A. Heinlein, U. Hetmaniuk, A. Klawonn, and O. Rheinbach

A Parallel Study of the Approximate Component Mode Synthesis Special Finite Element Method
A. Heinlein, U. Hetmaniuk, A. Klawonn and O. Rheinbach

A Parallel Study of the Approximate Component Mode Synthesis Special Finite Element Method
A PARALLEL STUDY OF THE APPROXIMATE COMPONENT MODE SYNTHESIS SPECIAL FINITE ELEMENT METHOD

A. HEINLEIN†, U. HETMANIUK‡, A. KLAWONN†, AND O. RHEINBACH§

September 16, 2014

Abstract. A special finite element method based on approximate component mode synthesis (ACMS) was introduced in [U. Hetmaniuk, R. Lehoucq, A special finite element method based on component mode synthesis, ESAIM: Mathematical Modelling and Numerical Analysis, 44 (2010), 401-420]. ACMS was developed for second order elliptic partial differential equations with rough or highly varying coefficients. Here, a parallel implementation of ACMS is presented and parallel scalability issues are discussed for representative examples. Additionally, a parallel domain decomposition preconditioner (FETI-DP) is applied to solve the ACMS finite element system. Weak parallel scalability results for ACMS are presented for up to 1024 cores. Our numerical results also suggest a quadratic-logarithmic condition number bound for the preconditioned FETI-DP method applied to ACMS discretizations.

Key words. multiscale, domain decomposition, finite elements, eigendecomposition, FETI-DP, component mode synthesis

1. Multiscale Problems. We consider problems

\[ -\nabla \cdot (A(x)\nabla u(x)) = f(x) \quad \text{in } \Omega, \]
\[ u = 0 \quad \text{on } \partial \Omega, \]

when the coefficient matrix \( A \) is rough or highly varying on a small scale. Such problems are also often referred to as multiscale problems. Multiscale problems are challenging to solve using standard finite element methods since very fine meshes are needed in order to resolve the features of the solution on the fine scale. The large number of degrees of freedom then leads to high demands with respect to memory and computational resources. One way to overcome these issues is to introduce methods which explicitly take into account the information on the small scale without resorting to a brute force discretization. This can be achieved, e.g., by including the coefficient information into the basis functions. Various approaches have been proposed in this field, including multiscale finite element (MsFEM) [12, 17], mixed multiscale finite element methods [1], heterogeneous multiscale finite element [11], adaptive multiscale [33], the generalized finite element [3, 2, 4], and component mode synthesis (CMS) methods [7, 18].


2.1. Introduction. The special finite element method considered in this paper was introduced by Hetmaniuk and Lehoucq in [15]. Additional theory has been provided in [16]. The method is designed as an approximation of a component mode synthesis method, using three different types of basis functions in order to so. It combines bubble-type eigenmodes, vertex-specific energy minimizing extensions of nodal
trace functions, and coupling edge-based eigenmodes. We will refer to these basis functions as ACMS\textsuperscript{1} shape functions or coarse basis functions. An important additional property of these basis functions is, in contrast to the basis functions used in the CMS method, their local support. The resulting linear system is therefore sparse and the construction of the coarse basis functions parallelizes well. This will be pointed out again later.

Using the ACMS method, in order to achieve a comparable accuracy, the discretized system can be smaller by one to two orders of magnitude compared to a brute force discretization; see also Section 5.2.

The numerical construction of the ACMS shape functions can be computationally expensive but this step is easily parallelized. In this paper, we therefore investigate the computational cost of using an ACMS discretization in a parallel context. In our view, the parallel use of this method makes the most sense.

Although the linear systems arising from ACMS discretizations are much smaller compared to a brute force discretization, these systems can still be large and ill conditioned. We therefore combine the ACMS discretization with a parallel FETI-DP domain decomposition method as iterative solution method. We see that the FETI-DP method applied to ACMS discretizations is numerically scalable and converges in a small number of iterations.

We will thus describe how to compute the ACMS shape functions in parallel and how to solve the resulting system with a FETI-DP method. Finally, we will provide numerical results which show, that our resulting algorithm, containing the parallel computation of the ACMS shape functions and the parallel solution phase, is parallel scalable.

2.2. Description of the Method. The model problem (1.1) can be transformed into the variational formulation

\begin{equation}
(a(u, v) = f(v) \quad \forall v \in H^1_0(\Omega)
\end{equation}

with the bilinear form and the linear functional

\begin{equation}
a(u, v) = \int_\Omega (\nabla u(x))^T A(x) \nabla v(x) dx \quad \text{and} \quad f(v) = \int_\Omega f(x)v(x) dx.
\end{equation}

We assume that the matrix $A$ is uniformly symmetric positive definite, and that it satisfies

\begin{equation}
0 < \alpha_{\text{min}} \xi^T \xi \leq \xi^T A(x) \xi \leq \alpha_{\text{max}} \xi^T \xi \quad \forall x \in \bar{\Omega} \text{ and } \xi \in \mathbb{R}^2 \setminus \{0\}
\end{equation}

with constants $\alpha_{\text{min}}, \alpha_{\text{max}}$ independent of $x$.

In order to define the finite element space of our special finite element method we consider a family $(\tau_h)_h$ of conforming partitions of $\Omega$ into triangles or convex quadrilaterals. The elements of the partition are assumed to be open sets, and the intersection of two distinct elements $K$ and $K'$ is either empty, a vertex, or a complete edge with two vertices. This partition introduces

\begin{equation}
\Gamma = \left( \bigcup_{K \in \tau_h} \partial K \right) \setminus \partial \Omega
\end{equation}

\textsuperscript{1}The letter “A” in ACMS stands for “Approximate” and emphasizes the approximation of a CMS technique.
and thus
\begin{equation}
\Omega = \left( \bigcup_{K \in \tau_h} K \right) \cup \Gamma.
\end{equation}

Additionally, we define discrete harmonic, i.e., energy-minimizing, extensions of trace functions on \( \Gamma \). We denote the space of trace functions on \( \Gamma \) by \( W_\Gamma \), and thus a discrete harmonic extension \( E_\Omega \tau \) of \( \tau \in W_\Gamma \) is characterized either by the minimization problem
\begin{equation}
\inf_{v \in H^1_0(\Omega)} a(v, v) \quad \text{with} \quad v|_\Gamma = \tau.
\end{equation}
or equivalently by
\begin{equation}
-\nabla \cdot (A(x) \nabla E_\Omega \tau) = 0 \quad \text{in} \ K, \ \forall K \in \tau_h
\end{equation}
\begin{align*}
E_\Omega \tau &= \tau \quad \text{on} \ \Gamma, \\
E_\Omega \tau &= 0 \quad \text{on} \ \partial \Omega.
\end{align*}

Based on the partition \((\tau_h)_h\) of \( \Omega \), subspaces
\begin{equation}
V_K = \{ v \in H^1_0(\Omega) : v|_K \in H^1_0(K) \text{ and } v|_{\Omega\setminus\bar{K}} = 0 \} \subset H^1_0(\Omega)
\end{equation}
for all \( K \in (\tau_h)_h \) and
\begin{equation}
V_\Gamma = \{ E_\Omega \tau \in H^1_0(\Omega) : \tau \in W_\Gamma \} \subset H^1_0(\Omega)
\end{equation}
can be introduced, such that
\begin{equation}
H^1_0(\Omega) = \left( \bigoplus_{K \in \tau_h} V_K \right) \oplus V_\Gamma.
\end{equation}
This decomposition is orthogonal with respect to \( a(\cdot, \cdot) \), and is a basic concept also in domain decomposition methods.

On the basis of this decomposition the problem (2.1) can equivalently be written
\begin{equation}
a(u_K, v_K) = f(v_K) \quad \forall K \in \tau_h, \forall v_K \in V_K \\
a(u_\Gamma, v_\Gamma) = f(v_\Gamma) \quad \forall v_\Gamma \in V_\Gamma
\end{equation}
with
\begin{equation}
u = \sum_{K \in \tau_h} u_K + u_\Gamma
\end{equation}
and \( u_K \in V_K \) and \( u_\Gamma \in V_\Gamma \).

It is, of course, important that our special finite element method can be implemented efficiently. Therefore, we will choose basis functions with local support. These shape functions are designed as local approximates to the CMS finite element space [15], cf. [7, 18], which was shown to have good approximations properties but unfortunately includes shape functions with global support.

The approximation properties of the (A)CMS finite element space can be explained in sense of the Kolmogoroff \( n \)-th width because the basis functions in the
finite element space are eigenmodes designed to approximate $u_K$ for all $K \in \tau_h$ and $V_{\Gamma}$ and $u_{\Gamma}$. For the Kolmogoroff $n$-th width, see [26, 32]. Thus there are eigenmodes contained in $V_K$ which are local and there are eigenmodes in $V_{\Gamma}$ which have global support. The computation of the latter can not be parallelized as easily. For a more detailed description see also [15].

In order to define the finite element space of the special finite element method different types of basis functions are used, i.e. vertex-specific, edge-based, and fixed-interface shape functions. We will refer to the latter shape functions also as interior bubble-type functions.

We will first introduce here the vertex-specific basis functions, which are of MsFEM basis function type, cf. [12, 17], briefly and describe local eigenvalue problems which then lead to the two other types of shape functions.

A basis function, which corresponds to a vertex $P$ of the partition $\tau_h$, is given by the following variational problem

$$
\begin{align*}
-\nabla \cdot (A(x)\nabla \varphi_P) &= 0 & \text{in } K, \forall K \in \tau_h \\
\varphi_P &= 0 & \text{on } \partial \Omega, \\
\varphi_P &\neq 0 & \text{on } \Gamma, \\
\varphi_P(P') &= \delta_{P,P'} & \text{on } \partial \Omega,
\end{align*}
$$

where $P'$ is also a vertex of the partition and $\delta_{P,P'}$ is the Kronecker delta function. The MsFEM basis functions are therefore discrete harmonic extensions of trace functions on $\Gamma$, and thus $\varphi_P \in V_{\Gamma}$.

It is left to define the trace values of $\varphi_P$ on $\Gamma$, and there are different possibilities to do so. The easiest way is to define $\varphi_P$ as a linear function on each edge between vertices. The corresponding finite element space is called $V_{\text{ACMS-L}}$ in [15]. This choice of the trace incorporates the oscillations of the coefficient matrix $A$ by means of the discrete harmonic extension but ignores oscillations on the edge.

Instead, the trace can be defined differently, i.e., we require the values on an edge $e \subset \Gamma$ to satisfy

$$
\begin{align*}
\frac{\partial}{\partial \tau} (A(x)\tau, \nabla \varphi_P(x)) &= 0 & \text{on } e \\
\varphi_P(P') &= \delta_{P,P'} & \text{on } \partial \Omega,
\end{align*}
$$

where $\tau$ denotes the tangential vector of the edge with $||\tau|| = 1.$
This leads, together with the eigenmodes defined beneath, to the finite element space \( V_{\text{ACMS}} \) which for simplicity is also just called \( V_{\text{ACMS}} \). When the matrix \( A \) is a constant multiple of the identity, both spaces are identical. A numerical comparison of these two spaces is presented in [15].

The other two types of basis functions are defined by eigenvalue problems. The so-called fixed-interface shape functions are given by: Find \((\tilde{z}_{*,K}, \lambda_{*,K}) \in V_{K} \times \mathbb{R}\) such that

\[
 a(\tilde{z}_{*,K}, \lambda_{*,K}) = \lambda_{*,K} (\tilde{z}_{*,K}, v)_{L^2(\Omega)} \quad \forall v \in V_{K}
\]

on each of the \( K \in \tau_h \). For any open edge \( \epsilon \subset \Gamma \) we define the edge-based coupling basis function by the corresponding eigenvalue problem in the space of harmonic extensions: Find \((\tilde{\tau}_{*,\epsilon}, \lambda_{*,\epsilon}) \in H^1_{00}(\epsilon) \times \mathbb{R}\) such that

\[
 a(E_{\Omega}(\tilde{\tau}_{*,\epsilon}), E_{\Omega}(\tilde{\eta})) = \lambda_{*,\epsilon} (E_{\Omega}(\tilde{\tau}_{*,\epsilon}), E_{\Omega}(\tilde{\eta}))_{L^2(\Omega)} \quad \forall \eta \in H^1_{00}(\epsilon)
\]

with \( \tilde{\eta} \) being the trivial extension of \( \eta \) by 0 on \( \Gamma \setminus \epsilon \). The eigenvalues \( \{\lambda_{i,K}\}_{i=1}^{\infty} \) and \( \{\lambda_{i,\epsilon}\}_{i=1}^{\infty} \) are assumed to be ordered into nondecreasing sequences, and the corresponding eigenmodes, respectively. They form orthonormal bases for the inner \( L^2 \)-product of \( V_{K} \) on the element \( K \), and of \( V_{\Gamma} \) on the edge \( \epsilon \).

The finite element space of the special finite element method is given by

\[
 V_{\text{ACMS}} = \left( \bigoplus_{K \in \tau_h} \text{span} \{ z_{i,K} ; 1 \leq i \leq I_K \} \right) \oplus \left( \bigoplus_{P \in \Omega} \text{span} \{ \varphi_P \} \right) \oplus \left( \bigoplus_{\epsilon \subset \Gamma} \text{span} \{ E_{\Omega}(\tilde{\tau}_{i,\epsilon}) ; 1 \leq i \leq I_{\epsilon} \} \right)
\]

with positive integers \( I_K \) and \( I_{\epsilon} \) corresponding to the number of eigenmodes used as basis functions. Note, that the Dirichlet boundary conditions are naturally built into \( V_{\text{ACMS}} \).

2.3. Error estimate. Recently an a priori error estimate for this special finite element method has been given by Hetmaniuk and Klawonn in [16, Prop. 3.4]; see also [16, Prop. 3.6] for an a posteriori error indicator.
Fig. 3. Support and shape of a fixed-interface basis function on a rectangular partition.

Under the assumption that $A$ is in $C^1$ on $\tilde{\Omega}$ and that the solution $u$ of (2.1) belongs to $H^2_0(\Omega)$, with $s_0 > \frac{3}{2}$, the error between the solution $u$ and the approximate solution $u_{ACMS} \in V_{ACMS}$ satisfies

$$a(u - u_{ACMS}, u - u_{ACMS}) \leq \sum_{K \in \mathcal{T}_h} \frac{\|f\|_{L^2(K)}^2}{\lambda_{K,K}} + C_{s_0,\sigma,A} h^{2s_0-3} \sum_{K \in \mathcal{T}_h} \frac{\|u\|_{H^{s_0}(K)}^2}{\min_{e \subset \partial K \cap \Gamma} \lambda_{e,e}},$$

where the constant $C_{s_0,\sigma,A}$ does not depend on $u$ and $h$. For further details, see [16, Prop. 3.4] and the related proof.

2.4. Parallel Implementation of the ACMS Discretization. Hetmaniuk and Lehoucq [15, Section 5.1] explains how to numerically compute the ACMS basis functions. In this section we will describe briefly the parallel computation of these coarse basis functions on a refined nested mesh, with mesh size $h_f < h$, using Q1 Lagrangian finite elements. Only the fine mesh size $h_f$ is chosen small enough such that the important features of the partial differential equation are resolved. Our special finite element method on the mesh of size $h$ then uses the basis functions constructed above.

For the implementation of the algorithm the library PETSc 3.2-p7 [5, 6] and MPI are used. Particularly, we make use of the matrix, vector and solver structures which are provided therein. The discrete harmonic extensions occurring in the ACMS method are computed using the sparse Cholesky decomposition implemented in PETSc.

The first step in the construction of the ACMS system is the assembly of the local Q1 fine elements. This step is local to a processor core and can be performed in parallel without communication: The local stiffness-matrices and right hand side are built, and it will be sufficient to store them locally. The same is valid for the local mass matrices which have to be assembled in order to be used within the generalized eigenvalue problems for the computation of the eigenmodes; see Section 2.2. We conclude that this part of the implementation needs no communication, and thus expect it to be perfectly scalable.

The next step is the construction of the basis functions for the ACMS elements. Basis functions with local support will be used, i.e., the support only contains a bounded number of coarse elements: The fixed-interface basis functions $z_{i,K}$ have
nonzero values only on one coarse element, cf. Figure 3. The computation of these bubble-type basis functions is independent of other coarse elements and can therefore be performed locally without any communication. Thus, this is scalable as well.

Although the vertex-specific basis functions $\varphi_P$ are nonzero on several coarse elements, see for example Figure 4 for a rectangular coarse mesh, the values can be computed separately and on parallel on each of those elements.

The next step involves nearest neighbor communication if an edge is shared by two different processes. The support of edge-based coupling basis functions $\tau_{i,e}$ consists of two coarse elements. If both adjoint coarse elements reside on the same process the computation does not require communication. Otherwise submatrices corresponding to the generalized eigenvalue problem have to be communicated.

In order to compute the trace $\eta$ of an edge-based basis function on the edge $e$ between the subdomains $\Omega_1$ and $\Omega_2$ the generalized eigenvalue problem

$$
(\begin{pmatrix}
-K_{11}^{-1}K_{1e} & -K_{22}^{-1}K_{2e} \\
-K_{22}^{-1}K_{2e} & I
\end{pmatrix})^T
\begin{pmatrix}
K_{11} & 0 & K_{1e} \\
0 & K_{22} & K_{2e} \\
K_{1e}^T & K_{2e}^T & K_{ee}
\end{pmatrix}
\begin{pmatrix}
K_{11} & 0 & K_{1e} \\
0 & K_{22} & K_{2e} \\
K_{1e}^T & K_{2e}^T & K_{ee}
\end{pmatrix}^T
\begin{pmatrix}
-K_{11}^{-1}K_{1e} & -K_{22}^{-1}K_{2e} \\
-K_{22}^{-1}K_{2e} & I
\end{pmatrix}
\eta
$$

(2.16)
has to be solved. We can derive

\[
\begin{pmatrix}
-K_{11}^{-1}K_{1e} \\
-K_{22}^{-1}K_{2e} \\
I
\end{pmatrix}^T
\begin{pmatrix}
K_{11} & 0 & K_{1e} \\
0 & K_{22} & K_{2e} \\
K_{1e}^T & K_{2e}^T & K_{ee}
\end{pmatrix}
\begin{pmatrix}
-K_{11}^{-1}K_{1e} \\
-K_{22}^{-1}K_{2e} \\
I
\end{pmatrix}
= -K_{1e}^TK_{11}^{-1}K_{1e} - K_{2e}^TK_{22}^{-1}K_{2e} + K_{ee} = S_e^{(1)} + S_e^{(2)}
\]

with \( S_e^{(i)} = -K_{ie}^TK_{ii}^{-1}K_{ie} + K_{ee} \). The matrices \( S_e^{(i)} \) can be computed locally on both coarse elements without any communication. Analogously, we have

\[
\begin{pmatrix}
-K_{11}^{-1}K_{1e} \\
-K_{22}^{-1}K_{2e} \\
I
\end{pmatrix}^T
\begin{pmatrix}
M_{11} & 0 & M_{1e} \\
0 & M_{22} & M_{2e} \\
M_{1e}^T & M_{2e}^T & M_{ee}
\end{pmatrix}
\begin{pmatrix}
-K_{11}^{-1}K_{1e} \\
-K_{22}^{-1}K_{2e} \\
I
\end{pmatrix}
= K_{1e}^TK_{11}^{-1}M_{11}K_{1e} - K_{1e}^TK_{11}^{-1}M_{1e} - M_{1e}^TK_{11}^{-1}K_{1e} + K_{2e}^TK_{22}^{-1}M_{22}K_{2e} - K_{2e}^TK_{22}^{-1}M_{2e} - M_{2e}^TK_{22}^{-1}K_{2e} + M_{ee} = \tilde{S}_e^{(1)} + \tilde{S}_e^{(2)}
\]

with \( \tilde{S}_e^{(i)} = K_{ie}^TK_{ii}^{-1}M_{ie}K_{ie} - K_{ie}^TK_{ii}^{-1}M_{ie} - M_{ie}^TK_{ii}^{-1}K_{ie} + M_{ie}^{(i)} \). Also the matrix \( \tilde{S}_e^{(i)} \) does only need local computations on the coarse element \( e \).

If an edge \( e \) is shared by two different processes the Schur complement matrices \( S_e^{(i)} \) and \( \tilde{S}_e^{(i)} \) are computed independently and in parallel. We then solve the edge eigenvalue problem on the process with the lower rank. Communication is therefore involved when transferring \( S_e^{(i)} \) and \( \tilde{S}_e^{(i)} \) to the process responsible for the eigenvalue problem. This is implemented by standard MPI calls. Subsequently, the computed eigenmodes are communicated back to the processor with the higher rank. In our numerical experiments the time for this communication will be visible, see Section 5.3.

Finally,

\[
\begin{pmatrix}
-K_{ie}^{-1}K_{ie} \\
I
\end{pmatrix}\eta
\]

computes the local portion of the basis function on the coarse element \( i \).

### 2.5. Computation of the Eigenvalue Problems

The edge-based eigenvalue problem (2.16), and the fixed-interface eigenvalue problem, respectively, are generalized eigenvalue problems of relatively small size. The latter has the dimension of the inner fine degrees of freedom of one coarse element, the first even only of the number of degrees of freedom on one edge.

Thus all matrices needed for the computation can be stored as dense matrices, and the generalized eigenvalue problems are computed directly using the LAPACK routine DSYGVX. This routine has a cubic complexity. Since our eigenvalue problems are defined on the edges and the interior of the coarse elements only, this remains affordable. Approximate, iterative eigensolvers could, of course, also be used for larger eigenvalue problems. This may be important in 3D. But already in our current implementation the most expensive step is the computation of the fixed-interface interior bubble functions.

The underlying algorithm of DSYGVX is a Cholesky decomposition in order to reduce the generalized eigenvalue problem to a standard eigenvalue problem. Then the resulting matrix is reduced to tridiagonal form using an orthogonal similarity transformation and a QR-algorithm is employed for the computation of the eigenvectors.
We recall that some MPI communication is necessary to transfer the necessary information to build the eigenvalue problems. We apply a very simple load balancing approach: For each edge, we always gather the information on the process with the lower rank and we also solve the eigenvalue problem on this process.

3. The FETI-DP Method. The FETI-DP domain decomposition method is a divide-and-conquer approach to the iterative solution of linear systems discretized by finite elements. It has been shown to be scalable and robust for a wide field of applications. It has been introduced by Farhat et al. in [14, 13]. FETI-DP [14, 13, 23, 24, 21, 25] and BDDC type [10, 8, 28, 27, 29] methods use coarse spaces constructed from constraints. These are typically implemented using partial assembly of the finite elements. This approach has facilitated the extension of the scalability of these methods, see, e.g., [39, 30, 20, 22, 31, 38, 36]. Among the extensions are inexact FETI-DP methods which were introduced in [20]. Their parallel scalability has been demonstrated in [23, 34] for up to 65,000 processors. Recently, new scalable nonlinear versions of the FETI-DP have been introduced in [19]. For an introduction to domain decomposition methods, see, e.g., [37, 35]. The parallel FETI-DP implementation used in this paper is based on [21, 34] and uses PETSc [5, 6] and UMFPACK [9].

In FETI-DP methods the domain $\Omega$ is decomposed into $N$ non-overlapping subdomains $\{\Omega_i\}_{i=1,...,N}$. The corresponding local stiffness matrices $K^{(i)}$ and right hand sides $f^{(i)}$ are assembled for $i = 1, ..., N$. The system

$$(K u) = (f)$$

has no unique solution, because the local stiffness matrices $K^{(i)}$ are not invertible for subdomains with $\partial \Omega_i \cap \partial \Omega \neq \emptyset$.

To obtain a unique and continuous solution, we partition the interface $\Gamma' = \bigcup_{i=1}^{N} \partial \Omega_i \setminus \partial \Omega$ in dual ($\Delta$) and primal variables ($\Pi$) first. We strongly enforce continuity in the primal variables by global assembly of the corresponding degrees of freedom. Continuity in the dual variables is enforced by the additional constraint $Bu = 0$. Here, $B$ is the standard FETI jump operator [37].

By introducing Lagrange multipliers $\lambda$ we can now write down the FETI-DP master system as a saddle point problem

$$\begin{pmatrix} \tilde{K} & BT \\ B & 0 \end{pmatrix} \begin{pmatrix} \tilde{u} \\ \lambda \end{pmatrix} = \begin{pmatrix} \tilde{f} \\ 0 \end{pmatrix}$$
with

\begin{equation}
\tilde{K} = \begin{pmatrix}
K_{BB}^{(1)} & \tilde{K}_{IB}^{(1)} \\
\vdots & \ddots \\
\tilde{K}_{IB}^{(N)} & \tilde{K}_{II}^{(N)} & K_{II}^{(1)}T
\end{pmatrix}
\end{equation}

and

\begin{equation}
\tilde{f} = \begin{pmatrix}
\tilde{f}^{(1)} \\
\vdots \\
\tilde{f}^{(N)}
\end{pmatrix},
\end{equation}

If a sufficient number of degrees of freedom are chosen as primal variables then the matrix \( \tilde{K} \) is invertible. In this paper, we use primal vertices; see Figure 6.

The system can then be written as

\begin{equation}
F\lambda = d
\end{equation}

with \( F = B\tilde{K}^{-1}B^T \) and \( d = B\tilde{K}^{-1}\tilde{f} \). By eliminating the inner variables first and using the Schur complement, \( F \) can also be written as \( B\Gamma\tilde{S}^{-1}\Gamma B^T \).

The Dirichlet preconditioner \( M_D^{-1} \)

\begin{equation}
M_D^{-1} = B_{D,\Gamma}\tilde{S}\Gamma B_{D,\Gamma}^T
\end{equation}

can then be defined using \( B_{D,\Gamma} \), which is a scaled version of \( B_\Gamma \). In this paper, we use simple multiplicity scaling, i.e., we scale by the inverse of the multiplicity of a node. For the many other possibilities of scaling please refer to the literature.

Typically, condition number bounds of the type

\begin{equation}
\kappa \left( M_D^{-1} \right) \leq C \left( 1 + \log \left( H/h \right) \right)^2,
\end{equation}

can be shown for the preconditioned FETI-DP system where \( h \) is the size of the finite elements and \( H \) is the size of the subdomains. The constant \( C \) is independent of \( h, H \), and coefficient jumps. Such estimates have been shown for finite element discretizations as well as higher order and spectral element discretizations. In all of these cases the bound implies that the number of conjugate gradient iterations is bounded independently of the number of subdomains and thus is independent of the problem size.

### 3.1. FETI-DP methods for ACMS discretizations.

Since all our data is distributed, i.e., our mesh is distributed and the ACMS shape functions are constructed in parallel, we may also apply a parallel solver building on the parallel distributed data. We use a parallel FETI-DP domain decomposition method where the subdomains are defined from the distribution of the ACMS elements.

A step essential to the fast convergence of the FETI-DP method is the selection of appropriate primal degrees of freedom. To get a better idea of how these are chosen, we identify the basis functions with nodes lying in the corresponding element, see Figure 7. One FETI-DP subdomain in general contains several ACMS elements, and following Figure 7 we see that the fixed-interface basis functions always correspond to inner degrees of freedom. In our FETI-DP method, for simplicity, we will chose only the vertices to be primal, and thus only the vertex-based basis functions may correspond to primal nodes. The vertex-based basis functions which are not primal correspond together with the edge-based basis functions to the dual degrees of freedom. For standard finite element spaces in 2D a vertex coarse space is sufficient to obtain numerical scalability in the sense of a \( \left( 1 + \log(H/h) \right)^2 \) condition number bound.
4. Model Problems. For the numerical experiments we employ the three example problems which have already been studied in [15]. We refer to [15] for comparisons with other special finite element methods. All the problems considered here have homogeneous Dirichlet boundary conditions.

4.1. Laplace equation (Problem 1). The first model problem is the Laplace equation

\begin{align}
-\Delta u &= f \quad \text{in } \Omega, \\
u &= 0 \quad \text{on } \partial \Omega,
\end{align}

with \( f(x, y) = 2x(1-x) + 2y(1-y) \). This corresponds to a coefficient matrix \( A(x, y) = 1 \). The exact solution then is

\begin{align}
u(x, y) &= x(1-x)y(1-y).
\end{align}

4.2. Equation with a varying coefficient (Problem 2). The second model problem

\begin{align}
-\nabla \cdot \left( \frac{1}{1.2 + \cos(32\pi x(1-x)y(1-y))} \nabla u(x,y) \right) &= f(x) \quad \text{in } \Omega, \\
u &= 0 \quad \text{on } \partial \Omega,
\end{align}

with \( f(x, y) = 64\pi (x(1-x) + 2y(1-y)) \) is equipped with a varying coefficient-matrix

\begin{align}
A(x, y) &= \left( \frac{1}{1.2 + \cos(32\pi x(1-x)y(1-y))} \right) I,
\end{align}

where \( I \) denotes the identity matrix; see also Figure 8. The exact solution then is

\begin{align}
u(x, y) &= (1.2 \cdot 32\pi x(1-x)y(1-y) + \sin(32\pi x(1-x)y(1-y))).
\end{align}

4.3. Equation with a highly-oscillating coefficient (Problem 3). Here, we consider a model problem with a highly-oscillating coefficient, cf. Figure 8,

\begin{align}
-\nabla \cdot (A(x, y) \nabla u(x, y)) &= -1 \quad \text{in } \Omega, \\
u &= 0 \quad \text{on } \partial \Omega,
\end{align}

with \( A(x, y) = \left( \frac{2+1.8\sin(25\pi x)}{2+1.8\cos(25\pi y)} + \frac{2+\sin(25\pi y)}{2+1.8\sin(25\pi x)} \right) I \), where \( I \) denotes the identity matrix.
5. Numerical Results. The linear systems arising from ACMS discretizations can be ill conditioned as seen in Table 1. Here, for a fixed $h/h_f = 30$ the number of ACMS elements is increased for Problem 2, see Section 4.2. The estimated condition number seems to grow according to $(1/h)^2$, see also Figure 10, and approaches $1.8 \times 10^4$ for $1/h = 512$. The use of an efficient preconditioner for the solution of the ACMS system is thus advisable. In this paper, we will apply the FETI-DP domain decomposition method. Many other parallel preconditioners are, of course, also possible.

5.1. Numerical scalability of the FETI-DP method for ACMS discretizations. In Table 2 and Figure 11 the condition number of the preconditioned FETI-DP system for the ACMS method applied to Problem 2 is presented. Figure 11
Parallel Adaptive Component Mode Synthesis

Table 1

<table>
<thead>
<tr>
<th>1/h</th>
<th>Cond.</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>55.69</td>
</tr>
<tr>
<td>64</td>
<td>269.22</td>
</tr>
<tr>
<td>128</td>
<td>1098.15</td>
</tr>
<tr>
<td>256</td>
<td>4407.93</td>
</tr>
<tr>
<td>512</td>
<td>17686.94</td>
</tr>
</tbody>
</table>

Estimated condition number for different ACMS discretizations for h/h_f = 30 (Problem 2). The condition number was estimated from the Lanczos process.

Fig. 10. Estimated condition number for different ACMS discretizations for h/h_f = 30; see Table 1. The dashed line represents the slope of 1/h^2 growth.

These numerical results strongly suggest a (1 + log(H/h))^2 bound for this problem with highly varying coefficient inside ACMS elements and thus inside subdomains. No theory is currently known for the setting presented here. The numerical results are therefore very encouraging.

Additionally, in Tables 5, 6, 7, and 9, we see that the condition number of the preconditioner FETI-DP operator (“FETI-DP”/“Cond.”) and thus also the number of FETI-DP conjugate gradient iterations (“FETI-DP”/“It.”) stay bounded for a growing number of subdomains (1/H)^2 if H/h and h/h_f are kept constant; see Section 5.3.

We can thus conclude that we have numerical scalability for the FETI-DP method for our ACMS problems.

5.2. Comparison of an ACMS discretization with a brute force approach. In Table 4 and Figure 12 a brute force Q1 discretization (“Q1”) is compared with different ACMS discretizations (“ACMS”) using the same number of (coarse) degrees of freedom (dof). We vary h/h_f = 5, 10, 20, 30, i.e., we consider a different number of (internal) fine degrees of freedom. For our comparison we observe the convergence of the energies E_{ACMS} and E_{Q1} to the energy E^* of the (known) exact solution.

In Figure 12 “ACMS” refers to the number of coarse degrees of freedom. For
Estimated condition number of the preconditioned FETI-DP system for the ACMS discretization for $1/H = 16$ and $h/h_f = 20$ (Problem 2).

<table>
<thead>
<tr>
<th>$H/h$</th>
<th>Cond.</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3.52</td>
</tr>
<tr>
<td>8</td>
<td>4.60</td>
</tr>
<tr>
<td>12</td>
<td>5.31</td>
</tr>
<tr>
<td>16</td>
<td>5.81</td>
</tr>
<tr>
<td>20</td>
<td>6.20</td>
</tr>
<tr>
<td>24</td>
<td>6.52</td>
</tr>
<tr>
<td>28</td>
<td>6.80</td>
</tr>
</tbody>
</table>

Table 2

Fig. 11. Estimated condition number for different $H/h$ for ACMS discretizations with $h/h_f = 20$ (Problem 2) and a fit of a second order polynomial in $\log(H/h)$ to the data; see Table 2 for the data.

comparison, for the ACMS method, we also consider the total sum of (internal) fine degrees of freedom, denoted by “ACMS (fine dof)”, i.e., the number of ACMS elements (with mesh size $h$) times $(h/h_f)^2$. In Figure 12 we see that the expected convergence order is achieved for all methods. We observe that for the same number of (coarse) degrees of freedom the accuracy of the ACMS method is always significantly higher than in the brute force method.

Let us briefly discuss the setting of Table 4 and Figure 12. For the ACMS approach we consider the coarse and also the (internal) fine degrees of freedom. For example, for $h/h_f = 30$ the sum of all degrees of freedom on the ACMS fine meshes is larger than the number of ACMS coarse degrees of freedom by approximately a factor of $30^2/4 = 225$, i.e., slightly more that two magnitudes. For $h/h_f = 20$ it is larger only by a factor of approximately $20^2/4 = 100$; for the exact numbers refer to Table 3. The fine dof can, of course, be viewed as degrees of freedom only during the construction of the ACMS shape functions. Later, they are dependent variables.

From Table 4, for $h/h_f = 30$ we can see that the distance of the energy of the (brute force) Q1 discretization to the exact energy falls below that of the distance of the ACMS discretization to the exact energy only if between 60 and 80 times more degrees of freedom are invested. Of course a large number of fine (internal) degrees of
Parallel Adaptive Component Mode Synthesis

Table 3

<table>
<thead>
<tr>
<th>ACMS dof</th>
<th>ACMS fine dof</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(h/h_f = 5)</td>
</tr>
<tr>
<td></td>
<td>49</td>
</tr>
<tr>
<td></td>
<td>225</td>
</tr>
<tr>
<td></td>
<td>961</td>
</tr>
<tr>
<td></td>
<td>3969</td>
</tr>
<tr>
<td></td>
<td>16129</td>
</tr>
<tr>
<td></td>
<td>65025</td>
</tr>
<tr>
<td></td>
<td>261121</td>
</tr>
<tr>
<td></td>
<td>1046529</td>
</tr>
</tbody>
</table>

ACMS degrees of freedom (dof) and the corresponding sum of fine degrees of freedom for \(h/h_f = 5, 10, 20, 30\). The fine dof are only internal degrees of freedom and are used to define the ACMS shape functions.

Table 4

<table>
<thead>
<tr>
<th>dofs</th>
<th>(h/h_f = 5)</th>
<th>(h/h_f = 10)</th>
<th>(h/h_f = 20)</th>
<th>(h/h_f = 30)</th>
</tr>
</thead>
<tbody>
<tr>
<td>49</td>
<td>4.51e+00</td>
<td>1.56e+00</td>
<td>8.26e-01</td>
<td>6.89e-01</td>
</tr>
<tr>
<td>225</td>
<td>1.12e+00</td>
<td>3.61e-01</td>
<td>1.72e-01</td>
<td>1.37e-01</td>
</tr>
<tr>
<td>961</td>
<td>2.77e-01</td>
<td>8.69e-02</td>
<td>3.96e-02</td>
<td>3.08e-02</td>
</tr>
<tr>
<td>3969</td>
<td>6.90e-02</td>
<td>2.14e-02</td>
<td>9.61e-03</td>
<td>7.42e-03</td>
</tr>
<tr>
<td>16129</td>
<td>1.72e-02</td>
<td>5.34e-03</td>
<td>2.38e-03</td>
<td>1.83e-03</td>
</tr>
<tr>
<td>65025</td>
<td>4.30e-03</td>
<td>1.33e-03</td>
<td>5.94e-04</td>
<td>4.57e-04</td>
</tr>
<tr>
<td>261121</td>
<td>1.08e-03</td>
<td>3.33e-04</td>
<td>1.48e-04</td>
<td>1.14e-04</td>
</tr>
<tr>
<td>1046529</td>
<td>2.69e-04</td>
<td>8.33e-05</td>
<td>3.71e-05</td>
<td>1.61e-03</td>
</tr>
</tbody>
</table>

Comparison of the energies for the ACMS special finite element discretization and a Q1 discretization (Problem 2). Here, \(E^*\) is the energy of the (known) exact solution, \(E_{ACMS}\) the energy of the ACMS solution and \(E_{Q1}\) using the (brute force) Q1 solution.

freedom are needed to achieve this. The situation is similar for \(h/h_f = 20\) although in Figure 12 one can clearly see that the horizontal distance of the black (ACMS) and the blue (Q1) curve is smaller for \(h/h_f = 20\) than for \(h/h_f = 30\). For \(h/h_f = 5, 10\) the approximation quality is reduced further but we still do not see a break down. This may be due to the structure of Problem 2. For problems with a clearly defined micro scale we would expect a sudden drop in the quality of the approximation as soon as the ACMS fine mesh fails to resolve the micro scale properly.

In Figure 12 we also see that the ACMS discretization profits from increasing the number of internal fine degrees of freedom. Of course, in our current implementation, the computational work to compute the ACMS shape functions increases superlinearly for growing \(h/h_f\). In the following sections we therefore choose \(h/h_f = 20\) as a compromise between accuracy and computational cost.

5.3. Weak Parallel Scalability. In this section, we present weak parallel scaling results for the parallel ACMS approach as discretization and using a parallel FETI-DP method as an iterative solver. The results for \(h/h_f = 20\) are shown in Tables 5, 6, and 7. We recall that \(h\) is the size of an ACMS element and \(h_f\) the size
of a fine Q1 finite element; see Figure 9. In these experiments, since \( h/h_f = 20 \) is fixed, the number of fine degrees of freedom for each ACMS element is kept constant. Moreover, since \( H/h = 28 \) is fixed, the number of ACMS elements for each FETI-DP subdomain is also constant. Since \( 1/H \) is growing, the number of subdomains increases, as well as the number of MPI ranks and processor cores, from \( 2^2 = 4 \) to \( 32^2 = 1024 \).

The number of FETI-DP subdomains \((1/H)^2 = 4, 16, 64, 256, 1024\) is always identical to the number of processor cores, i.e., we use up to 1024 cores of a Cray XT6. We also always have one FETI-DP subdomain for each process or processor core. The Cray XT6 has 24 cores per node (AMD Magny Cours 1.9 GHz).

Since the ACMS systems are small compared to the total number of fine degrees of freedom the time spent to solve the system by our parallel FETI-DP method is on the order of only 1 second or less; cf. Tables 5, 6, and 7.

Let us briefly describe the columns of Tables 5, 6, and 7. In “ACMS”/“Fine Q1” we measure the time for the assembly of the Q1 elements on the fine ACMS mesh of size \( h_f \). The fine mesh is needed to compute the ACMS shape functions. This phase should scales perfectly, and it does so.

The column “ACMS”/“Shape” presents the time for the construction of the
ACMS shape functions by computing harmonic extensions and computing generalized eigenvalue problems using the local fine Q1 meshes on each ACMS element. Because of MPI communication that is necessary in the construction of the ACMS edge shape functions, see Section 2.4, this phase does not scale perfectly. But a good scalability is still achieved.

The column “ACMS”/“Ass.” presents the time for the assembly of the ACMS system using the ACMS shape functions. Since the system is small the time spent here is not significant.

The column “FETI-DP”/“Time” denotes the time for the solution of the ACMS system by the FETI-DP method, “FETI-DP”/“It.” denotes the number of conjugate gradients iterations and “FETI-DP”/“Cond.” denotes the estimated condition number obtained from the Lanczos process. Since, again, the system is small compared to the number of processor cores the time spent here is also not significant.

“Total Time” denotes the complete time to solution and “Speedup” and “Efficiency” denote the corresponding parallel speedup and efficiency where $1/H = 2$, i.e., 4 processor cores, are the baseline. For perfect weak parallel scalability the “Total Time” should stay constant, resulting in a perfect speedup of 256 (compared to the baseline of 4 cores) and a perfect parallel efficiency of 100%. We achieve a parallel efficiency of 79%, 83%, and 84% for 1024 cores in Tables 5, 6, and 7.

Comparing the column “Fine Q1” in Tables 5, 6, and 7, we see that the time for the assembly of the fine Q1 problem is much larger for Problem 2 (21s) and again larger for Problem 3 (36s to 38s). This is due to the expensive evaluations of the trigonometric functions in the coefficient functions of Problem 2 and 3. We have one evaluation of a cosine function for each Gauß point for Problem 2, see Section 4.2, and four evaluations of trigonometric functions for Problem 3, see Section 4.3.

The parallel scalability is also illustrated in Figure 13.

5.3.1. Discussion of the FETI-DP solution phase. The FETI-DP method is known to scale well even on large supercomputers and for very large problems. In our case the ACMS system is indeed very small compared to the number of processor cores.
Let us briefly discuss details. For the FETI-DP method, we see numerical scalability, i.e., the number of conjugate gradient iterations stays bounded for increasing $1/H$. The parallel scalability of the FETI-DP solution phase by itself is far from perfect but this is mainly a result of the very short absolute solution times ($0.09s−1.35s$). The largest ACMS problem solved by FETI-DP in Tables 5, 6, and 7 is only 804 609 degrees of freedom in size. This is a very small problem for a FETI-DP method running on 1024 processor cores, it results in fewer than 800 degrees of freedom for each core. The number ACMS fine degrees of freedom is of course much larger, i.e., 321 162 241; see Table 8. But these fine dof have been eliminated already in the earlier phase. From a more detailed analysis, we have found that the increase of the FETI-DP time almost completely stems from an increasing time spent in the FETI-DP conjugate gradient iteration. This is a result of the growth in conjugate gradient iterations until $1/H = 8$, i.e., 64 FETI-DP subdomains, i.e., the asymptotic bound is approached only for $\geq 256$ subdomains. This is typical for FETI-DP methods in 2D. Moreover, the conjugate gradient iteration for the FETI-DP system includes global MPI communication as well as MPI collective operations that may indeed add up to a noticeable amount of a fraction of a second on 1024 cores of a Cray XT6. For larger linear systems this is usually an insignificant portion of the solution time. We thus expect to achieve good parallel scalability for much larger problems and numbers of cores than presented here.

5.3.2. Outlook. In Table 9 we have also included weak parallel scalability for Problem 3 and $h/h_f = 30$. Since the eigenvalue problems are now larger the dense linear algebra and the computation of the eigenvalue problems by dense QR becomes increasingly inefficient. Indeed, a quick analysis of detailed timers shows that a large amount of the computing time is spent in the computation of the eigenvalue problem for the interior bubble function. These results indicate that for three dimensional problems an approximate solution of the eigenvalue problems will have to be used.

<table>
<thead>
<tr>
<th>$1/H$</th>
<th>ACMS Time</th>
<th>FETI-DP</th>
<th>Total</th>
<th>Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fine Q1 / Shape / Ass.</td>
<td>Time / It. / Cond.</td>
<td>Time</td>
<td>Speedup / Efficiency</td>
</tr>
<tr>
<td>2</td>
<td>2.91s / 75.65s / 0.97s</td>
<td>0.09s / 4 / 2.43</td>
<td>79.33s</td>
<td>1 / 100 %</td>
</tr>
<tr>
<td>4</td>
<td>2.94s / 77.06s / 1.05s</td>
<td>0.11s / 6 / 6.11</td>
<td>80.81s</td>
<td>3.93 / 98.17 %</td>
</tr>
<tr>
<td>8</td>
<td>2.93s / 77.56s / 1.71s</td>
<td>0.33s / 6 / 6.66</td>
<td>81.53s</td>
<td>15.57 / 97.30 %</td>
</tr>
<tr>
<td>16</td>
<td>2.98s / 78.88s / 2.17s</td>
<td>0.67s / 14 / 6.78</td>
<td>83.24s</td>
<td>61.37 / 95.90 %</td>
</tr>
<tr>
<td>32</td>
<td>2.97s / 95.51s / 4.06s</td>
<td>1.10s / 13 / 6.80</td>
<td>100.31s</td>
<td>202.46 / 79.08 %</td>
</tr>
</tbody>
</table>

Table 5

Weak scaling for $H/h = 28$ and $h/h_f = 20$ (Problem 1). The number of MPI ranks is $(1/H)^2$.

6. Conclusion. We have presented a parallel implementation of the ACMS special finite element method which shows good parallel scalability. Using this implementation, we could perform a comparison with a large brute force discretization using standard bilinear finite elements. These computations allowed us to compare the accuracy of the ACMS method with standard bilinear finite elements for different settings. It was also possible to study the influence of the approximation quality of the eigensystems (fine mesh) on the approximation quality of the ACMS method. We also applied the FETI-DP domain decomposition preconditioner to the ACMS linear system. Our numerical results show that FETI-DP is numerically scalable in this case,
Parallel Adaptive Component Mode Synthesis

<table>
<thead>
<tr>
<th>1/H</th>
<th>ACMS Time</th>
<th>FETI-DP</th>
<th>Total</th>
<th>Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fine Q1 / Shape / Ass.</td>
<td>Time / It. / Cond.</td>
<td>Time</td>
<td>Speedup / Efficiency</td>
</tr>
<tr>
<td>2</td>
<td>21.14s / 75.30s / 0.95s</td>
<td>0.09s / 5 / 2.15</td>
<td>97.20s</td>
<td>1 / 100 %</td>
</tr>
<tr>
<td>4</td>
<td>21.18s / 76.58s / 1.04s</td>
<td>0.11s / 7 / 5.97</td>
<td>98.55s</td>
<td>3.95 / 98.63 %</td>
</tr>
<tr>
<td>8</td>
<td>21.21s / 77.13s / 1.73s</td>
<td>0.34s / 14 / 6.68</td>
<td>99.35s</td>
<td>15.65 / 97.84 %</td>
</tr>
<tr>
<td>16</td>
<td>21.33s / 78.23s / 2.83s</td>
<td>0.71s / 15 / 6.79</td>
<td>100.95s</td>
<td>61.62 / 96.29 %</td>
</tr>
<tr>
<td>32</td>
<td>21.37s / 94.29s / 4.22s</td>
<td>1.16s / 14 / 6.81</td>
<td>117.53s</td>
<td>211.72 / 82.70 %</td>
</tr>
</tbody>
</table>

Table 6
Weak scaling for $H/h = 28$ and $h/h_f = 20$ (Problem 2). The number of MPI ranks is $(1/H)^2$.

<table>
<thead>
<tr>
<th>1/H</th>
<th>ACMS Time</th>
<th>FETI-DP</th>
<th>Total</th>
<th>Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fine Q1 / Shape / Ass.</td>
<td>Time / It. / Cond.</td>
<td>Time</td>
<td>Speedup / Efficiency</td>
</tr>
<tr>
<td>2</td>
<td>35.94s / 77.72s / 0.97s</td>
<td>0.10s / 6 / 2.69</td>
<td>114.43s</td>
<td>1 / 100 %</td>
</tr>
<tr>
<td>4</td>
<td>35.94s / 78.99s / 1.09s</td>
<td>0.13s / 11 / 8.41</td>
<td>115.74s</td>
<td>3.95 / 98.87 %</td>
</tr>
<tr>
<td>8</td>
<td>36.06s / 79.47s / 1.92s</td>
<td>0.40s / 18 / 8.64</td>
<td>116.62s</td>
<td>15.70 / 98.12 %</td>
</tr>
<tr>
<td>16</td>
<td>36.49s / 81.27s / 3.06s</td>
<td>0.78s / 19 / 8.69</td>
<td>119.24s</td>
<td>61.42 / 95.97 %</td>
</tr>
<tr>
<td>32</td>
<td>37.53s / 96.52s / 4.79s</td>
<td>1.35s / 18 / 8.24</td>
<td>136.12s</td>
<td>215.21 / 84.07 %</td>
</tr>
</tbody>
</table>

Table 7
Weak scaling for $H/h = 28$ and $h/h_f = 20$ (Problem 3). The number of MPI ranks is $(1/H)^2$.

i.e., we could see that the computed condition number of the preconditioned system grows quadratic-logarithmically dependent on the size of the subdomain problems. This is the condition number estimate that is usually obtained for FETI-DP applied to standard finite elements and could motivate further theoretical investigations to analytically prove such a condition number estimate. In our present study in two dimensions, direct dense eigensolvers were used. For future ACMS discretizations in three dimensions, iterative sparse eigensolvers are probably necessary.

Acknowledgement. The authors acknowledge the use of the Cray XT6 computer at Universität Duisburg-Essen.

REFERENCES


ACMS degrees of freedom and ACMS internal fine degrees of freedom for $H/h = 28$ and $h/h_f = 20$, i.e. corresponding to the data in Tables 5, 6, 7.

Table 8

<table>
<thead>
<tr>
<th>1/H</th>
<th>ACMS dof</th>
<th>ACMS fine dof</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3 249</td>
<td>1 256 641</td>
</tr>
<tr>
<td>4</td>
<td>12 769</td>
<td>5 022 081</td>
</tr>
<tr>
<td>8</td>
<td>50 625</td>
<td>20 079 361</td>
</tr>
<tr>
<td>16</td>
<td>201 601</td>
<td>80 299 521</td>
</tr>
<tr>
<td>32</td>
<td>804 609</td>
<td>321 162 241</td>
</tr>
</tbody>
</table>

Table 9

<table>
<thead>
<tr>
<th>1/H</th>
<th>Fine Q</th>
<th>Shape</th>
<th>Ass.</th>
<th>Time</th>
<th>It.</th>
<th>Cond.</th>
<th>Total</th>
<th>Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>26.27s</td>
<td>/</td>
<td>193.79s</td>
<td>/</td>
<td>0.57s</td>
<td>6</td>
<td>/</td>
<td>2.41</td>
</tr>
<tr>
<td>4</td>
<td>26.35s</td>
<td>/</td>
<td>204.12s</td>
<td>/</td>
<td>0.61s</td>
<td>10</td>
<td>/</td>
<td>7.18</td>
</tr>
<tr>
<td>8</td>
<td>26.54s</td>
<td>/</td>
<td>206.32s</td>
<td>/</td>
<td>1.30s</td>
<td>16</td>
<td>/</td>
<td>7.38</td>
</tr>
<tr>
<td>16</td>
<td>26.43s</td>
<td>/</td>
<td>206.80s</td>
<td>/</td>
<td>2.35s</td>
<td>17</td>
<td>/</td>
<td>7.48</td>
</tr>
<tr>
<td>32</td>
<td>26.65s</td>
<td>/</td>
<td>210.85s</td>
<td>/</td>
<td>3.73s</td>
<td>17</td>
<td>/</td>
<td>7.16</td>
</tr>
</tbody>
</table>

Weak scaling for $h/h_f = 30$ and $H/h = 16$ (Problem 3).


