further uniform refinement. The time per iteration is almost linear, which is indicative of the suboptimal order of arithmetical complexity.

The case of jumping scalar permeability coefficients is considered in Table II: $\mathcal{A}(x, y) = a(x, y)\mathcal{I}$, where

$$a(x, y) = \begin{cases} \mathbf{a} & 1 - x > y \\ 1 & 1 - x < y \end{cases}$$

The jumps slightly affect the condition number and the convergence rate: halving the mesh size increases the condition number by approximately 30%. On the other hand, on a fixed mesh, the condition number stabilizes as the jump value tends to infinity. Remarkably, the number of iterations decreases for very large jumps, which may be explained by a clusterization of eigenvalues and the properties of the conjugate gradient method.

In Table III we allow the anisotropic tensor to jump and change the principal direction

$$\mathcal{A} = \begin{cases} \mathrm{diag}(\mathbf{a}, 1) & 1 - x > y \\ \mathrm{diag}(1, \mathbf{a}) & 1 - x < y \end{cases}$$

In addition, we repeat the experiment on the same Grid which is rotated by about 20°. The center of the rotation coincides with the baricenter of the Grid. This simulates the full diffusion tensor. The presence of anisotropy in the permeability tensor deteriorates the convergence rate to a greater extent: an increase of the anisotropy ratio by a factor of four causes in three-fold increase of the condition number and a 50% jump of the iteration count. The Grid rotation does not affect the performance.

[§] In all of the experiments, the initial residual was reduced by a factor of 10^6.

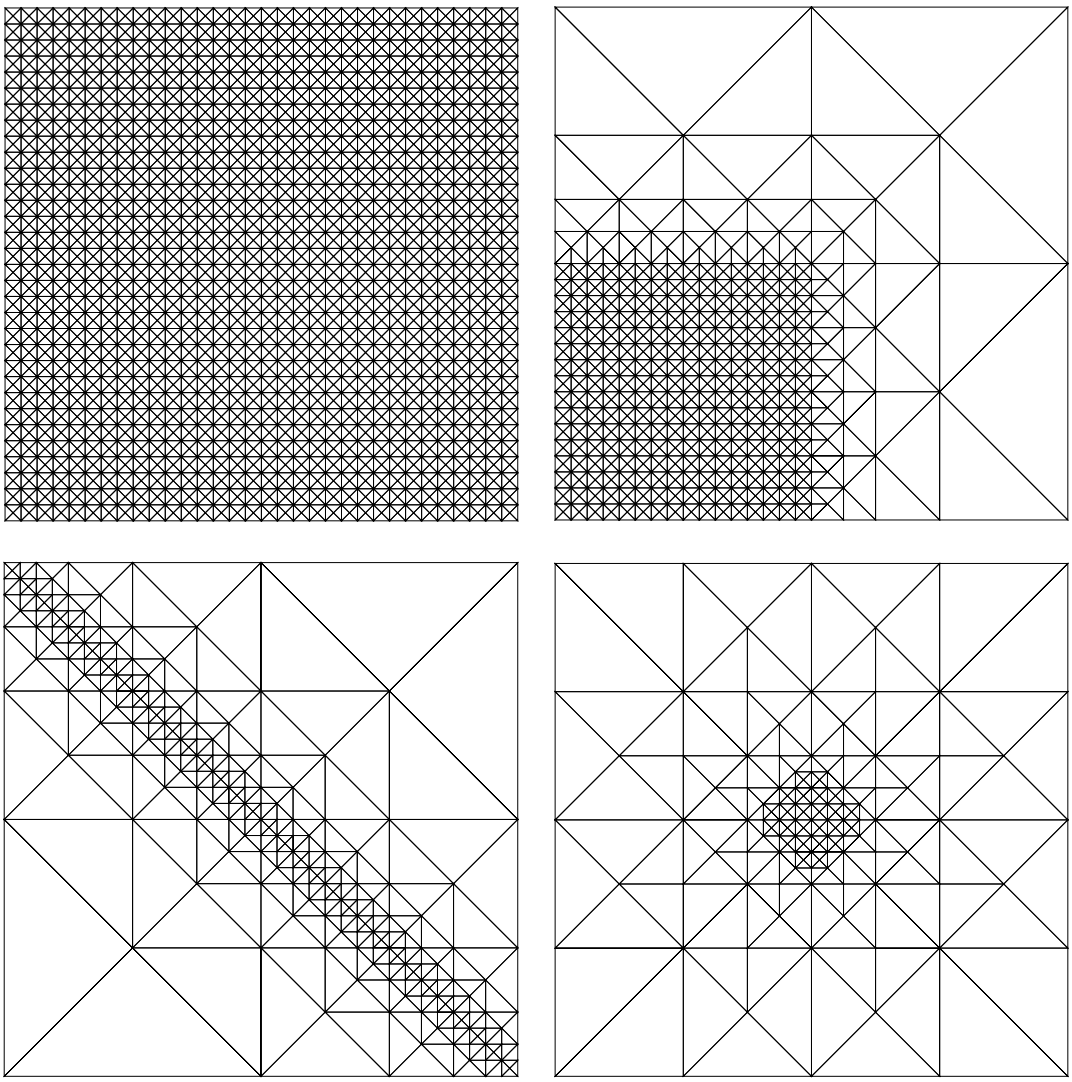


Figure 1. Two-dimensional case: the meshes with different refinements.

Table II. Two-dimensional case: performance of PCG iterations for the scalar jump in diagonal tensor, uniform Grid.

	K								
	13, $n_K = 24\,704$			15, $n_K = 98\,560$			17, $n_K = 393\,728$		
a	*#it*	*cnd*	*t*	*#it*	*cnd*	*t*	*#it*	*cnd*	*t*
1	12	2.9	0.7	14	3.4	3.6	16	3.9	18.3
10^4	31	34.3	1.8	39	44.9	9.9	46	57.0	50.4
10^6	28	36.3	1.6	34	47.5	8.5	41	60.0	45.2

Table III. Two-dimensional case: performance of PCG iterations for the jumps in anisotropic tensor, uniform and rotated Grids.

	K											
	Uniform Grid						Rotated (20°) Grid					
	13, $n_K = 24\,704$			17, $n_K = 393\,728$			13, $n_K = 24\,704$			17, $n_K = 393\,728$		
a	*#it*	*cnd*	*t*	*#it*	*cnd*	*t*	*#it*	*cnd*	*t*	*#it*	*cnd*	*t*
1	12	2.9	0.7	16	3.9	18.3	12	2.9	0.7	16	3.9	18.3
4	17	5.0	1.0	23	7.1	26.3	16	4.3	1.0	22	6.2	25.6
16	28	12.2	1.6	38	18.8	42.7	24	9.5	1.4	33	14.9	37.2
64	42	35.3	2.5	62	57.5	69.0	36	25.5	2.1	51	42.6	57.2

Table IV. Two-dimensional case: thresholding and performance of PCG iterations (uniform Grids).

	K											
	13				15				17			
τ	*#it*	*cnd*	*t*	S_K	*#it*	*cnd*	*t*	S_K	*#it*	*cnd*	*t*	S_K
0	12	2.9	0.7	623 085	14	3.4	3.6	2 601 133	16	3.9	18.3	10 636 013
0.01	12	2.9	0.6	467 013	14	3.5	3.2	1 881 361	17	4.0	17.0	7 538 759
0.05	13	2.9	0.6	297 915	15	3.4	3.0	1 167 929	17	4.0	14.6	4 605 607

Table V. Two-dimensional case: parallel performance of PCG iterations, initialization versus iterations.

		n_p									
		2		4		8			16		
Mesh	n_K	t_0	t	t_0	t	t_0	t	$t_{n_p=4}/2t_{n_p=8}$	t_0	t	$t_{n_p=4}/4t_{n_p=16}$
Uniform	131 581	1.1	3.4	0.6	2.3	0.3	1.2	0.96	0.2	1.0	0.57
	525 309	4.9	16.3	2.9	11.4	1.7	6.8	0.84	1.0	3.8	0.75
Ref. in a sbdm	133 431	1.1	3.8	0.6	2.4	0.4	1.3	0.92	0.2	1.0	0.60
	528 961	4.7	17.1	2.7	11.5	1.6	6.4	0.90	1.0	4.2	0.68
Ref. to a curve	229 197	2.8	6.9	1.7	5.4	1.0	2.8	0.96	0.6	2.1	0.64
	458 561	6.0	14.7	3.5	11.5	2.0	6.1	0.94	1.2	4.3	0.67

In Table IV, the economical modification is considered for the case described at the start of the section. The threshold parameter τ influences the aggregated size of all level matrices S_K. In certain cases, the size may be reduced by a factor of 2.5. However, the iteration time is not reduced essentially.

In Table V we present the parallel features of the two-dimensional algorithm: how the initialization (t_0) and iterative solution (t) time depend on the number of processors n_p. Two conclusions may be drawn from these data. First, independently of the number of processors, the initialization time does not exceed a quarter of the iteration time, and the speed-up of the initialization matches the speed-up of the iterations. Second, the speed-up of iterations is quite satisfactory: the two-fold and four-fold increases of the processor numbers, compared with the performance on 4 processors, yield at least 84 and 67% of the parallel efficiency on fine Grids. It is pertinent to note that the deterioration of the parallel efficiency for 16 processor runs is less severe on uniform Grids. The poor parallel efficiency with respect to the two-processor runs is conditioned by the parallel computer architecture: the MPI communications for dual processors have very small overheads because of shared memory. The actual interprocessor exchanges occur for the case of more than two processors. This is the reason why we present the parallel efficiency with respect to four-processor runs. The two-processor run measurements are shown in order to estimate the overhead of actual interprocessor communications.

We now examine the performance of the thee-dimensional algorithms. We consider problem (1) with $\Gamma_1 = \emptyset$, identity matrix $\mathcal{A}$, $f = 1$. The coarsest mesh $\mathcal{E}_0$ consists of two tetrahedra T_1 and T_2 with vertices $\{(0,0,0),(1,0,0),(0,1,0),(0,0,1)\}$ and $\{(0,0,0),(1,0,0),(0,1,0),(0,0,-1)\}$. The single processor performance of the algorithm on different meshes is shown in Table VI. The types of meshes are shown in Figure 2. The results are similar to those shown in Table I: the condition numbers are small (less than 6.2) and in the worst case (uniform refinements) an eight-fold increase of n_K increments the iteration count by three. We expect that the saturation of *cnd* and #*it* may occur on finer Grids that are beyond the computer memory capacitance. However, the time per iteration is almost proportional to the number of unknowns: an eight-fold increase of the matrix order produces nine or ten-fold slow downs of one iteration. The arithmetical complexity of the evaluation of the preconditioner does not seem to be sensitive to the type of Grid refinement.

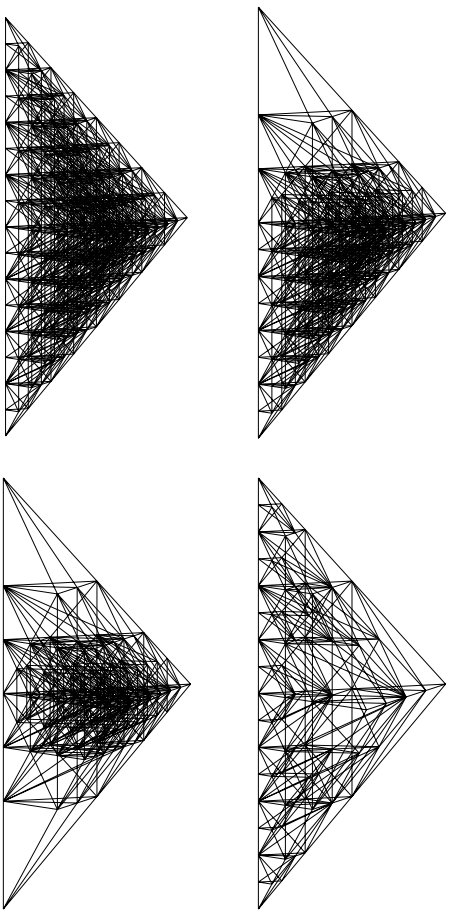


Figure 2. Three-dimensional case: the meshes with different refinements.

The case of jumping scalar coefficients is considered in Table VII: $\mathcal{A}(x, y, z) = a(x, y, z)\mathcal{I}$, where $a(x, y, z) = \mathbf{a}$, if $z > 0$, else $a(x, y, z) = 1$. In Table VIII we allow the anisotropic tensor to jump and change the principal direction

$$\mathcal{A} = \begin{cases} \operatorname{diag}(\mathbf{a}^2, \mathbf{a}, 1) & z > 0 \\ \operatorname{diag}(\mathbf{a}, \mathbf{a}^2, 1) & z < 0 \end{cases}$$

The permeability tensor $\mathcal{A}$ thus has a jump on the plane $\{z = 0\}$. In addition, we repeat the experiment on the same Grid which is then rotated by about 20° in planes xz and xy. The center of the rotation coincides with the baricenter of the Grid.

Table VI. Three-dimensional case: performance of PCG iterations for meshes with different types of local refinement.

	K											
	10				13				16			
Mesh	n_K	#*it*	*cnd*	*t*	n_K	#*it*	*cnd*	*t*	n_K	#*it*	*cnd*	*t*
Uniform	4416	13	3.5	0.2	34 048	15	4.1	2.0	267 264	19	5.6	26.3
Ref. in a sbdm	2739	13	3.4	0.1	19 360	15	4.0	1.2	143 694	18	5.3	13.3
Ref. to a plane	1974	17	5.9	0.1	9106	18	5.9	0.7	39 534	18	5.9	3.9
Ref. to a point	960	17	6.2	<0.01	2770	17	6.2	0.2	7080	18	6.2	0.6

Table VII. Three-dimensional case: performance of PCG iterations for the scalar jump in diagonal tensor, uniform Grid.

	K								
	10, $n_K = 4416$			13, $n_K = 34\,048$			16, $n_K = 267\,264$		
a	#*it*	*cnd*	*t*	#*it*	*cnd*	*t*	#*it*	*cnd*	*t*
1	13	3.5	0.2	15	4.1	2.0	19	5.6	26.3
10^4	20	11.8	0.2	28	20.1	3.8	39	33.8	53.8
10^6	18	12.5	0.2	26	23.3	3.5	39	43.3	53.9

Table VIII. Three-dimensional case: performance of PCG iterations for the jumps in anisotropic tensor, uniform and rotated Grids.

	K											
	Uniform Grid						Rotated (20°) Grid					
	10, $n_K = 4416$			16,$n_K = 267\,264$			10, $n_K = 4416$			16, $n_K = 267\,264$		
a	#*it*	*cnd*	*t*	#*it*	*cnd*	*t*	#*it*	*cnd*	*t*	#*it*	*cnd*	*t*
1	13	3.5	0.2	19	5.6	26.3	13	3.5	0.2	19	5.6	26.3
4	24	11.1	0.3	30	13.7	41.4	32	21.6	0.4	38	21.2	52.5
8	37	33.0	0.4	49	36.0	67.7	48	59.3	0.6	60	58.5	82.5

Table IX. Three-dimensional case: thresholding and performance of PCG iterations.

	K											
	10				13				16			
τ	*#it*	*cnd*	*t*	S_k	*#it*	*cnd*	*t*	S_k	*#it*	*cnd*	*t*	S_k
0	13	3.5	0.2	156 156	15	4.1	2.0	1 864 932	19	5.6	26.3	18 948 525
0.01	13	3.5	0.1	96 081	16	4.5	1.6	904 339	19	6.3	16.8	7 528 827
0.05	14	4.0	0.1	69 868	19	6.2	1.6	600 337	25	10.4	18.6	4 612 142

Similarly to the two-dimensional cases, on a fixed Grid, the iteration count saturates as the value of the jump in the scalar permeability tensor tends to infinity, whereas halving the mesh size increases the iteration count by approximately 30%. The four-fold increase of the anisotropy in the permeability tensor roughly triples the condition number and augments the iteration count by 50%. The Grid rotation (simulating the full permeability tensor) has a greater impact on the performance of the method than in the two-dimensional case.

The economical modification is considered in Table IX. The threshold parameter τ influences the aggregated size of all level matrices S_K. In certain cases, the size may be reduced by a factor of 4.0. As a consequence, the iteration time may drop by 30–50%, depending on the extent of the iteration count increase.

In Table X, we exhibit the parallel features of our three-dimensional algorithm. The table is the complete analog of Table V. The data presented in Table X differ from those in the two-dimensional case. First, on two and four processors, the initialization cost approaches the iteration expenses. With a larger number of processors, the time of initialization is as little as one-half of the iteration time (or less). The parallel efficiency on fine Grids is almost as good as in the two-dimensional case: for four-processor runs, two-fold and four-fold increases of $n_p = 4$ yield at least 87 and 59% of parallel efficiency, respectively. The worse parallel efficiency in the 16 processor runs is explained by a larger weight of global communications due to a larger number of global degrees of freedom. That is why our parallel implementation loses its efficiency on a larger number of processors. The reason for presenting the parallel efficiency with respect to the four-processor run was explained earlier.

6. CONCLUSIONS

The efficient parallel solution of conservative approximations of elliptic equations was discussed. The method is the combination of several contemporary technologies: the two- and three-dimensional marked edge bisection algorithms for the generation of locally refined meshes; the inertial bisection algorithm for efficient and topologically simple data partitioning; the MFEM for locally conservative approximations and the algebraic condensation for its compact storage; the modern multilevel method (with the modification) applied to the algebraically condensed system; the parallelization technologies providing the load balance and the *mathematical equivalence* of the parallel versions.

Table X. Three-dimensional case: parallel performance of PCG iterations, initialization versus iterations.

		K									
		2		4		8			16		
Mesh	n_K	t_0	t	t_0	t	t_0	t	$t_{n_p=4}/2t_{n_p=8}$	t_0	t	$t_{n_p=4}/4t_{n_p=16}$
Uniform	51 707	1.4	1.9	0.6	1.0	0.3	0.7	0.71	0.2	1.0	0.25
	403 451	20.0	22.1	9.3	12.9	4.2	7.3	0.88	1.9	5.5	0.59
Ref. to a plane	95 907	4.1	4.7	1.7	2.8	0.8	1.8	0.78	0.4	1.5	0.47
	394 975	22.6	23.8	10.8	14.6	4.8	7.6	0.96	2.3	5.5	0.66
Ref. to a line	138 347	4.1	5.9	2.1	3.7	1.1	2.3	0.80	0.7	1.5	0.62
	283 443	9.2	12.4	4.8	8.0	2.5	4.6	0.87	1.4	3.1	0.64

The numerical experiments confirm that the implemented algorithm possesses the following properties. It has almost linear arithmetical complexity and data size, is robust to jumps in the permeability tensor, and shows satisfactory speed-ups even on highly non-uniform meshes.

ACKNOWLEDGEMENTS

The authors are very grateful to B. Beckner, S. Lyons and S. Maliassov from URC, ExxonMobil Corp. for inspiration and fruitful discussions. The work has been performed within the framework of the project 'Linear solvers for fluid flow problems in porous media' sponsored by ExxonMobil Corp.

REFERENCES

1. Raviart P, Thomas J. A mixed finite element method for 2nd order elliptic problems. *Mathematical Aspects of Finite Element Methods*, Galligani I, Magenes E (eds.) (*Lecture Notes in Mathematics*, vol. 606). Springer: New York, 1977; 192–315.
2. Amestoy P, Duff I, L'Excellent J-Y. Multifrontal parallel distributed symmetric and unsymmetric solvers. *Computational Methods in Applied Mechanical Engineering* 2000; **184**:501–520.
3. Gupta A. Improved symbolic and numerical factorization algorithms for unsymmetric parse matrices. *SIAM Journal of Matrix Analysis and Applications* 2002; **24**:529–552.
4. Saad Y. *Iterative Methods for Sparse Linear Systems*. PWS: Boston, MA, 1996.
5. Nepomnyaschikh S. Mesh theorems of traces, normalizations of function traces and their inversion. *Soviet Journal of Numerical Analysis and Mathematical Modelling* 1991; **6**:1–25.
6. Dyadechko V, Iliash Yu, Vassilevski Y. Structuring preconditioners for unstructured meshes. *Russian Journal of Numerical Analysis and Mathematical Modelling* 1996; **11**:139–154.
7. Stüben K. A review of algebraic multigrid. *GMD Report 69*, Sankt Augustin, Germany, 1999.
8. Cleary A, Falgout R, Henson V, Jones J, Manteuffel T, McCormick S, Miranda J, Ruge J. Robustness and scalability of algebraic multigrid. *SIAM Journal of Scientific Computing* 2000; **21**:1886–1908.
9. Stüben K. Algebraic multigrid (AMG): Experiences and comparisons. *Applied Mathematical Computing* 1983; **13**:419–452.
10. Kickinger F. Algebraic multigrid for discrete elliptic second-order problems. *Multigrid Methods*, vol. 5. Springer: Berlin, 1998; 157–172.

11. Brezina M, Cleary A, Falgout R, Henson V, Jones J, Manteuffel T, McCormick S, Ruge J. Algebraic multigrid based on element interpolation (AMGe). *SIAM Journal of Scientific Computing* 2000; **22**:1570–1592.
12. Chartier T, Falgout R, Henson V, Jones J, Manteuffel T, McCormick S, Ruge J, Vassilevski P. Spectral AMGe. *SIAM Journal of Scientific Computing* 2003; **25**:1–26.
13. Gupta A, Karypis G, Kumar V. A highly scalable parallel algorithm for sparse matrix factorization. *IEEE Transactions on Parallel and Distributed Systems* 1997; **8**:502–520.
14. Kaporin I, Konshin I. A parallel block overlap preconditioning with inexact submatrix inversion for linear elasticity problems. *Journal of Numerical and Linear Algebra Applications* 2002; **9**:141–162.
15. Henson V, Yang U. BoomerAMG: a parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics* 2002; **41**:155–177.
16. Bänsch E. Local mesh refinement in 2 and 3 dimensions. *IMPACT of Computing in Science and Engineering* 1991; **3**:181–191.
17. Rivara M. Mesh refinement processes based on the generalized bisection of simplexes. *SIAM Journal of Numerical Analysis* 1984; **21**:604–613.
18. Arbogast T, Chen Z. On the implementation of mixed methods as nonconforming methods for second order elliptic problems. *Mathematical Computation* 1995; **64**:943–672.
19. Chen Z. Equivalence between multigrid algorithms for nonconforming and mixed methods for second-order elliptic problems. *East–West Journal of Numerical Mathematics* 1996; **4**:1–33.
20. Chen Z. On the convergence of Galerkin-multigrid methods for nonconforming finite elements. *East–West Journal of Numerical Mathematics* 1999; **7**:79–104.
21. Wheeler M, Yotov I. Multigrid on the interface for mortar mixed finite element methods for elliptic problems. *Computational Methods in Applied Mechanics and Engineering* 2000; **184**:287–302.
22. Arnold D, Brezzi F. Mixed and nonconforming finite element methods: Implementation, postprocessing and error estimates. *RAIRO Modelisation Mathematique et Analyse Numerique* 1985; **19**:7–32.
23. Williams R. Unification of spectral and inertial bisection, 1994. http://www.cacr.caltech.edu/Publications/techpubs/PAPERS/ccsf048.pdf.
24. Williams R. Performance of dynamic load balancing algorithms for unstructured mesh calculations. *Concurrency* 1991; **3**:457.
25. Chugunov V, Vassilevski Yu. Parallel multilevel data structures for a non-conforming finite element problem on unstructured meshes. *Russian Journal of Numerical Analysis and Mathematical Modelling* 2003; **18**(1):1–11.
26. PLTMG Homepage. http://www.scicomp.ucsd.edu/∼reb/.
27. UG Homepage. http://cox.iwr.uni-heidelberg.de/∼ug/.
28. ALBERT Homepage. http://www.alberta-fem.de.
29. Rivara M. Selective refinement/derefinement algorithms for sequences of nested triangulations. *International Journal of Numerical Methods in Engineering* 1989; **28**:2889–2906.
30. Bramble J. *Multigrid Methods* (*Pitman Research Notes in Mathematics*, vol. 294), Longman: London, 1993.
31. http://www.netlib.org/linalg/metis-4.0.tar.gz.

