COMMUNICATIONS IN NUMERICAL METHODS IN ENGINEERING
Commun. Numer. Meth. Engng, **14**, 897–906 (1998)

AN ASSESSMENT OF SOME PRECONDITIONING TECHNIQUES IN SHELL PROBLEMS

MICHELE BENZI[1], REIJO KOUHIA[2]* AND MIROSLAV TŮMA[3]

[1]*Scientific Computing Group CIC-19, Los Alamos National Laboratory, Los Alamos, NM 87545, U.S.A.*
[2]*Laboratory of Structural Mechanics, Helsinki University of Technology, P.O. Box 2100, 02015 HUT, Finland*
[3]*Institute of Computer Science, Czech Academy of Sciences, 182 07 Prague 8, Czech Republic*

SUMMARY

Preconditioned Krylov subspace methods have proved to be efficient in solving large, sparse linear systems in many areas of scientific computing. The success of these methods in many cases is due to the existence of good preconditioning techniques. In problems of structural mechanics, like the analysis of heat transfer and deformation of solid bodies, iterative solution of the linear equation system can result in a significant reduction of computing time. Also many preconditioning techniques can be applied to these problems, thus facilitating the choice of an optimal preconditioning on the particular computer architecture available.

However, in the analysis of thin shells the situation is not so transparent. It is well known that the stiffness matrices generated by the FE discretization of thin shells are very ill-conditioned. Thus, many preconditioning techniques fail to converge or they converge too slowly to be competitive with direct solvers. In this study, the performance of some general preconditioning techniques on shell problems is examined.

KEY WORDS preconditioning; conjugate gradient; finite elements; shells

PRECONDITIONING TECHNIQUES

In this paper, the iterative solution of the discretized equilibrium equations of shells, written as a system of linear equations

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{1}$$

is considered. The stiffness matrix $\boldsymbol{A}$ is large, sparse and symmetric. In most cases it is also positive definite, but can be indefinite at some stage in non-linear analyses. However, the number of negative eigenvalues is usually small, mostly one, on unstable equilibrium paths which are of practical interest. The preconditioned conjugate gradient (PCG) method is a powerful tool for solving such a system. Even though it is foolproof only for positive definite matrices, in practice it usually performs well without breakdowns.

To speed up the convergence of the conjugate gradient iteration, preconditioning of the original system (1) is introduced:

$$\boldsymbol{M}^{-1}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{M}^{-1}\boldsymbol{b}$$

* Correspondence to: R. Kouhia, Laboratory of Structural Mechanics, Helsinki University of Technology, PO Box 2100, FIN 02015 HUT, Finland; e-mail reijo.kouhia@hut.fi

Contract grant sponsor: Czech Grant Agency; Contract grant numbers: GA AS CR 205/96/0921; A2030706.

CCC 1069–8299/98/100897–10$17·50

Received 3 October 1997
Accepted 24 February 1998

where $\boldsymbol{M}$ is the preconditioner matrix, which should be symmetric and positive definite. The purpose of the preconditioner is to reduce the condition number of the problem. Even more important is the ability to cluster the eigenvalues. A good preconditioner should have close resemblance to $\boldsymbol{A}$, should be easy to construct and apply, and should have low storage requirements. Usually these demands are in conflict with each other, thus leaving the possibility to compromise. For a thorough treatment of preconditioned iterative methods, see References 1 and 2.

The most general preconditioning strategies can be grouped into classes: (a) preconditioners based on classical iterations like Jacobi, SSOR; (b) sparse incomplete Cholesky decompositions (IC); (c) polynomial preconditioners; (d) sparse approximate inverse preconditioners; (e) multigrid or multilevel preconditioners. Most preconditioning techniques can also be classified as being implicit or explicit. A preconditioner is implicit if its application requires the solution of a linear system. For explicit methods the preconditioning operation reduces to forming matrix–vector products.

It should be stressed that the effectiveness of a preconditioning strategy is highly problem- and architecture-dependent. For instance, incomplete factorizations (implicit) are difficult to implement on vector and parallel computers, due to the sequential nature of the triangular solution. On the other hand, sparse approximate inverse preconditioners (explicit) need only matrix–vector products, which are relatively easy to vectorize and parallelize, but they often result in slower convergence rates and are usually not as robust as IC factorization-based strategies.

In this paper, the following preconditioning techniques are considered:

(a) Jacobi, SSOR
(b) implicit incomplete factorizations:
  (i) level-fill incomplete Cholesky IC(k), $k = 0, 1, 2, \ldots$, no-fill IC = IC(0)
  (ii) drop tolerance-based incomplete Cholesky IC(ψ)
(c) explicit incomplete factorizations based on truncated Neumann expansion
(d) least-squares polynomial preconditioners
(e) explicit sparse approximate inverse preconditioners:
  (i) Frobenius norm-based preconditioner FSAI[3]
  (ii) incomplete conjugation-based preconditioner AINV.[4]

In this paper we do not consider multigrid or multilevel preconditioners, although in practice these methods seem to be quite effective.[5] Also the element-by-element approaches, which are potentially attractive on parallel computers, are not considered.

The Jacobi, SSOR and incomplete Cholesky factorization preconditioners are well known and will not be described here. The polynomial preconditioners considered in the present study are based on least-squares polynomial approximation with Jacobi weight function; see Reference 2.

The explicit methods based on truncated Neumann expansions are similar to the methods proposed in Reference 6; see also Reference 7. The SSOR and incomplete Cholesky preconditioners require, at each step of the PCG method, the solution of triangular linear systems of the form $(\boldsymbol{I} - \boldsymbol{L})\boldsymbol{y} = \boldsymbol{d}$, where $\boldsymbol{L}$ is a strictly lower triangular matrix. The solution can be written explicitly as

$$\boldsymbol{y} = (\boldsymbol{I} - \boldsymbol{L})^{-1}\boldsymbol{d} = (\boldsymbol{I} + \boldsymbol{L} + \boldsymbol{L}^2 + \cdots + \boldsymbol{L}^{n-1})\boldsymbol{d}$$

where n is the dimension of the system. This suggests that an approximate solution $\hat{\boldsymbol{y}} \approx \boldsymbol{y}$ be computed as

$$\hat{\boldsymbol{y}} = (\boldsymbol{I} + \boldsymbol{L} + \boldsymbol{L}^2 + \cdots + \boldsymbol{L}^m)\boldsymbol{d}$$

where the integer m is much smaller than n. The advantage of this approach is that the highly recursive back-substitution process is now replaced by sparse matrix–vector products, which are relatively straightforward to vectorize and parallelize. For this approach to be cost-effective, it is important that m be kept small, say $m = 3$ or less. Thus, unless the Neumann expansion converges rapidly to $(\boldsymbol{I} - \boldsymbol{L})^{-1}$, the approximation may be a poor one and a degradation of the rate of convergence, as compared with the one obtained with an exact solution of $(\boldsymbol{I} - \boldsymbol{L})\boldsymbol{y} = \boldsymbol{d}$, is to be expected. In other terms, these explicit variants of SSOR and IC preconditioning can be expected to work well only if $\boldsymbol{L}$ is relatively small in norm. This is typically the case when $\boldsymbol{A}$ is strictly diagonally dominant. Unfortunately, finite element discretizations do not usually result in diagonally dominant stiffness matrices.

Sparse approximate inverse preconditioners have recently received considerable attention, mainly because of their good vectorization and parallelization properties. These techniques are based on the explicit construction of a sparse matrix $\boldsymbol{M}^{-1}$ which directly approximates $\boldsymbol{A}^{-1}$. This is in contrast with more traditional implicit techniques where the matrix $\boldsymbol{M}$, rather than $\boldsymbol{M}^{-1}$, is explicitly available. The preconditioning step with an approximate inverse preconditioner $\boldsymbol{M}^{-1}$ only requires matrix–vector products, and is easily implemented on vector and parallel architectures. On the other hand, the construction of the preconditioner itself can be time-consuming, and the convergence rates obtained are often not as good as those obtained with implicit techniques.

For use with the PCG method, it is imperative that the preconditioner $\boldsymbol{M}^{-1}$ be a symmetric positive-definite matrix. One way to ensure this is to express the preconditioner as the product of a triangular matrix and its transpose. To this end, *factorized* sparse approximate inverse preconditioners were proposed in Reference 3. With this approach, commonly referred to as the FSAI method, a lower triangular matrix $\boldsymbol{G}$ is computed as the (unique) solution of the constrained minimization problem

$$\text{minimize } \|\boldsymbol{I} - \boldsymbol{G}\boldsymbol{L}\| \text{ subject to } \boldsymbol{G} \in \mathcal{L}$$

where $\boldsymbol{L}$ now denotes the Cholesky factor of $\boldsymbol{A}$ and $\mathcal{L}$ is a set of lower triangular matrices with a prescribed non-zero pattern (which must include the main diagonal). Here the matrix norm is the Frobenius norm or some weighted variant of it. Remarkably, it is possible to solve the above minimization problem without any knowledge of $\boldsymbol{L}$, just working with the original matrix $\boldsymbol{A}$; see Reference 3. The minimization problem decouples in n independent linear systems of relatively small size which can be solved in parallel. The approximate inverse preconditioner is then $\boldsymbol{M}^{-1} = \boldsymbol{G}^{\mathrm{T}}\boldsymbol{G}$. The main difficulty associated with this approach is the choice of the sparsity pattern of $\boldsymbol{G}$, i.e. the determination of the constraint set $\mathcal{L}$. A simple solution is to restrict $\boldsymbol{G}$ to have the same sparsity pattern as the lower triangular part of $\boldsymbol{A}$, but this choice works well only for simple problems. Non-zero patterns associated with higher powers of $\boldsymbol{A}$ could also be used, but then the costs associated with the preconditioner construction and application increase. Moreover, for difficult problems even this more expensive approach may be ineffective.

Another approach to factorized approximate inverse preconditioning was proposed in Reference 4. This approach, which does not require that the sparsity pattern be known in

advance, is based on an A-orthogonalization process — that is, a Gram–Schmidt process with respect to the energy inner product $\langle x, y\rangle = x^{\mathrm{T}}Ay$. Given A and an arbitrary set of n linearly independent vectors, this algorithm computes a set of n vectors $\{z_i\}_{i=1}^n$ which are conjugate with respect to A, i.e. A-orthogonal. If we introduce the matrix $Z = [z_1, z_2, \ldots, z_n]$, then

$$Z^{\mathrm{T}}AZ = D = \mathrm{diag}(p_1, p_2, \ldots, p_n)$$

where $p_i = z_i^{\mathrm{T}}Az_i \neq 0$. It follows that

$$A^{-1} = ZD^{-1}Z^{\mathrm{T}} = \sum_{i=1}^{n} \frac{z_i z_i^{\mathrm{T}}}{p_i}$$

and a factorized form of A^{-1} is obtained.

When the A-orthogonalization process is applied to the standard basis vectors $e_1, \ldots, e_n$, it is easy to see that Z is unit upper triangular, and indeed $Z = L^{-\mathrm{T}}$, where $A = LDL^{\mathrm{T}}$ is the root-free Cholesky factorization of A. If a_i^{T} denotes the ith row of A, the A-orthogonalization process for computing Z and D can be written as follows:

(1) Let $z_1^{(0)} = e_1$; $p_1^{(0)} = a_{11}$
(2) **for** $i = 2, \ldots, n$
  $z_i^{(0)} = e_i$
  for $j = 1, \ldots, i-1$
    $p_i^{(j-1)} := a_j^{\mathrm{T}} z_i^{(j-1)}$

    $z_i^{(j)} := z_i^{(j-1)} - \left(\frac{p_i^{(j-1)}}{p_j^{(j-1)}}\right) z_j^{(j-1)}$

  end
  $p_i^{(i-1)} := a_i^{\mathrm{T}} z_i^{(i-1)}$
end

To get a sparse preconditioner, Z is computed incompletely, by dropping entries in the vector update operations. This can be done either on the basis of position, whereby non-zero entries outside a prescribed non-zero pattern are dropped, or on the basis of magnitude, whereby non-zeros are dropped if smaller than a prescribed drop tolerance in absolute value. This leads to approximate factors $\bar{Z} \approx Z$ and $\bar{D} \approx D$, and a factorized approximate inverse is obtained as $M^{-1} = \bar{Z}\bar{D}^{-1}\bar{Z}^{\mathrm{T}}$. The stability of this procedure for certain classes of matrices, including diagonally dominant ones, was proved in Reference 4. In addition, numerical experiments in References 4, 8 and 9 showed that this approach performs well on linear systems arising from various applications, such as the discretization by finite differences of elliptic partial differential equations and the finite element analysis of simple structures. In particular, the experiments in References 8 and 9 showed that on vector computers this technique can be superior to IC methods because of good vectorization properties.

To our knowledge, until now there have been no reports of the application of explicit preconditioners to shell problems.

NUMERICAL EXPERIMENTS

In this Section, the results of computations with one commonly used 'simple' shell problem are presented. The shell elements used in this study are facet-type 3-node triangular or 4-node quadrilateral elements using the Hughes–Brezzi formulation[10] for drilling rotations. The plate bending part of the element is based on the stabilized MITC theory[11] or the discrete Kirchhoff condensation. In the MITC formulation the stabilization parameter has been 0·4 for both triangular and quadrilater elements.

The value of the regularizing penalty parameter γ used in the formulation of Hughes 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 is demonstrated in Reference 12. Here the value $\gamma = G/1000$ is chosen, where G is the shear modulus. Adding an Allman-type displacement field[13] to the in-plane interpolation also has an effect on the convergence; however, there seems to be no definite trend in that dependency.[12]

A well-known shell element test is the pinched cylinder; see, for example, Reference 14. The 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 ratio is 0·3. In the computations the following dimensions are chosen: $R = 1$ unit and Young's modulus $= 10^6$ units. The performance of the preconditioning techniques is studied with respect to the relative thickness and some relevant parameters in the finite element model. As expected, the problem gets harder when the thickness to radius ratio, i.e. the characteristic thickness, gets smaller. Three cases, $R/t = 10$, 10^2 and 10^3, are examined, but only the results from the extreme ones are presented. One octant of the shell is discretized by uniform 30×30 (see Figure 1) or 150×100 triangular or quadrilateral meshes resulting in 5489 or 90499 unknowns, respectively. The corresponding matrices can be obtained from the Matrix Market under the set CYLSHELL; (`http://math.nist.gov/MatrixMarket/data/misc/cylshell/cylshell.html`). The examples shown correspond to the matrices `S1RMQ4M1`, `S3RMQ4M1` and `S3DKQ4M2` in the CYLSHELL set.

Some characteristics of the stiffness matrices are recorded in Table I, including the problem size n, the number of non-zeros $nz(\boldsymbol{A})$, and the spectral condition number. Results of the

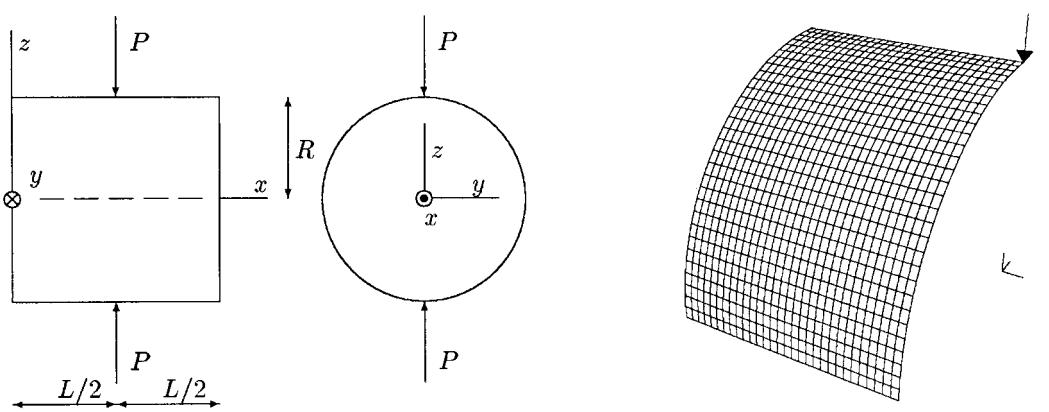

Figure 1. Uniform 30×30 mesh for an octant of a pinched cylindrical shell

Table I. Cylindrical shells

Case	R/t	Mesh	Element	n	$nz(\boldsymbol{A})$	cond($\boldsymbol{A}$)
S1RMQ4M1	10	30×30	stab. MITC4	5489	143300	$1{\cdot}81 \times 10^{6}$
S3RMQ4M1	1000	30×30	stab. MITC4	5489	143300	$1{\cdot}77 \times 10^{10}$
S3DKQ4M2	1000	150×100	DKQ	90499	2455670	$1{\cdot}90 \times 10^{11}$

Table II. Cylindrical shell: P = preconditioner evaluation time, I = iteration time

(a) $R/t = 10$, stab. MITC4 elements, 30×30 mesh, problem S1RMQ4M1

		Cray			SGI Power Challenge			
Preconditioner	Iter.	P	I	P + I	P	I	P + I	$nz(\boldsymbol{M})$
expl IC(0·04)	467	0·33	2·50	2·83	0·35	20·46	20·81	19989
IC(0)	102	1·40	1·94	3·36	1·00	8·35	9·35	143300
Jacobi	796	0·05	4·08	4·13	0·11	32·59	32·70	5489
IC(0·04)	322	0·33	4·17	4·50	0·33	15·94	16·27	19989
FSAI(0)	285	3·50	2·89	6·39	4·80	23·10	27·90	143300
IC(1)	55	5·88	1·18	7·06	4·71	5·42	10·13	202514
AINV(0)	314	4·95	3·38	8·33	6·27	25·72	31·99	143300
IC(2)	36	9·28	0·81	10·09	7·62	4·17	11·78	259613
IC(3)	27	13·28	0·63	13·85	10·98	3·46	14·44	314597

(b) $R/t = 10^3$, stab. MITC4 elements, 30×30 mesh, problem S3RMQ4M1

		DEC AlphaServer			SGI Power Challenge			
Preconditioner	Iter.							$nz(\boldsymbol{M})$
IC(0)	177	0·33	6·61	6·94	1·00	14·78	15·78	143400
IC(1)	61	0·93	3·34	4·27	4·71	6·24	10·95	202514
IC(2)	57	1·00	2·98	3·98	7·62	6·62	14·24	259613
IC(3)	44	1·41	2·81	4·22	10·98	6·04	17·02	314597
FSAI(0)	523	0·87	22·79	23·36	4·80	44·85	49·65	143400

(c) $R/t = 10^3$, DKQ elements, 150×100 mesh, problem S3DKQ4M2

		DEC AlphaServer			
Preconditioner	Iter.	P	I	P + I	$nz(\boldsymbol{M})$
IC(1)	1563	16	1883	1899	3513260
IC(2)	549	19	549	568	4560113
IC(3)	347	27	468	495	5596229
IC(4)	380	27	490	517	5616203

computations are shown in Table II. The reported computing times are obtained with a Cray C-90 vector processor (Météo France, Toulouse), an SGI Power Challenge and a DEC AlphaServer 8400 (Center for Scientific Computing, Espoo, Finland). Only converged results are shown and the list is presented for increasing total solution times on the Cray. The maximum number of allowed iterations is 2000.

It should be noted that all tested preconditioners could be built. Two strategies for shifting small or negative pivots have been used: a local strategy in which small pivots are shifted whenever they occur during the decomposition phase[4] and Manteuffel's shifting method[15] in which the decomposition is carried out for the shifted matrix $\boldsymbol{A} + \alpha\mathrm{diag}(\boldsymbol{A})$, where α is a parameter. From the numerical experiments it can be concluded that Manteuffel's diagonal shifting strategy is more robust; however, it should be kept in mind that finding a good value for the shift α is a non-trivial task. In practice, several incomplete factorizations with different trial values of α may be needed before a satisfactory preconditioner is obtained. This makes Manteuffel's shifting strategy costly to use in many cases.

We have also tested the robust preconditioner by Ajiz and Jennings,[16] which is referred to as *corrected IC* (CIC) by Saint-Georges *et al.*[17] These authors concluded that CIC performs satisfactorily in thin shell problems. However, our conclusions are more concordant with the observations by Dickinson and Forsyth:[18] this method, although reliable, results in too large modifications to the diagonal, thus slowing down the convergence. Dickinson and Forsyth[18] analysed the behaviour of different IC preconditioners in three-dimensional elasticity problems. Their conclusions regarding the drop tolerance and level-of-fill based preconditioners are consistent with other reports dealing with second-order elliptic problems. However, it should be remembered that shell problems are much more difficult for preconditioned iterations than two- or three-dimensional elasticity problems. One reason for that is the more rapid growth of the spectral condition number, which is of order $n^2 \sim h^{-4}$ as compared to second-order problems: $\mathrm{cond}(\boldsymbol{A}) \sim n^{2/d} \sim h^{-2}$, where d is the space dimension, n the dimension of $\boldsymbol{A}$ and h is the mesh parameter, i.e. the width of the largest element.

The stiffness matrix and the explicit preconditioners are stored by using the *jagged diagonal scheme* in order to take advantage of vectorization in the matrix–vector products.[2] In addition, no advantage of symmetry is taken to save storage for the stiffness matrix, as this is known to cause a degradation of the performance on supercomputers. For the dot products and the vector updates in the CG algorithm, standard BLAS are used. The stiffness matrix is rescaled by dividing its elements by the absolute value of the largest non-zero entry. With this scaling, the drop tolerance can usually be taken to be a number between 0 and 1.

The number of iterations shown in Table II corresponds to computations where the right-hand side is determined from the solution vector of all ones. If the point loading is used as in Reference 14, somewhat different iteration counts are obtained. In Table III the number of iterations with this loading are shown for a few choices of the preconditioner as well as the deflection w_P under the loaded point (the last unknown). To have a finite value for the limiting deflection for the Kirchhoff model when $t \to 0$, the load is scaled with respect to the characteristic thickness of the shell. The value used is $P = -(1/4)(E/10)(t^3/R)$, where E is Young's modulus. For the Reissner–Mindlin kinematical model, the point load is not allowed; that is, $w_P \to \infty$ when the

Table III. Pinched cylindrical shells, n_P = equation number for the loaded point, w_P = deflection under the point load

Case	R/t	Mesh	Element	Precond.	n_P	w_P	Iter.
S1RMQ4M1	10	30×30	stab. MITC4	IC(0)	5489	$-4{\cdot}7985 \times 10^{-2}$	109
S3RMQ4M1	1000	30×30	stab. MITC4	IC(0)	5489	$-4{\cdot}0881 \times 10^{-2}$	177
S3DKQ4M2	1000	150×100	DKQ	IC(2)	90499	$-4{\cdot}0877 \times 10^{-2}$	442

mesh is refined, i.e. when $h \to 0$. Therefore the finer models are discretized with discrete Kirchhoff-type elements.

The initial guess for the PCG iteration is the zero vector and the stopping criterion is $\|\boldsymbol{r}_k\|_2 < 10^{-9}$, where $\boldsymbol{r}_k$ is the unpreconditioned updated residual after k iterations. Codes are written in standard Fortran 77 and compiled with the -Zv option on the Cray and -O3 option on the SGI and DEC.

In Table II, expl IC(0·04) denotes the explicit preconditioner obtained from a truncated Neumann expansion of first degree ($m = 1$) applied to the incomplete Cholesky factorization with drop tolerance $\psi = 0{\cdot}04$. On the Cray, this was the fastest of all the methods. Notice that the loss of convergence rate due to truncation (from 322 to 467 iterations) is more than compensated by the faster execution of the iterations themselves, which is due to better vector performance. Using an expansion of higher degree ($m \geqslant 2$) results in better convergence rates but higher computing times.

Among the incomplete Cholesky preconditioners based on levels of fill, the no-fill incomplete factorization performed reasonably well. For the easier problem (Table II(a)), IC(0) was the fastest among the IC(k) methods on both the Cray and the Power Challenge; on the latter machine, this was the fastest method overall. The higher-level IC preconditioners, however, become attractive if several problems with the same stiffness matrix and different load vectors have to be solved, because in this case the cost of the construction of the preconditioner would become negligible, and it would pay off to have more rapid convergence. It should be noticed that for IC(k), $k > 0$, the computation of the preconditioner is very time-consuming on the Cray. The reason is that with high k the computation is completely dominated by integer arithmetic, which is needed to keep track of the level count for each new fill-in and there are relatively few floating point operations. On Cray machines the real arithmetic is very fast, but the integer arithmetic is relatively slow.

For the easiest of the three problems (matrix `S1RMQ4M1`) no pivot shifts were needed. However, pivot shifts were used with IC(2) and IC(3) on problem `S3RMQ4M1`, with $\alpha = 2 \times 10^{-4}$ and $\alpha = 10^{-4}$, respectively. Also, a shift $\alpha = 10^{-2}$ was used with IC(1) on problem `S3DKQ4M2`.

For the problem in Table II(a) the simple Jacobi preconditioner, which is not competitive on the Power Challenge, is the third fastest method on the Cray. Considering the ease with which this simple technique is implemented, this is an attractive method on vector and parallel computers. However, it is not recommended if several linear systems with the same stiffness matrix and different load vectors have to be solved, or for difficult problems (thin shells).

In Table II, FSAI(0) and AINV(0) denote, respectively, the sparse approximate inverse preconditioners based on Frobenius norm minimization and on incomplete $\boldsymbol{A}$-orthogonalization with a preset non-zero pattern equal to that of $\boldsymbol{A}$. The computation of these preconditioners is relatively expensive, and the convergence rates are not very good. The stiffness matrix is far from being diagonally dominant, and decay of the entries in $\boldsymbol{A}^{-1}$ is slow. Thus, it is difficult to approximate $\boldsymbol{A}^{-1}$ with a sparse matrix. It should be noticed that even on the Cray, in spite of vectorization, the time for the iterative part is higher for the approximate inverse methods than for the IC(k) methods. This is not only because the convergence rate is slower, but also because the triangular solves with the IC preconditioner do not run at scalar speed, but at vector speed, achieving about half the MFLOPS rate achieved by the approximate inverse preconditioners. This is due to the fact that the incomplete factors are not very sparse, and they are well structured. The average number of non-zeros in each row of the Cholesky factor is large enough for vectorization to have an impact in the back-substitution phase. When the incomplete factors are

very sparse, the approximate inverse techniques are often better than IC on vector computers (see the experiments in References 8 and 9).

Although our aim is not to compare iterative methods with direct ones, we report that the solution times with a standard in-core skyline solver are 1·74 s for the 30×30 model and 530 s for the larger one on the DEC AlphaServer 8400. Thus, for this particular choice of test matrices, computer architectures and implementation, direct solvers are competitive with preconditioned iterative methods. However, we think that PCG would be the method of choice for very large shell problems and different types of architectures, e.g. massively parallel computers.

The difficulty of the shell problems is underscored by the high failure rate encountered in our experiments. The least-squares polynomial preconditioner resulted in extremely slow convergence, and cannot be recommended for this kind of problem. The explicit preconditioners based on truncated Neumann expansions of the inverse SSOR and IC(ψ) factors also failed to give reasonably rapid convergence. The approximate inverse preconditioners based on incomplete $\boldsymbol{A}$-orthogonalization computed with drop tolerances suffered from instability and were not useful. Pivot shifts, analogous to those used in the context of IC preconditioning, were tried but resulted in poor preconditioners. The IC preconditioners based on drop tolerances were also found to be prone to instability, especially when applied to thin shell models.

For these more difficult problems, only the IC preconditioners based on levels of fill performed satisfactorily. However, for problem `S3DKQ4M2` the number of iterations increased when going from level-3 to level-4 incomplete Cholesky preconditioning. This conforms with the well known fact that adding fill-in does not always imply faster convergence.

In addition, some experiments were performed in which the stiffness matrix was reordered with the Reverse Cuthill–McKee heuristic, as it is known that such reordering can be beneficial in some cases (see, for example, Reference 19). Unfortunately, the reordering led to an increase in the number of PCG iterations in all cases.

CONCLUSIONS

In spite of the excellent performance of preconditioned Krylov subspace methods in heat transfer and stress analysis of solid bodies, shells still present preconditioner developers with a challenging task. In the present comparison, the level-based IC factorizations with low levels $k = 1,2$ appeared to be the best ones and the only ones that worked for large models of thin shells. Drop tolerance-based IC strategies, on the other hand, did not perform well. This is in contrast with the situation in other applications, where methods based on drop tolerances are usually more robust and show better performance than incomplete factorizations based on levels of fill. Also, explicit preconditioners based on truncated Neumann expansions and sparse approximate inverse techniques were not competitive with IC methods, generally speaking. However, these techniques may become competitive on parallel computers, at least for easier problems. It should be mentioned that in the present study the sparsity pattern used was far from being optimal (the same as for the original matrix), and better results might be achieved with a more sophisticated choice. However, how to determine a good sparsity pattern for these inverse preconditioners is still an open question. Also, for easier problems Jacobi (diagonal) preconditioning may be attractive, due to its simplicity and good vector and parallel performance. However, for large models of thin shells, finding a good parallelizable preconditioner appears to be a very difficult task.

ACKNOWLEDGEMENTS

Part of this work was performed when the first author was with CERFACS (Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique) in Toulouse, France. The support and excellent research environment provided by CERFACS are gratefully acknowledged. The work of the third author was supported in part by the Czech Grant Agency through grants GA AS CR 205/96/0921 and A2030706.

REFERENCES

1. O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, Cambridge, 1994.
2. Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, Boston, 1996.
3. L. Y. Kolotilina and A. Y. Yeremin, 'Factorized sparse approximate inverse preconditionings I. Theory', *SIAM J. Matrix Anal. Appl.*, **14**, 45–58 (1993).
4. M. Benzi, C. D. Meyer and M. Tůma, 'A sparse approximate inverse preconditioner for the conjugate gradient method', *SIAM J. Sci. Comput.*, **17**(5), 1135–1149 (1996).
5. V. E. Bulgakov, 'The use of the multi-level iterative aggregation method in 3-D finite element analysis of solid, truss, frame and shell structures', *Comput. Struct.*, **63**(5), 927–938 (1997).
6. H. A. van der Vorst, 'A vectorizable variant of some ICCG methods', *SIAM J. Sci. Stat. Comput.*, **3**(3), 350–356 (1982).
7. I. Gustafsson and G. Lindskog, 'Completely parallelizable preconditioning methods', *Numer. Linear Algebr. Appl.*, **2**(5), 447–465 (1995).
8. M. Benzi and M. Tůma, 'Approximate inverse preconditioning for the conjugate gradient method on a vector computer', in K. Segeth (Ed.), *Proceedings of the Prague Mathematical Conference 1996 — PMC'96*, 1996, pp. 29–34.
9. M. Benzi and M. Tůma, 'A comparative study of sparse approximate inverse preconditioners', *Technical Report LA-UR-98-0024*, Los Alamos National Laboratory, Los Alamos, NM, to appear in *Appl. Numer. Math.*
10. T. J. R. Hughes and F. Brezzi, 'On drilling degrees of freedom', *Comput. Methods Appl. Mech. Eng.*, **72**, 105–121 (1989).
11. M. Lyly, R. Stenberg and T. Vihinen, 'A stable bilinear element for the Reissner-Mindlin plate model', *Comput. Methods Appl. Mech. Eng.*, **110**, 343–357 (1993).
12. R. Kouhia, 'Using iterative solvers in non-linear continuation algorithm', in J. Aalto and T. Salmi (Eds.), *Proceedings of the 6th Finnish Mechanics Days*, 1997, pp. 253–267.
13. D. J. Allman, 'A compatible triangular element including vertex rotations for plane elasticity analysis', *Comput. Struct.*, **19**: 1–8 (1984).
14. H. Hakula, Y. Leino and J. Pitkäranta, 'Scale resolution, locking and high-order finite element modelling of shells', *Comput. Methods Appl. Mech. Eng.*, **133**, 157–182 (1996).
15. T. A. Manteuffel, 'An incomplete factorization technique for positive definite linear systems', *Math. Comput.*, **34**, 473–497 (1980).
16. M. A. Ajiz and A. Jennings, 'A robust incomplete Cholesky conjugate gradient algorithm', *Int. j. numer. methods eng.*, **20**, 949–966 (1984).
17. P. Saint-Georges, G. Warzee, Y. Notay and R. Beauwens, 'Fast iterative solvers for finite element analysis in general and shell analysis in particular', in B. H. V. Topping (Ed.), *Advances in Finite Element Technology*, Civil-Comp Press, Edinburgh, 1996, pp. 273–282.
18. J. K. Dickinson and P. A. Forsyth, 'Preconditioned conjugate gradient methods for three-dimensional elasticity', *Int. j. numer. methods eng.*, **37**, 2211–2234 (1994).
19. I. Hladík, M. B. Reed and G. Swoboda, 'Robust preconditioners for linear elasticity FEM analyses', *Int. j. numer. methods eng.*, **40**, 2109–2127 (1997).

