European Congress on Computational Methods in Applied Sciences and Engineering
ECCOMAS 2000
Barcelona, 11-14 September 2000

ON SOME NEW DEVELOPMENTS IN APPROXIMATE INVERSE PRECONDITIONING

Michele Benzi[†], Reijo Kouhia[⋆], and Miroslav Tůma[‡]

[†]Scientific Computing Group CIC-3, Los Alamos National Laboratory, Los Alamos NM 87545, USA e-mail: benzi@lanl.gov, Web page: http://www.c3.lanl.gov/

[⋆]Laboratory of Structural Mechanics, Helsinki University of Technology, PO Box 2100 02150 HUT, Finland e-mail: reijo.kouhia@hut.fi, Web page: http://www.hut.fi/

[‡]Institute of Computer Science, Czech Academy of Sciences, 182 07 Prague 8 Czech Republic e-mail: tuma@cs.cas.cz, Web page: http://www.uivt.cas.cz/

Key words: factorized sparse approximate inverses, preconditioning, conjugate gradient, block algorithms, finite elements

Abstract. *Some new developments of conjugation-based sparse approximate inverse preconditioners are considered in application to problems of solid and structural mechanics. A block version of the AINV algorithm and its stabilized modification are presented. The stabilized AINV algorithm, which in exact arithmetic is breakdown-free for any SPD matrix, is found to result in robust preconditioners even in difficult thin shell applications.*

1 INTRODUCTION

The solution of sparse linear systems, $\boldsymbol{Ax} = \boldsymbol{b}$, where the coefficient matrix is symmetric and positive definite (SPD), by the preconditioned conjugate gradient method is considered. In the last few years there has been considerable interest in explicit preconditioning techniques based on directly approximating the inverse of the coefficient matrix with a sparse matrix. Sparse approximate inverses have been shown to result in good rates of convergence of the preconditioned iteration, comparable to those obtained with incomplete factorization methods, while being well-suited for implementation on vector and parallel architectures. Because the application of the preconditioner reduces to matrix–vector products, parallelization is straightforward.

Although the main motivation for the development of sparse approximate inverse techniques comes from parallel processing, an additional important aspect is their robustness and reliability. It is hardly necessary to emphasize that reliability is paramount in industrial applications. The main reason why direct solvers are still favored over iterative ones in industrial environments is the lack of reliability of the latter. Yet, direct solvers are just not viable for solving very large linear systems arising from the discretization of three-dimensional problems, in which case iterative methods must be used. Besides requiring far less storage, iterative methods can also outperform direct ones in speed. Unfortunately, iterative solvers can fail on difficult problems. In certain application areas, failure of the iterative solver is not infrequent, e.g., in the analysis of thin shells in structural mechanics.

In this paper, some new developments of conjugation-based sparse approximate inverse preconditioners (AINV) are considered in application to problems of solid and structural mechanics. These new developments include the stabilization approach and blocking. The basic AINV approach may not be well defined for an arbitrary SPD matrix, resulting in breakdowns of the conjugation process. Stabilized AINV (SAINV for short) is a modification of the basic approach which, in exact arithmetic, is well defined for any SPD matrix [4, 13]. It is mathematically equivalent to the standard AINV algorithm when no dropping is applied. Even though the pointwise stabilized AINV is breakdown-free, the PCG iterations may fail due to slow convergence. It is shown that blocking can improve the convergence behaviour. Also, the effects of different ordering strategies are studied.

2 THE AINV ALGORITHM

The AINV algorithm [7, 8] builds a factorized sparse approximate inverse of the form

$$\boldsymbol{M} = \boldsymbol{Z}\boldsymbol{D}^{-1}\boldsymbol{Z}^T \approx \boldsymbol{A}^{-1} \tag{1}$$

where $\boldsymbol{Z}$ is a unit upper triangular matrix and $\boldsymbol{D}$ is diagonal. The approximate inverse factor $\boldsymbol{Z}$ is a sparse approximation of the inverse of the $\boldsymbol{L}^T$ factor in the $\boldsymbol{LDL}^T$ decomposition of $\boldsymbol{A}$. When the AINV process is performed exactly, the diagonal matrix $\boldsymbol{D}$ is the same in the two decompositions, and contains the *pivots* down the main diagonal.

The AINV algorithm computes $\boldsymbol{Z}$ and $\boldsymbol{D}$ directly from $\boldsymbol{A}$ by means of an incomplete conjugation ($\boldsymbol{A}$-orthogonalization) process applied to the unit basis vectors. In this process, small elements are dropped to preserve sparsity. The underlying assumption is that most entries in $\boldsymbol{L}^{-1}$ are small in magnitude. This is true for many problems of practical interest, particularly for discretizations of partial differential equations of elliptic type. Sparsity can also be achieved by combining dropping of small entries with suitable sparse matrix orderings, such as Nested Dissection and Minimum Degree. These orderings are beneficial in that they result in smaller time and space requirements for forming and storing the preconditioner, while at the same time improving the quality of the preconditioner in a significant number of cases; see [9, 10, 11].

3 BLOCK ALGORITHMS

In the finite element discretization of solids and shells in particular, the corresponding matrix has a block form. It will be shown that the block approach has a considerable stabilizing effect on the preconditioner construction in difficult thin shell applications. The block form also enhances the efficiency of the code through the use of BLAS3 arithmetic.

In the finite element context where there are several unknowns in a node, the construction of the block structure is trival. However, the block structure, or approximate block structure, can be constructed based on the graph compression procedure [3]. The aim of the graph compression is to find cliques in the undirected graph of the matrix. Consider the graph $G = (V, E)$ of a symmetric matrix. $V = \{1, \ldots, |V|\}$ is the set of vertices that correspond to rows and columns of the matrix. $E \subseteq V \times V$ is the set of its edges where $\langle i, j\rangle \in E$ iff the entry a_{ij} is nonzero. Then the adjacency set of a vertex v is

$$adj(v) = \{u | \langle v, u\rangle \in E\}.$$

Vertices $u \in V$ and $v \in V$ are adjacent iff $v \in adj(u)$. This is the same condition as $u \in adj(v)$. A clique in G is a set of vertices which are all mutually adjacent.

The task to find dense blocks as large as possible in a matrix $\boldsymbol{A}$ is equivalent to the task of finding all cliques which are maximal with respect to inclusion in the graph G of the matrix $\boldsymbol{A}$. For further description of such an algorithm, see [3].

4 BLOCK AINV CONSTRUCTION

We restrict our attention to the left-looking block AINV. The (often better) right-looking algorithm will be treated elsewhere, as well as the case of nonsymmetric matrices.

Consider the coefficient matrix partitioned in the block form as follows:

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} & \cdots & \boldsymbol{A}_{1N} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} & \cdots & \boldsymbol{A}_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}_{N1} & \boldsymbol{A}_{N2} & \cdots & \boldsymbol{A}_{NN} \end{pmatrix}.$$

Here $\boldsymbol{A}_{ij}$ has order $n_i \times n_j$, where $1 \leq n_i \leq n$. N is the block dimension of the matrix, n is its dimension. Denote $\sum_{j<i} n_j$ by p_i (it is the offset of the i-th block). Denote also the block rows of $\boldsymbol{A}$ by $\boldsymbol{A}_i, i = 1, \ldots, N$. Note that the diagonal matrix blocks are square symmetric positive definite matrices.

The block AINV algorithm computes block partitioned $\boldsymbol{Z}$ and $\boldsymbol{D}$ directly from $\boldsymbol{A}$ by means of a block incomplete conjugation process applied to the block unit basis vectors. In this process, blocks with small norms are dropped to preserve sparsity. In our case we used the infinity norm for the matrix blocks.

The basic block $\boldsymbol{A}$-conjugation procedure can be written as follows.

Algorithm 4.1 Block $\boldsymbol{A}$-conjugation algorithm:

(1) Let $\boldsymbol{Z}_i^{(0)} = (\boldsymbol{0}_{p_i},\ \boldsymbol{I}_{n_i},\ \boldsymbol{0}_{n-p_{i+1}})^T \quad (1 \leq i \leq N)$
(2) For $i = 1, 2, \ldots, N$ *do*
(3) *For* $j = 1, \ldots, i-1$ *do*
(4) $\boldsymbol{P}_i^{(j-1)} := \boldsymbol{A}_j^T \boldsymbol{Z}_i^{(j-1)}$
(5) $\boldsymbol{Z}_i^{(j)} = \boldsymbol{Z}_i^{(j-1)} - \boldsymbol{Z}_j^{(j-1)} {\boldsymbol{P}_j^{(j-1)}}^{-1} \boldsymbol{P}_i^{(j-1)}$
(6) *End do*
(6) $\boldsymbol{P}_i^{(i-1)} = \boldsymbol{A}_i^T \boldsymbol{Z}_i^{(i-1)}$
(7) *End do*
(8) End do
(9) Let $\boldsymbol{Z}_i := \boldsymbol{Z}_i^{(i-1)}$ *and* $\boldsymbol{D}_i := \boldsymbol{P}_i^{(i-1)}$, *for* $1 \leq i \leq N$. *Return*
$\boldsymbol{Z} = [\boldsymbol{Z}_1,\ \boldsymbol{Z}_2, \ldots, \boldsymbol{Z}_N]$ *and* $\boldsymbol{D} = \mathrm{diag}(\boldsymbol{D}_1, \boldsymbol{D}_2, \ldots, \boldsymbol{D}_N)$.

The block stabilized AINV algorithm (see [4]) is obtained from Algorithm 1 by replacing the computation of $\boldsymbol{P}_i^{(j-1)}$ and $\boldsymbol{P}_i^{(i-1)}$ by $\boldsymbol{Z}_i^{(i-1)} \boldsymbol{A} \boldsymbol{Z}_i^{(j-1)}$ and $\boldsymbol{Z}_i^{(i-1)} \boldsymbol{A} \boldsymbol{Z}_i^{(i-1)}$, respectively.

5 NUMERICAL EXAMPLES

In this section the performance of various AINV-preconditioners is discussed. Rather comprehensive sets of test data for the performance of AINV preconditioning and its stabilized versions can also be found in [4, 7, 8, 9].

All the computations reported here have been performed on a single processor of an SGI Origin 2000 at the Center for Scientific Computing, Espoo, Finland. Convergence of the iterative process is achieved when $\|\boldsymbol{r}_i\| = \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_i\| \leq RTOL\|\boldsymbol{b}\| + ATOL$, and the values of $RTOL = 10^{-5}$ and $ATOL = 10^{-9}$ have been used in all test runs reported here.

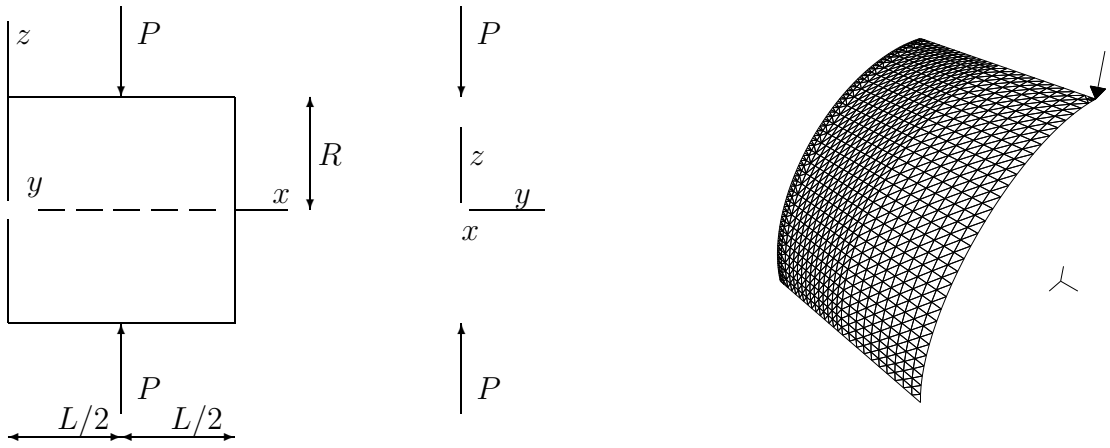

Figure 1: Pinched cylindrical shell.

5.1 Cylindrical shell

Matrices arising from the finite element discretizations of thin shells have been found to be generally difficult to solve by preconditioned iterations, especially cases which cover bending-dominated deformations of the shell.

In [5] the standard AINV approach was tested on cylindrical shell examples with modest results. The reason was identified to be pivot modifications needed to avoid negative or zero pivots. However, it will be demonstrated here that the stabilized AINV and especially the block stabilized AINV preconditioners are successful in solving these problems.

The example case which will be considered in the following corresponds to the matrix `S3RMT3M3` in the Matrix Market set CYLSHELL.[1] It corresponds to a well-known shell element test where a cylinder is loaded by two normal and equal point loads applied centrally at the opposite sides of the cylindrical surface.

The length of the shell is chosen to be equal to its diameter ($L = 2R$) and the Poisson's ratio is 0.3. The shell is thin, the radius to thickness ratio is $R/t = 10^3$. One octant of the shell is discretized in 1666 triangular 3-noded elements resulting in 5357 unknowns, see fig. 1.

The shell element is a facet-type 3-node triangular element using the Hughes–Brezzi formulation [12] for drilling rotations. Also, an Allman-type displacement field [2] is added to the in-plane interpolation to improve the coarse mesh accuracy of the element. The plate bending part of the element is based on the stabilized MITC theory [14] and the stabilization parameter has the value 0.4.

The value of the regularizing penalty parameter γ used in the formulation of Hughes

[1] `http://math.nist.gov/MatrixMarket/data/misc/cylshell/cylshell.html`

Table 1: Matrix `S3RMT3M3`, performance of various AINV algorithms, MMD nodal ordering. α is a shift, i.e. the preconditioner is computed from the shifted matrix $\boldsymbol{A} + \alpha\text{diag}(\boldsymbol{A})$. ψ, ρ are the drop tolerance and preconditioner density, respectively. $n = 5357, nz = 207695, N = 938$.

preconditioner	scaling	ψ	ρ	iterations	notes
static AINV(0)	-	-	1.000	1834	$\alpha = 0.01$
block AINV	-	0.01	0.900	956	
	-	0.005	2.888	440	
	-	0.0025	4.821	294	
block AINV	Jacobi	0.5	1.181	873	
	Jacobi	0.25	2.066	568	
	Jacobi	0.1	4.674	298	
SAINV	Jacobi	0.05	0.952	1667	
	Jacobi	0.025	1.838	1162	
	Jacobi	0.01	4.078	736	
block SAINV	-	0.01	0.707	1081	
	-	0.005	1.820	668	
	-	0.0025	4.555	296	
block SAINV	Jacobi	0.5	0.590	1000	
	Jacobi	0.25	1.126	794	
	Jacobi	0.1	2.764	402	

and Brezzi affects the condition number of the stiffness matrix and thus the convergence of the PCG iteration. The effect of the parameter γ on the spectral condition number as well as the convergence of the PCG iteration will be demonstrated in [6]. In this case the value is equal to $\gamma = 10^{-3}G$, where G is the shear modulus.

Performance of the pointwise right-looking and the block left-looking AINV preconditioners in the `S3RMT3M3` case are shown in Table 1. In this case the standard AINV algorithm works when the block strategy is used. It can also be seen from the figures, that the block version is not very sensitive to scaling. However, there is evidence which is in favour of Jacobi scaling also for the block algorithm in more difficult problems [6]. The pointwise SAINV performs satisfactorily in this example, although it is consistently worse than the block version. The block version seems to be less sensitive to nodal orderings than the pointwise versions [6].

For comparison, results with a robust incomplete Cholesky preconditioner by Ajiz and Jennings [1] are shown in Table 2. It can be seen that the block AINV results in similar convergence rates. However, incomplete Cholesky preconditioners are difficult to parallelize.

Table 2: Matrix `S3RMT3M3`, performance of the A-J preconditioner with RCM nodal ordering.

ψ	ρ	iter	ψ	ρ	iter	ψ	ρ	iter
0.01	0.747	1464	0.001	1.837	585	0.0001	3.993	203

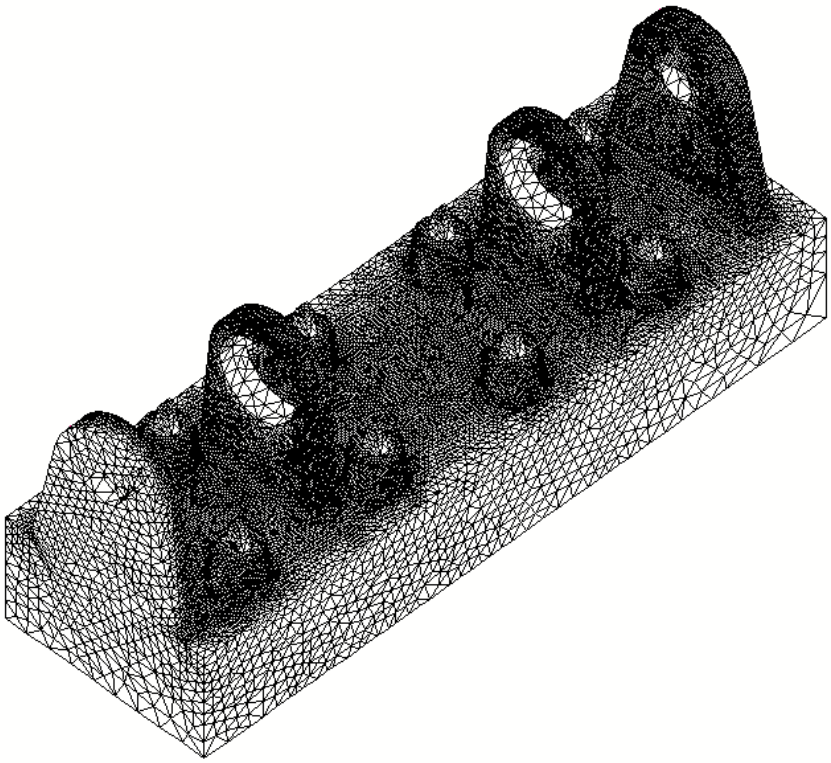

Figure 2: Engine head, 151483 linear tetrahedral elements, 48989 nodes.

5.2 Engine head

In figure 2 the finite element model of an engine head is shown. Unstructured discretization in 151483 linear 4-node tetrahedral elements results in 143571 unknowns (number of nonzero elements in the stiffness matrix $nz = 4706073$). This problem is an easy one to solve with preconditioned iterative methods. The standard AINV approach as well as IC preconditioners can be constructed without shifting. In Table 3 some results from computations are shown where the load has been a thermal load from one centigrade temperature increase. The preconditioners used in this comparison are: standard right-looking AINV, left-looking ILUT-type AINV, right-looking SAINV, block left-looking AINV, block left-looking SAINV and block symmetric-ILUT, standard no-fill IC and Ajiz-Jennings type block IC. MMD reordering of nodes has been used in computations with the AINV algo-

ritmn, and the Gibbs-King profile reduction resulted in smallest RMS-bandwidth (2566), thus it was used for the IC-type preconditioners.

In Table 3 the column LSize indicates the maximum number of entries in each column for the ILUT-type AINV approach and the maximum number of blocks in addition to the blocks of the matrix itself in each column for the block symmetric ILUT-type preconditioner.

Since the CPU-times in RISC-architecture based machines is not independent of the workload of the computer, the CPU-timings shown are averages from three runs.

Table 3: Engine head, $n = 143571, nz = 4706073, N = 47857$.

preconditioner	ψ	ρ	iterations	P-time	I-time	P+I	LSize
Right-looking AINV preconditioners							
AINV	0.1	0.302	637	10	292	302	-
AINV	0.05	0.631	536	17	264	281	-
AINV	0.025	1.219	414	32	264	296	-
AINV	0.01	2.651	284	86	268	354	-
SAINV	0.1	0.291	568	15	226	241	-
SAINV	0.05	0.628	427	26	209	235	-
SAINV	0.025	1.240	375	51	236	287	-
Left-looking AINV preconditioners							
ILUT-type AINV	0.01	0.792	798	104	506	610	20
ILUT-type AINV	0.01	1.058	733	112	448	560	30
ILUT-type AINV	0.01	1.280	589	120	706	826	40
block AINV	0.1	0.801	756	15	659	674	-
block SAINV	0.1	0.551	687	39	492	531	-
Incomplete factorization-based preconditioners							
IC(0)	-	1.000	379	2	193	195	-
block IC(0)	-	1.000	320	3	243	246	-
block Ajiz-Jennings	0.01	1.604	192	7	165	172	-
block symm. ILUT	0.1	0.460	1414	2	819	821	10*
block symm. ILUT	0.01	1.339	211	5	230	235	10*

* different meaning than in ILUT-type AINV, see text.

In this serial implementation, the block version of the Ajiz-Jennings scheme gives the fastest execution. However, the basic no-fill IC performs well as do the right-looking pointwise AINV algorithms if the preconditioner density is relatively small.

5.3 Surface mounted transistor

In figure 3 the finite element model from a thermal stress analysis of a surface mounted transistor is shown. Only half of the transistor is modelled with a piece of a printing wiring board (PWB). The model consists of 1704 reduced triquadratic brick elements (20 node serendipity elements) resulting in 25710 unknowns. Due to the quadratic interpolation the number of nonzero elements in the stiffness matrix is almost the same as in the engine head problem, i.e. $nz = 3753184$.

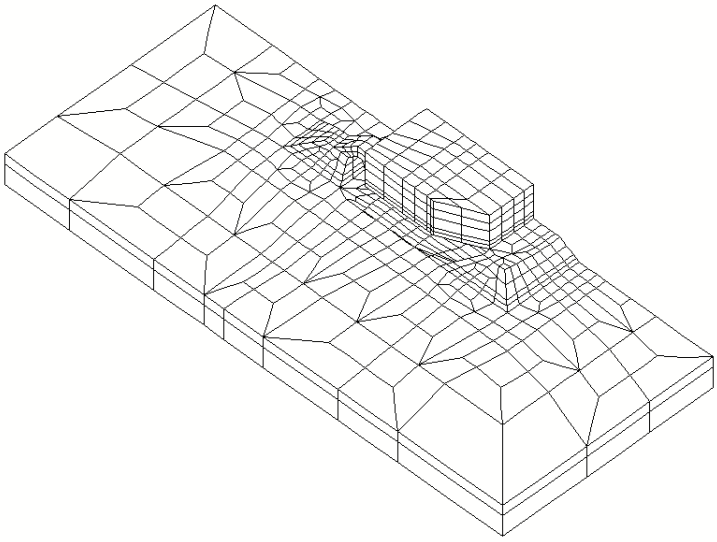

Figure 3: Surface mounted transistor, 1704 triquadratic elements, 8744 nodes.

This problem is more difficult for preconditioning than the previous example, and the standard IC and AINV approaches need shifting to behave well.

Table 4: Surface mounted transistor, $n = 25710, nz = 3753184, N = 8744$.

preconditioner	ordering	ψ	ρ	iterations	P-time	I-time	P+I
block IC(0)	GK	-	1.000	339	30	203	233
block Ajiz-Jennings	GK	0.01	1.189	286	11	149	160
SAINV	MMD	0.025	0.516	348	33	103	136
SAINV	QMD	0.025	0.466	325	30	92	122
SAINV	GND	0.025	0.458	330	30	93	123

There is no big difference between nodal reorderings like MMD, QMD and GND, to the preconditioner density and to the number of iterations. The computing times are also

comparable, in this case the QMD and GND orderings give roughly 10 % shorter iteration time. Results are shown in Table 4. This phenomenon has also been noticed for other examples (in many cases the MMD ordering gives the fastest execution, cf. [9]) and it can be explained in terms of better cache use.

The Gibbs-King reordering gave the smallest RMS-bandwidth (1231) and it was used with incomplete factorization based preconditioners. The symmetric ILUT-type preconditioner was unsuccessful; without shifting the matrix it was impossible to find working pair of parameters ψ and LSize.

In this example the SAINV preconditioner results in similar convergence rates than the IC-based preconditioners although the preconditioner is only half of the density of the IC-type methods.

6 CONCLUDING REMARKS

The stabilized AINV approach has been shown to result in robust and versatile preconditioners for large-scale structural analysis, even on scalar computers. For thin shell applications the block stabilized AINV gives considerable improvement in convergence rate over the pointwise version of the SAINV approach. The multiple minimum degree, the quotient minimun degree and the generalized nested dissection orderings give with the same AINV-preconditioner density similar convergence rates and the computing times are within 10 % range. On cache-based architectures, the use of blocking is also beneficial in terms of performance (cf. also [6]).

REFERENCES

[1] M. A. Ajiz and A. Jennings. A robust incomplete Cholesky conjugate gradient algorithm, *Int. J. Num. Meth. Engng*, 20:949–966, 1984.

[2] D. J. Allman. A compatible triangular element including vertex rotations for plane elasticity analysis, *Computers and Structures*, 19:1–8, 1984.

[3] C. Ashcraft. Compressed graphs and the minimum degree algorithms, *SIAM J. Sci. Comput.*, 16:1404–1411, 1995.

[4] M. Benzi, J. K. Cullum and M. Tůma. Robust approximate inverse preconditioning for the conjugate gradient method, *SIAM J. Sci. Comput.*, to appear.

[5] M. Benzi, R. Kouhia and M. Tůma. An assessment of some preconditioning techniques in shell problems, *Comm. Numer. Meth. Engng*, 14:897-906, 1998.

[6] M. Benzi, R. Kouhia and M. Tůma. On conjugation-based factorized approximate inverse preconditioners in solid and structural mechanics, in preparation.

[7] M. Benzi, C. D. Meyer and M. Tůma. A sparse approximate inverse preconditioner for the conjugate gradient method, *SIAM J. Sci. Comput.*, 17:1135–1149, 1996.

[8] M. Benzi and M. Tůma. A sparse approximate inverse preconditioner for nonsymmetric linear systems, *SIAM J. Sci. Comput.*, 19:968–994, 1998.

[9] M. Benzi and M. Tůma. Orderings for factorized sparse approximate inverse preconditioners, *SIAM J. Sci. Comput.*, to appear.

[10] R. Bridson and W.-P. Tang. Ordering, anisotropy and factored sparse approximate inverses, *SIAM J. Sci. Comput.*, 21:867–882, 1999.

[11] M. R. Field. Improving the performance of factorized sparse approximate inverse preconditioner, Technical Report HDL-TR-98-199, Dublin, Ireland, 1998.

[12] T. J. R. Hughes and F. Brezzi. On drilling degrees of freedom, *Comp. Meth. Appl. Mech. Engng*, 72:105–121, 1989.

[13] S. A. Kharchenko, L. Yu. Kolotilina, A. A. Nikishin and A. Yu. Yeremin. A reliable AINV-type preconditioning method for constructing sparse approximate inverse preconditioners in factored form. Preprint, 1999.

[14] M. Lyly, R. Stenberg, and T. Vihinen. A stable bilinear element for the Reissner-Mindlin plate model, *Comp. Meth. Appl. Mech. Engng*, 110:343–357, 1993.