Iterative Methods for Substructured Elascticity Problems in Structural Analysis

PETTER E. BJØRSTAD* AND ANDERS HVIDSTEN*

Abstract. This study of iterative methods applied to problems in structural analysis, has been motivated by new results and methods for solving elliptic problems on regions partitioned into substructures. An implementation of a recent algorithm in the framework of a large commercial finite element analysis code is described. Results from several different problems in structural analysis are given. They include the use of membrane, shell, beam and solid elements.

Introduction. Structural analysis provided the problems, motivation and pioneering work that led to the development of the powerful finite element method and this field is still the most important application area of finite element based analysis. There exist a number of highly successful commercial codes that can be used to analyze the behavior of almost any kind of structure under rather general conditions. In almost all of these codes the technique called substructuring [1] is used in order to simplify modeling and post processing. The physical domain is divided into several disjoint pieces called substructures, and each substructure can be assembled separately. The global stiffness matrix is then usually formed in order to solve the equations. One commercial code Sesam [2], pioneered the further use of this concept, by also taking advantage of the substructures (in particular identical ones) in the solution algorithm. In this code the solution procedure is carried out by the elimination of interior unknowns from each substructure, followed by the calculation of Schur complements corresponding to the unknowns on interior interfaces between substructures.

The renewed interest from the numerical analysis community in substructuring as a way of breaking up the solution of elliptic problems into problems on smaller domains has provided a new understanding and a theoretical foundation for the use of iterative methods in this process. An early numerical analysis paper in this direction is [3]. More recent work is described in [4], [5], [6], [7], [8], [9] see also references found in these papers.

In our paper the use of preconditioned conjugate gradients to solve the interface equations arising from structural analysis, is investigated. Issues of actual efficiency of

* Institutt for Informatikk, University of Bergen, Allegt. 55, N-5000 Bergen, NORWAY. Supported by NTNF contract IT0228.20484

this approach are discussed, and the numerical results are compared with results obtained with state of the art software. The results from a comprehensive number of different problems are reported. The paper briefly describes the issues involved in actual software implementation of these methods into a large structural analysis package. This report demonstrates the feasibility of using iterative substructure methods in structural analysis codes. An important aspect of this paper is the generation of suitable test examples and a nontrivial set of computational results. We plan a continued study of additional test examples and algorithmic variants for these problems.

This investigation reveals an important difference between the model problems treated in earlier papers, having only one degree of freedom in each node, and the typical problems in structural analysis with at least two and up to six unknowns per node. This difference causes problems for the algorithm that work very well in the case of only one variable in every node. We report on preliminary experiments in order to address this difficulty. More extensive experiments will be reported in a forthcoming paper.

Substructures in Structural Analysis Codes. The partition of structural problems into subproblems or substructures is a technique that has been used since the finite element method itself gained popularity in the early 1960s [10]. In most finite element codes the modeling of each substructure and the assembly of the corresponding finite elements can be performed independently. These tasks can then be assigned to different engineering groups, computer systems or processors with coordination required only at the interior boundaries in order for the triangulation to match.

In order to understand the underlying data structure in more detail, consider the case of a structure divided into two substructures 1 and 2 with an interior interface 3 between them. The corresponding stiffness matrices are K_{11}, K_{22} and K_{33}, and the coupling between the substructures and the interface K_{13} and K_{23}. Note that K_{33} is assembled from two parts coming from elements belonging to the two different substructures.

$$K_{33} = K_{33}^{(1)} + K_{33}^{(2)}$$

The global stiffness matrix is then assembled from these submatrices.

$$K = \begin{pmatrix} K_{11} & 0 & K_{13} \\ 0 & K_{22} & K_{23} \\ K_{13}^T & K_{23}^T & K_{33} \end{pmatrix}$$

The solution of the global problem is most often carried out in one big step using a direct, secondary storage form of Cholesky decomposition.

One commercial code however, carries the substructuring idea one step further [11], [12], by computing the Cholesky factorization of the individual substructures independently. This code became operational in 1969 and was called Sesam'69. The code used in this paper is a redesigned successor of Sesam'69, it became operational in the early 1980's.

Elimination of the interior degrees of freedom belonging to each substructure, corresponds to block Gaussian elimination computing the Schur complement of K with respect to the interior unknowns represented by the two diagonal blocks K_{11} and K_{22}. This corresponds to a certain update of the coefficient matrix for the unknowns on the interior boundary between the substructures:

$$K_{33}^{(1)} = K_{33}^{(1)} - K_{13}^T K_{11}^{-1} K_{13}$$

$$K_{33}^{(2)} = K_{33}^{(2)} - K_{23}^T K_{22}^{-1} K_{23}$$

$$K_{33} = K_{33}^{(1)} + K_{33}^{(2)}.$$

In this way one may take advantage of identical substructures. The approach is also quite flexible when only some particular parts of the solution (corresponding to specific substructures), are wanted. Perhaps most important, is the ability to solve very large problems with limited computer resources, in a structured way. This was indeed the original motivation in the algorithm design.

In the Sesam package, all the primitive submatrices introduced above, keep their individual data structures throughout the analysis. It is therefore fairly convenient to access and manipulate the individual pieces. In the next section it is shown that an iterative method for solving the interface equations, without computing the Schur complement K_{33} above, can be implemented within this framework.

Iterative solution of substructured elliptic problems. In this section, we will give a detailed description of the actual implementation of an algorithm based on [7] for problems in structural analysis. This demonstrates that it is feasible to incorporate such an algorithm into a large, real world, commercial structural analysis package. A detailed understanding of the data structures described in the previous section is crucial for the actual implementation work.

We refer to [7] for a detailed analysis of this algorithm for elliptic problems in one unknown. Here, we implement the same algorithm for problems from structural analysis. Following the notation in this reference, we permute K into the form

$$K = \begin{pmatrix} K_{11} & K_{13} & 0 \\ K_{13}^T & K_{33} & K_{23}^T \\ 0 & K_{23} & K_{22} \end{pmatrix}$$

We will solve the problem

$$Ky = b$$

by using the preconditioner corresponding to the matrix

$$K_0 = \begin{pmatrix} K_{11} & K_{13} & 0 \\ K_{13}^T & K_{33}^{(1)} & 0 \\ 0 & K_{23} & K_{22} \end{pmatrix}$$

It should be noticed that K_0 is lower block triangular and that the upper 2 by 2 block corresponds to solving a problem on substructure 1 having the interface variables included as unknowns. This corresponds to a free boundary on the substructure. On the other hand, all unknowns of the second problem are specified on the same interface. We assume that these two problems can be solved. In the current implementation the Cholesky factorizations of the two matrices are computed. The upper 2 by 2 block problem need not be formed explicitly, only pointer information to already existing data structures are manipulated.

Set $y = u - v$ where $K_0 u = b$, v must satisfy

$$Kv = (K - K_0)u \equiv \tilde{b} = \begin{pmatrix} 0 \\ \hat{b} \\ 0 \end{pmatrix}.$$

Write this as

$$KK_0^{-1}K_0v = \tilde{b}.$$

Set $x = K_0v$ and

$$KK_0^{-1}x = \tilde{b}.$$

As is shown in [7], this problem can be solved by a conjugate gradient algorithm. An implementation of this algorithm in the present context, is given below. Finally one must solve $K_0v = x$ and compute $y = u - v$.

A conjugate gradient algorithm. Let Ex extend the vector x defined on the interface, to the entire region and let Ry be the restriction of a global vector y to the interface variables.

Given an initial guess x for the interface unknowns, the conjugate gradient iteration proceeds as follows:

$$\begin{aligned} r &\leftarrow \hat{b} - R(K - K_0)K_0^{-1}Ex - x \\ p &\leftarrow r \\ w &\leftarrow K_0^{-1}Er \\ q &\leftarrow Rw \\ t &\leftarrow R(K - K_0)w \\ a_2 &\leftarrow r^Tq \end{aligned}$$

For k=1,2,.. until convergence do

$$\begin{aligned} s &\leftarrow p + t \\ \alpha &\leftarrow a_2/s^Tq \\ a_1 &\leftarrow a_2 \\ x &\leftarrow x + \alpha p \\ r &\leftarrow r - \alpha s \\ w &\leftarrow K_0^{-1}Er \\ s &\leftarrow Rw \\ a_2 &\leftarrow r^Ts \\ \beta &\leftarrow a_2/a_1 \\ q &\leftarrow s + \beta q \\ p &\leftarrow r + \beta p \\ s &\leftarrow R(K - K_0)w \\ t &\leftarrow s + \beta t \end{aligned}$$

This implementation requires one solution on both substructures in every iteration when solving the linear system K_0, but the multiply by K has been changed to a multiply by the sparse $(K - K_0)$ instead. The iteration requires one long temporary vector w, the 6 other vectors have length equal to the number of unknowns on the interface. The vectors r and $\hat{b}$ can share storage. It should be mentioned that in a completely secondary storage based implementation like this, the vector storage is insignificant. The vector arithmetic in the algorithm itself is also quite modest. The operation count is completely dominated by the operations involving K_0 and $(K - K_0)$. In our work the initial guess has been taken to be $x = 0$. This eliminates the arithmetic in the first line of the algorithm. The convergence criteria used in all experiments reported in this paper, has been

$$\left(\frac{1}{N}\sum_{i=1}^{N} r_i^2\right)^{1/2} \leq \epsilon$$

with $\epsilon = 10^{-5}$. This has been enough to get the printed displacements from the code identical to the results from a traditional run. To develop a practical convergence test would require a more detailed study and a more sophisticated criteria and could save work by stopping early. The required reduction of the residual vector r in the algorithm, may be different for different problems.

Problems with two substructures. A number of different problems have been tried using different finite elements in 2 and 3 dimensions. The following tables have two typical layouts. The table columns represent different meshes on the same geometry or comparisons between the iterative method and the standard direct solution algorithm (Sestra), [13], [14]. The substructures are numbered 1, 2 and 3 consistent with the previous sections. Note that the number of unknowns in region 1 also include the interface unknowns. All entries in the tables are operation counts in Mflops (millions of floating point operations). It should be noted that these numbers are counted during the execution, operations on zeros are not included if the sparse code take advantage of them. The figures are most accurate for large problems since the counting is restricted to certain computational kernel subroutines. *Factor* refers to the Cholesky factorization, *Schur* to the explicit computation of the Schur complement after the factorization has been carried out. *Total* is an estimate for the arithmetic work in the solution algorithm, when solving a problem with one right hand side.

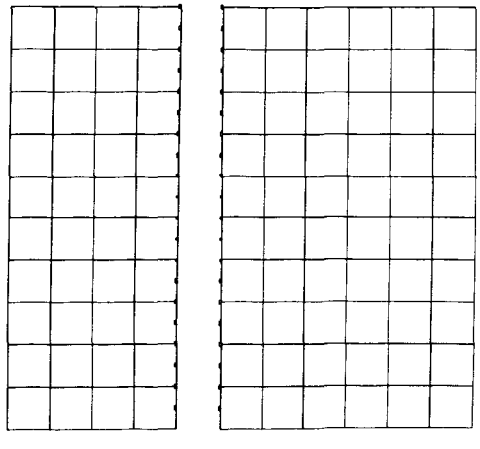

Figure 1: Plate consisting of two substructures.

The eigenvalues of the iteration operator are estimated during the iteration process, and the maximum eigenvalue is listed. Since all eigenvalues are larger than one, this also gives us an estimate for the condition number of the preconditioned system. Finally, the number of conjugate gradient iterations and the work per iteration are listed.

The numbers show that the factorization of the two substructures often dominate. The factorization of substructure 2 should in principle, be equally expensive in both methods. The standard solver uses a bandwidth minimization algorithm at the node level in order to minimize fill. The solver applied to substructure 1 in the iterative algorithm, must factor a matrix that is built from 3 different submatrices. This matrix cannot be easily reordered at the node level. The matrix is never explicitly formed, the factorization algorithm will access it by fetching 42 by 42 sub-blocks directly from disk storage. When used in the iterative algorithm, the factorization algorithm was directed to reorder, using a minimum degree algorithm applied at the block level, in order to at least minimize block

fill. This is likely to degrade performance somewhat on smaller banded problems and it makes the factorization of substructure 2 different in the two cases.

Two-dimensional membrane problems. We first present a rectangular membrane problem cut symmetrically into two squares. The membrane element has 4 nodes, each having two unknown displacement components. The membrane is fixed along the entire edge. The conjugate gradient iteration will converge in one step if the two Schur complements $K_{33}^{(1)}$ and $K_{33}^{(2)}$ are equal. The two matrices are not equal due to the different orientation of the two displacement unknowns with respect to a common global coordinate system. A local change of the coordinates in one of the substructures does not affect this situation. In this special case

$$K_{33}^{(1)} = PK_{33}^{(2)}P$$

where P is a diagonal matrix having alternating $+1$ and -1 on the diagonal. A permutation of the Schur complement is therefore block diagonal since $K_{33} = K_{33}^{(1)} + K_{33}^{(2)}$. Notice that this means that the x and y displacements have been decoupled. The convergence of the preconditioned system now depends on the values of a Rayleigh quotient that takes the form

$$1 + \frac{x^T P K_{33}^{(1)} P x}{x^T K_{33}^{(1)} x}.$$

If no specific information is known about $K_{33}^{(1)}$, then this preconditioning is no better than just using the conjugate gradient algorithm on the problem K_{33}. In order to use a good preconditioner also in this case, we have tried using

$$K_{33}^{(1)} + PK_{33}^{(1)}P.$$

This preconditioning will cause the symmetric problem to converge in one step. It is implemented by subtracting the coupling between x and y displacements off from $K_{33}^{(1)}$ and add the same quantity to $K_{33}^{(2)}$. Using this method, we observe one step convergence both for symmetric regions and for the problem dividing a square membrane into 4 smaller squares. The method also behaves very well for unsymmetric cuts, the spectrum of the iteration operator and its dependence on aspect ratios are very similar to the results reported in [7].

An efficient implementation of this modification has not yet been attempted. The modified algorithm depends on a global change of coordinates. The results above require the interface between the regions to line up with the global coordinate system. If this is not the case, a rotation can be applied in the conjugate algorithm. The effect of this on larger unsymmetric problems has not been investigated.

This discussion shows some of the problems that surface when trying to apply a method that work well for a problem having one unknown per node, to a system of equations leading to several coordinate dependent unknowns per node. Similar problems must be expected in the other cases reported here. This may explain some of the very large eigenvalues observed for plates and solids in the last sections of the paper.

All the computational results reported in the following sections, are based on using the original algorithm without modifying the preconditioning $K_{33}^{(1)}$ in any way.

Shell problems. We first give results for an 8 node shell element. Figure 1 shows a plate fixed along one side only, with the internal interface being normal to that side. The element has nodes at all corners and midpoints of its four sides. Each node has 5 degrees of freedom, three displacements and two rotations (the third rotation being fixed for coplanar elements). The fine grid has 900 unknowns in 1, 600 in 2 and 100 in 3. The coarse grid

has 225, 150 and 50 unknowns in the 3 domains. The largest eigenvalue of the iteration operator is 10., the problem is slightly better conditioned than the corresponding coarse grid problem (with largest eigenvalue 11.). Table 1 shows that the two methods are quite close in performance. The extra work forming the Schur complements compete against the cost of the conjugate gradient iterations.

	Coarse grid		Fine grid	
	Cg	Sestra	Cg	Sestra
Factor 1	1.8	0.5	16.0	7.6
Factor 2	0.5	0.2	3.7	2.5
# cg-iterations	19	-	24	-
Mflop/iteration	0.2	-	0.9	-
Schur 1	-	1.6	-	17.8
Schur 2	-	0.7	-	8.4
Total	5.7	3.1	43.1	37.5

Table 1: Plate problem.

Three-dimensional solid problems. This problem consist of two boxes, one on top of the other as shown in figure 2. The box is fixed at the base.

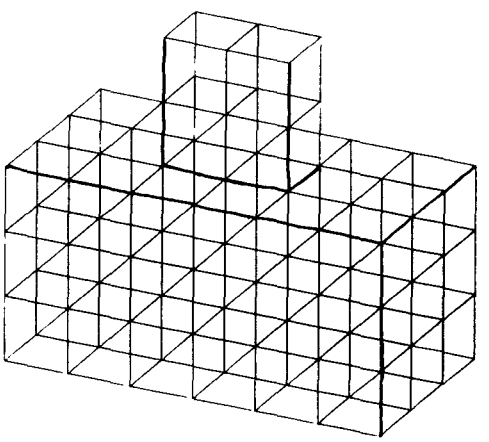

Figure 2: Solid box.

The elements used are 8 node solids with 3 displacement unknowns in each node. The coarse problem has 18 unknowns on the interface, 234 unknowns in the lower box (region 1) and 18 unknowns in region 2. The finer grid is a uniform refinement resulting in 1593, 135 and 45 unknowns in region 1, 2 and 3. The eigenvalues of the iteration operator ranges from 1.0 to 1.93 for the coarse problem while the range is from 1.0 to 2.58 for the fine problem.

We next turn to the influence of the geometry (aspect ratio) of the two regions. In principle, there is always freedom to choose which substructure should be 1 and then call the other 2. In practice, the current implementation will require at least some external Dirichlet boundary condition on the region 1. This is to ensure that the solution procedure for the substructures do not encounter singularities. We use the same solid model, but with a fixed top as well as bottom. The largest eigenvalue of the iteration operator increases

	Coarse grid		Fine grid	
	Cg	Sestra	Cg	Sestra
Factor 1	1.3	0.5	43.9	33.2
Factor 2	0.02	0.02	0.9	0.6
# cg-iterations	5	-	7	-
Mflop/iteration	0.06	-	0.7	-
Schur 1	-	0.4	-	15.7
Schur 2	-	0.04	-	1.5
Total	1.7	1.1	50.7	52.0

Table 2: 3-dimensional solid problem.

to around 35 when the small, top box is numbered 1. This nearly doubles the number of conjugate gradient iterations. In addition, each iteration becomes more expensive. On the other hand, a substantial saving in the initial factorization cost, is obtained. It is interesting to observe that the total cost is almost the same for the two alternatives.

	Bottom is 1		Bottom is 2	
	Small	Large	Small	Large
Unknowns in 1	252	1638	36	180
Unknowns in 2	18	135	234	1593
Unknowns in 3	18	45	18	45
Factor 1	1.31	43.9	0.02	0.9
Factor 2	0.00	0.6	0.9	33.2
# cg-iterations	5	6	11	13
Mflop/iteration	0.06	0.7	0.07	1.1
Largest eigenvalue	1.98	2.7	34.9	36.5
Total	1.7	49.6	1.9	50.5

Table 3: Solid dependence on geometry.

Three-dimensional frame problems. The structure in figure 3, represents a crude model of an offshore oil drilling platform (a jacket). The jacket consists of beam

	Jacket	
	Cg	Sestra
Factor 1	8.2	5.0
Factor 2	1.7	1.2
# cg-iterations	22	-
Mflop/iteration	0.4	-
Schur 1	-	1.1
Schur 2	-	0.6
Total	19.1	7.9

Table 4: Jacket problem.

elements having 6 unknowns at each node. The lower part has 900 unknowns, the upper part 600 unknowns, while the interface contains 100 unknowns. The jacket is fixed at its base only. All eigenvalues except one, of the iteration operator are in the interval 1.0 to 10., the largest one is 167.

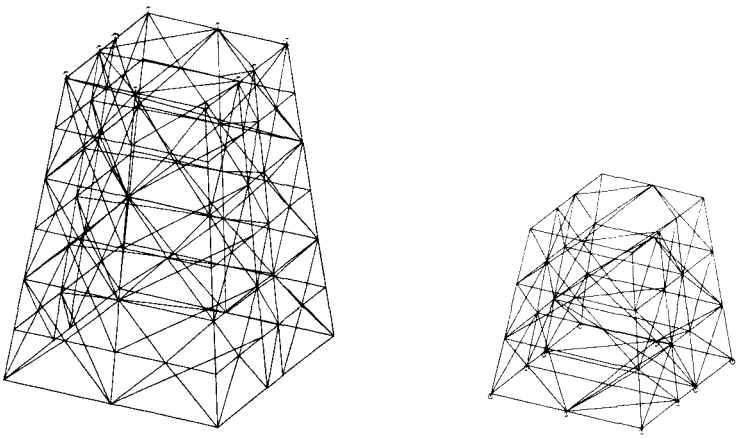

Figure 3: Jacket model.

A comparison with direct methods. The results from the previous section show that the new method sometimes are quite close to the direct solution strategy, but also examples where the direct method appears to be superior. The actual factorization algorithms used on the individual substructures and in particular, the reordering strategy clearly have a critical influence on the performance. A test confirming that the current factorization algorithm is far from optimal in our algorithm reported here, is discussed towards the end of the paper. It should be possible to improve the implementation such that the factorization of problem 1 takes at most the same time as the sum of factorization and Schur complement for the standard method. If this is the case, then the conjugate gradient iterations will compete with the computation of the Schur complement with respect to region 2 plus the cost of the final factorization of the interface problem. There are also clear indications that larger problems with relatively many interface unknowns tend to favor the new method.

One should keep in mind that the standard code has been through a long history of improvements, while the current iterative method is implemented first to demonstrate feasibility and robustness. There may be cases where the initial factorization of the individual substructures can be ignored. This is to a large extent true when building a library of factored substructures in order to perform reanalysis of a given (large) structure during its operational life.

Problems with many substructures. This section will report on experiments dividing a membrane into 4 pieces, a plate in 4 and 16 pieces, and a solid box into 8 substructures. In each case, the number of unknowns within one substructure is varied. The experiment is carried out using only two substructures. We divide the region in a red-black fashion. In the experiments that we report here, the red domain is called 1 and the black 2. Figure 4 shows the red half of the domain using both plates and solids. This will make problem 2 block diagonal. The substructure 1 will be almost block diagonal, since the coupling between the individual pieces will be cross points in two dimensional problems and lines in three dimensional problems. This means that the matrix can be written as a block diagonal plus a correction of low rank. Thus, it can be separated into independent problems by standard linear algebra techniques, see [15].

There are several alternative strategies, the most appealing is perhaps to solve for the unknowns that couple as part of a much coarser problem. This approach ties

	PM4B	PM4C	PS4A	PS4B	PS4C	PS4D
Unknowns in 1	62	254	35	155	625	2555
Unknowns in 2	36	196	10	90	490	2250
Unknowns in 3	26	58	25	65	135	305
Factor 1	0.04	0.6	0.010	0.24	5.1	66.8
Factor 2	0.01	0.2	0.000	0.05	2.3	30.9
# cg-iterations	7	17	10	19	39	50
Mflop/iteration	0.02	0.1	0.005	0.06	0.6	2.6
Max eigenvalue	2.23	52.1	6.95	32.48	30347	1500
Total	0.18	3.2	0.341	3.07	34.1	249.8

Table 5: Membrane and Plate divided in 4 pieces.

the substructure methods to the methods based on multigrid. The approach taken here assumes that problem 1 can be further simplified, the convergence rates and the spectral information are of course independent of the specific solution algorithm. An algorithm that allows more than two substructures will be implemented in the near future.

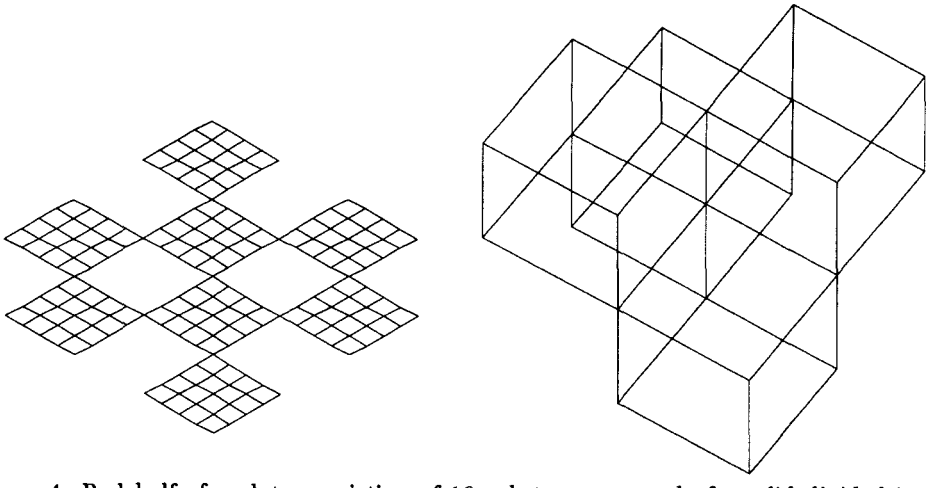

Figure 4: Red half of a plate consisting of 16 substructures and of a solid divided into 8 pieces.

The results for membrane and 4 node shells are displayed in table 5. The numbers are not too encouraging, in particular, two cases had not converged after 50 iterations. The residual in these cases had been reduced from 403 to 0.2 for PS4D and from 288. to 1.6E-4 for PS16C. The eigenvalue estimates are also quite erratic, PS4C being the most surprising. It should be noted that PM4B and PM4C both converge in one step (having all eigenvalues equal to 2) if the modifications discussed earlier had been implemented. The results clearly show that the standard method needs changes in order to be attractive on this class of problems.

Finally, we give results for a three dimensional problem consisting of 8 boxes. The numbers looks much better than for the plates, the second largest eigenvalue in S8B is substantially smaller than the largest eigenvalue in S8A. The increase in iterations is also quite modest.

In order to show how critical the factorization algorithm influences the comparisons, substructure 1 in problem S8B (including the interface unknowns) was also factored using

the standard method. The required work turned out to be 120.0 Mflops to be compared with 172.3 in table 6. This shows that there is substantial room for improvements in the new algorithm.

	S8A	S8B	PS16B	PS16C
Unknowns in 1	651	2280	205	765
Unknowns in 2	270	1425	40	360
Unknowns in 3	381	855	165	405
Factor 1	17.4	172.3	0.55	5.1
Factor 2	0.4	26.7	0.00	0.5
# cg-iterations	43	46	30	50
Mflop/iteration	1.0	5.3	0.15	0.8
Max eigenvalue	154.9	237.5	55.31	386.2
Total	63.2	451.6	6.31	51.1

Table 6: Solid box divided in 8 pieces (S8A and S8B) and plate divided in 16 pieces (PS16B and PS16C).

Conclusions and future work. This work demonstrates a fully working iterative substructure code implemented within the framework of a large industrial structural analysis code. The computational results are of a preliminary nature, substantial improvements can be expected in the time to come. Improvements in the direct solution strategy for the individual substructures or even the use of iterative methods also for the substructures, will decrease the cost of each iteration compared to the numbers reported here. Our work has shown that there can be a dramatic difference between the one variable per node problem and a problem with several variables defined in each node. Significant changes in the algorithms may be necessary in order to overcome these differences.

The main motivation for providing detailed, actual operation counts is to get a picture of what must be obtained in order for this new class of methods to be significantly better than the old ones. The results are quite promising for the case of separate subdomains. The general case with cross points, is more unclear from our tests, in particular for membranes and shells.

It is hoped that this investigation will contribute to focus on an important class of problems, so that more insights will be gained and better algorithms will be developed.

Acknowledgment. The authors extend thanks to professor Olof Widlund for many helpful discussions. We also thank Veritec Sesam Systems for providing the Sesam code for these experiments.

References

[1] K. Bell, B. Hatlestad, O.E. Hansteen, and P.O. Araldsen. *NORSAM, a programming system for the finite element method. users manual, part 1, general description.* NTH, Trondheim, 1973.

[2] *SESAM'80 Description.* Veritec Sesam Systems, August 1986.

[3] P. Concus, G.H. Golub, and D. O'Leary. *A Generalized Conjugate Gradient Method for the Numerical Solution of Elliptic Partial Differential Equations.* Academic Press, 1976.

[4] M. Dryja. A finite element-capacitance method for elliptic problems partitioned into substructures. *Numer. Math. 44, pp.153-168.*, 1984.

[5] Petter E. Bjørstad and Olof Widlund. *Solving elliptic problems on regions partitioned into substructures.* Academic Press, 1984.

[6] J.H. Bramble, J.E. Pasciak, and A.H. Schatz. Preconditioners for interface problems on mesh domains. *Math. Comp.*, 1987.

[7] Petter E. Bjørstad and Olof Widlund. Iterative methods for the solution of elliptic problems on regions partitioned into substructures. *SIAM Journal of Numerical Analysis.*, 1986.

[8] J.H. Bramble, J.E. Pasciak, and A.M. Schatz. An iterative method for elliptic problems on regions partitioned into substructures. *Math. Comp. 46, pp.361-370.*, 1986.

[9] J.H. Bramble, J.E. Pasciak, and A.M. Schatz. The construction of preconditioners for elliptic problems by substructuring. *Math. Comp. July.*, 1986.

[10] J.S. Przemieniecki. Matrix structural analysis of substructures. *Am. Inst. Aero. Astro. J. vol. 1, pp. 138-147.*, 1963.

[11] P.G. Bergan and E. Ålstedt. *A programming system for Finite Element Problems (in Norwegian).* Technical Report, The Norwegian Institute of Technology, 1968.

[12] H.F. Klem. *SESAM'69, General Description.* Technical Report, Det norske Veritas, July 1978.

[13] Petter E. Bjørstad. A large scale, sparse, secondary storage, direct linear equation solver for structural analysis and its implementation on vector and parallel arcchitectures. *Proceedings from an International Conference on Vector and Parallel Computing,June 1986 Loen Norway. To appear in Journal on Parallel Computing.*, 1987.

[14] Petter E. Bjørstad and Jon Brækhus. Implementation and performance of the large scale finite element code sesam on a wide range of scientific computers. *Supercomputer Applications*, 1987.

[15] M. Dryja, W. Proskurowski, and Olof Widlund. *A Method of Domain Decomposition with Cross Points for Elliptic Finite Element Problems.* 1986.