Preconditioned Conjugate Gradient Methods for Three Dimensional Linear Elasticity *

J.K. Dickinson and P.A. Forsyth†

February 9, 1993

Abstract

Finite element modelling of three dimensional elasticity problems gives rise to large sparse matrices. Various preconditioning methods are developed for use in preconditioned conjugate gradient iterative solution techniques. Incomplete factorizations based on levels of fill, drop tolerance, and a two level hierarchical basis are developed. Various techniques for ensuring that the incomplete factors have positive pivots are presented. Computational tests are carried out for problems generated using unstructured tetrahedral meshes. Quadratic basis functions are used. The performance of the iterative methods is compared to a standard direct sparse matrix solver. Problems with up to 70,000 degrees of freedom, and small ($\ll 1$) element aspect ratio are considered.

Keywords: Three dimensional elasticity, preconditioning, hierarchical basis

Running Title: PCG methods for 3-d elasticity

AMS Subject Classification 65F10, 65N30

*This work was supported by the Natural Sciences and Engineering Research Council of Canada, and by the Information Technology Research Center, which is funded by the Province of Ontario.

†Department of Computer Science, University of Waterloo, Waterloo, Ontario, N2L 3G1

1 Introduction

Three dimensional finite element stress analysis gives rise to large, sparse matrices. In most cases, the cost of the finite element analysis is dominated by the cost of solution of the resulting large sparse system. In many fields, such as computational fluid dynamics [1, 2, 3], petroleum reservoir simulation [4, 5] and semiconductor device simulation [6], iterative methods are typically used for solution of three dimensional problems. However, for elasticity problems, direct methods are commonly used.

Of course, for direct methods, the cost of solution and storage requirements increase dramatically for three dimensional problems, as compared to two dimensions. Consequently, there has been a recent upsurge of interest in applications of iterative methods for three dimensional elasticity problems [7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]. In some cases, iterative methods have been quite successful, in comparison to direct solvers. However, there are other situations where the performance of iterative methods is quite poor [16]. This generally seems to occur for problems where the elements have a small ($\ll 1$) aspect ratio [10, 16], which produces matrices having a large condition number. Although it is advisable, from a finite element error standpoint, to produce meshes which result in well shaped elements, in practice, this may not always be possible. Also, many problems which were traditionally modelled using two-dimensional plate and shell elements, are now being modelled using quasi-three-dimensional elements. This is particularly useful for analysing laminated composite materials [18, 16]. In this case, any practical finite element mesh will be composed of small aspect ratio elements.

The objective of this article is to develop robust iterative methods for three-dimensional stress analysis, which can be applied to problems with unstructured, tetrahedral meshes. We will attempt to ascertain which iterative techniques can reliably outperform a direct solver. As a model problem, we will consider solution of the linear elasticity equations for three-dimensional objects using quadratic basis functions defined on tetrahedra. This gives rise to a symmetric positive definite problem for the displacements. Preconditioned conjugate gradient (PCG) methods will be used. Although multi-grid methods have also been suggested for stress analysis [14, 17], application of multi-grid methods to unstructured grid problems is a non-trivial task. Consequently, in this work, attention will be restricted to PCG methods.

The convergence of a PCG method is strongly influenced by the type of preconditioning. Level based [4, 5, 19, 20] and drop tolerance based [21, 22, 23, 20, 24] incomplete LU factorization (ILU) preconditioning will be used. In addition, the ILU methods will be combined with two level hierarchical basis functions [25, 26, 27, 28, 29, 30, 31, 32]. Storage limitations permitting, these PCG methods will be compared with direct solution times.

The above methods have been tested on a variety of test problems. The selected problems range from simple model problems (a cube with varying aspect ratio) to fairly realistic complex three-dimensional objects. These problems have from 1,000 to 70,000 degrees of freedom. All results are obtained on a workstation.

2 Formulation

The three-dimensional linear elasticity problem for an isotropic material is given (in terms of the displacement vector $\mathbf{u}$) by

$$\mu \nabla^2 \mathbf{u} + (\lambda + \mu) \nabla \nabla \cdot \mathbf{u} \;=\; 0 \tag{1}$$

with the boundary conditions $\mathbf{u}$ specified, surface tractions specified, or natural boundary conditions. In the above, $\mathbf{u}$ is the vector of displacements in the three co-ordinate directions, and

$$\begin{aligned} \mu &= \frac{E}{2(1+\nu)} \\ \lambda &= \frac{E\nu}{(1+\nu)(1-2\nu)} \end{aligned}$$

where E is Young's modulus, and ν is Poisson's ratio.

Using the usual quadratic Lagrange polynomial basis functions defined on tetrahedra (the ten node tetrahedra), a standard finite element procedure applied to equation (1) results in the symmetric positive definite stiffness matrix A

$$Au \;=\; b \tag{2}$$

where u is the vector of the nodal displacements, and b is the right hand side vector. Within each tetrahedra, let ϕ_i be the basis function at node i (vertex or midside), then

$$\mathbf{u} \;=\; \sum_{i=1}^{i=10} \mathbf{u}_i \phi_i \tag{3}$$

where the ϕ_i are the usual nodal basis functions.

Alternatively, we can use another set of basis functions to derive an equivalent stiffness matrix. Let $\hat{\phi}_i$ be the new basis functions defined on each tetrahedra. Let $i = \{1,2,3,4\}$ be the vertex nodes of the tetrahedra, and $i = \{5,6,7,8,9,10\}$ be the midside nodes. Then let $\hat{\phi}_i, i = 1,..,4$ be the trilinear functions

$$\begin{aligned} \hat{\phi}_i &= 1 \text{ at node } i \\ &= 0 \text{ at nodes } j \;, j \neq i \\ &\qquad (i,j) = 1,...,4 \end{aligned}$$

Let $\hat{\phi}_i, i = 5,..,10$, be the quadratic functions where

$$\begin{aligned} \hat{\phi}_i &= 1 \text{ at node } i \\ &= 0 \text{ at nodes } j \;, j \neq i \\ &\qquad i = 5,...,10 \;;\; j = 1,...,10 \end{aligned}$$

In other words, $\hat{\phi}_i, i = 1, .., 4$ are the usual linear basis, and $\hat{\phi}_i, i = 5, ..., 10$ are the usual quadratic basis, i.e.

$$\phi_i \quad = \quad \hat{\phi}_i \qquad i = 5, ..., 10$$

Note that $\hat{\phi}_i, i = 1, .., 4$ do not vanish at neighbouring midside nodes. This basis is known as a hierarchical basis [25, 33, 26, 10, 29, 12, 14]. The analogue of equation (3) is

$$\begin{aligned} \mathbf{u} \quad = \quad & \sum \hat{\mathbf{u}}_i \hat{\phi}_i \\ & \mathbf{u}_i = \hat{\mathbf{u}}_i \qquad i = 1, ..., 4 \\ & \mathbf{u}_i = \hat{\mathbf{u}}_i + \frac{\hat{\mathbf{u}}_{v1} + \hat{\mathbf{u}}_{v2}}{2} \qquad i = 5, ..., 10 \end{aligned} \tag{4}$$

where $v1, v2$ are the neighbouring vertex nodes to a midside node.

Since the usual quadratic basis can be written as linear combination of the hierarchical basis, it is possible to simply transform equation (2) into a new stiffness matrix [34]. Alternatively, the stiffness matrix in the new basis may be constructed directly.

If the vector of unknowns $\hat{u}$ in the hierarchical basis is partitioned into vertex unknowns $\hat{u}_v$ and midside unknowns $\hat{u}_m$, then the stiffness matrix in the hierarchical basis A^H can be written

$$A^H = \begin{pmatrix} A_{mm} & A_{mv} \\ A_{vm} & A_{vv} \end{pmatrix} \begin{pmatrix} \hat{u}_m \\ \hat{u}_v \end{pmatrix} = \begin{pmatrix} \hat{b}_m \\ \hat{b}_v \end{pmatrix} \tag{5}$$

where we have ordered the midside unknowns first, and the vertex unknowns last. The block element A_{vv} in equation (5) is simply the usual stiffness matrix for a linear basis function solution to equation (1).

It is possible to define various other new basis functions [34, 15, 27, 31]. Depending on the type of preconditioning used, some of these other techniques may be superior to a hierarchical basis if many hierarchical levels are used and a very simple preconditioning method is employed [31]. However, for a suitable preconditioning method, is is possible to obtain good results for a hierarchical basis in three-dimensions, even for a large number of hierarchical bases [27, 30].

In this work, since we are dealing with a quadratic basis for an unstructured tetrahedral mesh, we will restrict attention to a two level p-version of the hierarchical basis [12]. It would appear to be a non-trivial task to use a large number of h-version hierarchical basis functions for an unstructured mesh, with possible discontinuous material properties.

It is of course possible to use more p-levels in the basis, i.e. linear, quadratic, cubic, and so on. This idea has been considered in [14], but we will not pursue this idea in this work. The idea of using a two-level method is similar to the technique used in [9], in the context of a multi-grid-like method.

3 Preconditioning Methods

ILU preconditioning is a widely used, robust method. However, there are many different issues which arise in ILU preconditioning. For example, an ILU can be based on the graph of the matrix (levels of fill) [5, 19], a drop tolerance [21, 22], or both [20]. The ordering of the unknowns can also strongly effect the convergence rate [35, 19]. As well, stress analysis problems typically do not result in M-matrices. Consequently, an incomplete LU factorization (ILU) of a symmetric positive definite matrix may produce negative pivots [36]. If a two level hierarchical basis is used, there are also several ways to approximate the block ILU factorization.

In the following, the notation $ILU(level, \epsilon)$ will be used to denote an incomplete factorization, with level $level$, and drop tolerance ϵ. For example

$$ILU(\infty, .01)$$

would refer to a pure drop tolerance ILU (drop tolerance = .01), while

$$ILU(3, 0.0)$$

would refer to a pure level based ($level = 3$) ILU. For a given ordering of the unknowns, there is a standard definition of a level based ILU [19]. On the other hand, there are various ways to define a drop tolerance ILU [21, 22, 20]. For a given pivot sequence, let $A^{(k)}$ be the submatrix in the incomplete factorization which remains after eliminating the first $k-1$ variables (columns $1, .., k-1$ have been eliminated). Note that

$$\begin{aligned} A^{(1)} &= A \\ \{A^{(k)}\}_{ij} &= a_{ij}^{(k)} \end{aligned}$$

In this work, if a drop tolerance ILU is selected, then if

$$\begin{aligned} |a_{ij}^{(k)}| &< \epsilon\, a_{ii}^{(k)} \\ & \quad i \geq k \;\; ; \;\; j > i \end{aligned} \tag{6}$$

then the element $a_{ij}^{(k)}$ is dropped from the ILU, otherwise it is kept. Note that since A is symmetric, we need only consider the upper triangular factor. We have experimented with various criteria, and, in accordance with previous work [21, 23], we have found that the quality of the ILU seems to be fairly insensitive to the precise form of the drop tolerance, but it is, of course, sensitive to drop parameter ϵ. Criteria (6) happens to be easy to implement efficiently in our software. The precise choice of the best drop tolerance criterion is still an open question. More discussion of this may be found in [21, 22, 23, 20].

As mentioned previously, an incomplete factorization of a symmetric positive definite matrix which is not an M-matrix, may result in a negative diagonal, i.e. $a_{kk}^{(k)} < 0$. This

problem is well known, but in the past was not regarded as too serious. In many codes, if a negative pivot is encountered, the pivot is simply replaced by the a suitably chosen positive number, and the factorization proceeds [22, 36]. In our experience, this approach works well in many fluid dynamics applications, but is very poor for stress analysis problems, possibly because stiffness matrices may have off-diagonals which are several orders of magnitude larger (in absolute value) than the diagonals.

Another possibility is to add suitable multiples of the absolute value of any dropped fill terms (for either a drop tolerance or level of fill ILU) to the diagonal of the ILU. It can be shown that the resulting ILU matrix is positive definite [21]. However, the addition of the dropped terms to the diagonal may produce a poor ILU, and can result in slow convergence. It is clear that this approach overestimates the amount that must be added to the diagonal in order to preserve positive definiteness. To see this, consider an ILU of a symmetric positive definite, diagonally dominant M-matrix. It is well known that a ILU of such a matrix will be positive definite, without addition of any terms to the diagonal. However, the approach described above [21, 22, 23] would unnecessarily increase the size of the diagonal of the ILU, and hence slow convergence. Nevertheless, this method is completely reliable, in that a successful ILU factorization is always guaranteed. In the following, we will refer to this approach as *J&M Add*. Assuming that the original diagonals of the stiffness matrix have been scaled to one, then the following implementation of the method [23] is used in this work. If $a_{ij}^{(k)}$ is a term which is dropped from the ILU (for either level or drop tolerance), then,

$$\begin{aligned} a_{ii}^{(k)} &:= a_{ii}^{(k)} + |a_{ij}^{(k)}| \\ a_{jj}^{(k)} &:= a_{jj}^{(k)} + |a_{ij}^{(k)}| \\ & \quad i \geq k \; ; \; j > i \end{aligned} \tag{7}$$

In equation (7) it is assumed that only the upper triangular part of A is being processed.

The negative pivot problem can also be avoided in the following way. The diagonal $a_{kk}^{(k)}$ is monitored. If

$$\begin{aligned} a_{kk}^{(k)} &< \gamma \; R_k \\ & \quad R_k = \max_{m \geq k} \left(|a_{km}^{(k)}| \right) \end{aligned} \tag{8}$$

Then the ILU factorization is aborted, and a new ILU is attempted on the perturbed system[7]

$$\left(a_{ii}^{(1)} \right)' = (1 + \alpha) \; a_{ii}^{(1)} \quad ; \quad \forall i \tag{9}$$

If equation (8) is violated again, then α is increased, and the ILU is repeated, until equation (8) is satisfied for all k. Typically, we use

$$\gamma = 0.0$$

$$\begin{aligned} \alpha &= (p-1)\ \eta \\ \eta &= 10^{-3} \end{aligned} \tag{10}$$

where p is the number of the current attempt to ILU factor the matrix.

It is possible to simply precondition the stiffness matrix A resulting from the standard Lagrange quadratic basis, using an ILU factorization. In other words, we incompletely factor A in equation (2) as

$$\mathbf{P0} = LL^t \ \simeq A \tag{11}$$

Henceforth, this preconditioning will be referred to as **P0**.

If the hierarchical basis is used, then we will consider two approximate block factorizations of the hierarchical stiffness matrix (5). The simplest preconditioning [25, 10], which will referred to as **P1**, is

$$\mathbf{P1} = \begin{pmatrix} L_m L_m^t & 0 \\ 0 & L_v L_v^t \end{pmatrix} \simeq \begin{pmatrix} A_{mm} & A_{mv} \\ A_{vm} & A_{vv} \end{pmatrix} \tag{12}$$

where L_m, L_v are (possibly) approximate factors of the block diagonals

$$\begin{aligned} L_m L_m^t &\simeq A_{mm} \\ L_v L_v^t &\simeq A_{vv} \end{aligned} \tag{13}$$

A more accurate preconditioning, which will be referred to as **P2** attempts to take into account some of the vertex-midside coupling terms A_{mv}, A_{vm} [25, 29].

$$\mathbf{P2} = \begin{pmatrix} L_m L_m^t & 0 \\ A_{vm} & L_v L_v^t \end{pmatrix} \begin{pmatrix} I & \left(L_m L_m^t\right)^{-1} A_{mv} \\ 0 & I \end{pmatrix} \simeq \begin{pmatrix} A_{mm} & A_{mv} \\ A_{vm} & A_{vv} \end{pmatrix} \tag{14}$$

If h is the mesh size parameter, then the spectral condition number of A_{vv} is $O(h^{-2})$, since A_{vv} is the usual stiffness matrix for a linear basis function discretization of equation (1). It can be shown that the condition number of A_{mm} is independent of h [25, 10]. An intuitive explanation for this is simply that A_{mm} represents the stiffness matrix of a discretization of equation (1) with all vertex (v) nodes fixed [9].

Since A_{mm} has condition number independent of h, then, at least in theory, a very simple preconditioning of A_{mm} can be used. Various authors have proposed a **P1** preconditioning of the form [25, 10]

$$\begin{aligned} L_m L_m^t &= diag\ (A_{mm}) \\ L_v L_v^t &= ILU(\infty, 0.0) \end{aligned} \tag{15}$$

(i.e. an exact factorization of A_{vv}, and diagonal scaling of A_{mm}). Consider a sequence of discretizations of equation (1), with different mesh sizes h, then if x_h is a vector in the finite element solution space for a given mesh size h, V_h, then, for **P1** as in equation (15)

$$c_1(x_h, \mathbf{P1}_h x_h) \leq (x_h, A_h^H x_h) \leq c_2(x_h, \mathbf{P1}_h x_h)\ ; \quad \forall x_h \in V_h \quad , \quad \forall h \tag{16}$$

where c_1, c_2 are independent of h [10]. In other words, **P1** and A^H are spectrally equivalent, which implies that the number of iterations required to solve the stiffness matrix using a PCG method is independent of mesh size. It is also known that if $L_v L_v^t$ is an exact factorization of A_{vv} and $L_m L_m^t$ is any approximate factorization which is spectrally equivalent to A_{mm}, then **P2** and A^H are also spectrally equivalent [29].

Of course, this assumes that $L_v L_v^t$ is an exact factorization of A_{vv}. This matrix is about 1/8 the size of A for typical three-dimensional triangulations, and is much more sparse than A (linear basis functions as opposed to quadratic). Hence the cost of exactly factoring A_{vv} is small compared to the cost of exactly factoring A. However, it is possible to use more hierarchical levels, and to use a recursive form of the **P2** preconditioner, to develop a preconditioner which is spectrally equivalent to A, but does not require an exact factorization, except on the coarsest level [27, 30]. Note that in this case the preconditioner [27, 30] is not simply based on diagonal scaling of high levels, but uses an accurate approximation to the Schur complement at each level.

However, use of more than two levels requires definition of "coarse grid" linear basis functions, which is a non-trivial task for unstructured tetrahedral meshes. Consequently, we will not consider more than two levels in this work. As mentioned above, the computational work for an exact factorization of A_{vv} is small compared to a direct solve of the original problem, and if this is the price to be paid for a robust iterative method, so be it.

Unfortunately, the situation is not as rosy as it might appear. Although the constants c_1, c_2 are independent of h for suitable L_m, L_v, these constants are functions of Poisson's ratio ν and the element aspect ratio [10]. In fact, if the **P1** preconditioner with L_m defined as in equation (15) is used, the condition number of the preconditioned system (c_2/c_1) may be very large for bad ($\ll 1$) element aspect ratios [10]. Consequently, it will often be necessary to use a fairly accurate ILU factorization of A_{mm}, even though A_{mm} has condition number independent of h. Note that in reference [9], use of an exact factorization of A_{mm} was advocated.

As mentioned previously, ordering of the unknowns can have a significant effect on the convergence of PCG methods [19]. In this work, a Reverse Cuthill McGee (RCM) ordering will be used [37] for an ILU factored submatrix. The RCM ordering is based on the nodal graph of the finite element mesh. This nodal ordering is then converted to a degree of freedom ordering by ordering all the displacements at each node consecutively. In other words, this is a block RCM ordering, with all displacements at a single node contained in each block. If an exact factorization of a submatrix is used, then minimum degree ordering [37] is used to minimize work and fill.

Note that these preconditionings all require some operation like

$$A_{pp}^{-1} x \tag{17}$$

which is approximated by

$$\simeq \quad (L_p L_p^t)^{-1} x \ , \quad p \in \{m, v\} \tag{18}$$

where x is some vector. Instead of simply approximating the inverse in equation (18) by the ILU $L_p L_p^t$, it is possible to determine the solution to equation (17) to any degree of precision by using a PCG method, with $L_p L_p^t$ as a preconditioner to solve equation (17). This gives rise to nested levels of a PCG iteration. Each outer PCG iteration, requires several inner PCG iterations to solve equation (17).

4 Test Problems

The convergence tolerance for all tests was

$$\frac{\|r^k\|_2}{\|r^0\|_2} \quad < \quad 10^{-6} \tag{19}$$

where $\|r^0\|_2$ is the initial l_2 norm of the residual, and $\|r^k\|_2$ is the norm of the residual after k iterations. After symmetrically scaling the matrix (and consequently the solution vector and right hand side), the zero vector was used as the initial guess. Note that in some cases we will be comparing results for an ILU of the original stiffness matrix A and the hierarchical basis A^H, in which case the residual, and hence the convergence criteria, are not precisely the same. However, we have computed the difference between the usual residual and the hierarchical basis residual (at convergence) in all our runs. The maximum difference in the norms of these two residuals (at convergence) was about 20%, indicating that if a nodal residual was used in the convergence test for the hierarchical basis, or *vice versa*, the number of iterations would change by one or two, which would not have any appreciable impact on our results.

Where storage limitations permit, we have also solved the test problems using the Yale Sparse Matrix Package (YSMP), which is a direct solver [38]. Minimum degree ordering was used for the direct solve.

A summary of test problem data for each problem is given in Table 1. For general tetrahedral meshes, it is useful to have a well defined measure of element aspect ratio. A common measure [39] is three times the ratio of the radius of the circumsphere to the radius of the inscribed sphere for each tetrahedra. Table 1 shows the element aspect ratio for each test problem. The test problems are described in detail below.

4.1 Cube Problem

This problem consists of a cube-like solid, with physical dimensions $l \times l \times lz$. This solid is first gridded using an $n \times n \times n$ rectangular grid, which results in $(n-1) \times (n-1) \times (n-1)$ bricks. Each brick is then further subdivided into six tetrahedra.

The solid is constrained at the four bottom corners, and has a specified deflection at one top corner. By varying the thickness lz, the effect of varying element aspect ratio can be observed. This problem was solved using a $4 \times 4 \times 4$ and $10 \times 10 \times 10$ mesh.

Table 1: Summary of Test Problem Data

Problem Descriptions	Degrees of Freedom	Nonzeros in U(A)	Element Aspect Ratios Minimum	Average	Material Properties ν
$4 \times 4 \times 4$ $(l/lz = 1)$	1029	34377	0.717	0.717	0.400
Cube $(l/lz = 10)$			0.190	0.190	
$(l/lz = 100)$			0.020	0.020	
$10 \times 10 \times 10$ $(l/lz = 1)$	20577	816081	0.717	0.717	0.400
Cube $(l/lz = 10)$			0.190	0.190	
$(l/lz = 100)$			0.020	0.020	
Balljoint	21411	847719	1.6×10^{-4}	0.779	0.266
Block Containing Prism	19479	757050	1.2×10^{-4}	0.784	Block 0.300 Prism 0.450
Bushing	49896	2001024	0.451	0.595	0.346
Multi-material Bracket	50637	1998375	1.0×10^{-4}	0.783	Supports 0.450 Base 0.200
Corroded Pipe	73191	2646231	0.184	0.356	0.301

4.2 Balljoint Problem

The balljoint problem is a quarter model of a balljoint. The ball has a radius of 0.01 m and the shaft extending from it has a radius of 0.008 m and a length of 0.016 m from the center of the ball. The back of the ball has been cut off perpendicular to the axis of the shaft at a distance of 0.006 m from the center of the ball. A depiction of the surface mesh is shown in Figure 1. The tetrahedral mesh was generated by the I-DEAS VI: Mesh Generation Package [40].

For boundary conditions, the end of the shaft has been deflected along and across the direction of the shaft axis. The ball surface was fixed in a slanted band near where the shaft connects to the ball.

4.3 Block Containing Prism Problem

This problem consists of a block of one material with a prism of another material through the length of the block. The block has a width of 0.07 m, a height of 0.1 m and a length of 0.4 m. The cross section of the triangular prism is defined by the three points (0.02, 0.05), (0.05, 0.02) and (0.05, 0.08). The tetrahedral meshes were generated by the I-DEAS VI: Mesh Generation Package [40]. Refer to Figures 2 & 3 to see the surface mesh of the two separated components.

Boundary conditions are created by first fixing the position of all the nodes on one of the

exposed ends of the prism and then applying a uniform, skewed force over the entire surface area of the other end of the block.

4.4 Bushing Problem

The bushing problem is a model of thick washer made from a soft material. The bushing has an outer radius of 0.04 m, an inner radius of 0.01 m, and a thickness of 0.01 m. A view of the surface mesh for the bushing can be found in Figure 4.

The boundary conditions are designed to simulate the use of the bushing to protect a block of material from direct contact with a C–clamp being applied right on the edge of the block. Thus the bushing is being bent over the lip of the block by a skew load. The position of all nodes on the bottom surface of the bushing to the right of 0.03 m are fixed and every surface triangle on the top of the bushing is subjected to a skew load.

4.5 Multi-material Bracket Problem

The bracket problem is another multi-material test problem. The base and face plates are made of the same material and the supports are made of a different material. The base and face plates are 0.03 m deep/thick, the two triangular plate supports are also 0.03 m thick, and the supports are 0.1 m apart. In total the bracket/brace is 0.13 m deep, 0.13 m high and 0.16 m wide. Figure 5 depicts the surface mesh for the base and face plates combined and Figure 6 depicts the surface mesh for the supports. The tetrahedral mesh was generated by the I-DEAS VI: Mesh Generation Package [40].

For boundary conditions the base of the plate is fixed, and the face plate nodes are displaced perpendicular to the plate's surface as a function of displacement from the base plate (i.e. $0.001 \times z$ coordinate). This effectively slants the face plate as if heavily loaded.

4.6 Corroded Pipe Problem

The mesh for the pipe problem was contributed by B. Chouchaoui [41] who is studying the impact of corrosion on pipe strengths. Only the mesh representing half the arc of the pipe was used. The boundary conditions were a simplified version of the original experimental data. The pipe segment is 350.0 mm long with an inner and outer radius of 155.46 and 161.86 mm respectively. The corrosion pit is centered on the pipe segment and is marked by a finer grid. For our analysis, each hexahedral element was divided into six tetrahedral elements. Figure 7 shows the surface mesh from the hexahedral discretization.

For boundary conditions, the pipe experiences tension along its length caused by fixing the nodes of one end and applying a normal force to all surface triangles on the other end. To avoid rotation around the pipe's axis, the positions of the nodes along one lengthwise edge are fixed so no motion is allowed out of the cut plane, while allowing displacements within the cut plane.

5 Results

There are a very large number of possible combinations of preconditionings, as discussed in Section 3. In the following, we will give some detailed results for a representative sample of tests. Space does not permit an exhaustive catalogue of all the results. We have carried out many more tests than will be reported in detail, and we will indicate qualitatively the trends in these test cases as well.

In the following tables the notation $ILU(..,..)$ will refer to an ILU factorization of the original nodal basis, while $ILU_v(..,..)$, $ILU_m(..,..)$ will refer to the ILU of A_{vv}, and A_{mm} , as in equation (5).

Unless otherwise indicated, the method of equations (8, 9) will be used to ensure a positive definite ILU factor. The parameters are those in equations (10). If a negative pivot is encountered on the fifth attempt at ILU factoring ($p = 5$ in equation (10)), the ILU is aborted.

All results are given in terms of total CPU time and number of iterations. The total CPU time includes all the time required to symbolically and numerically ILU factor the matrix (these steps are combined when a drop tolerance ILU is used), and the iteration cost (forward and back solve, acceleration work). In addition, the total CPU time includes all the time required for repeated attempts to perform the ILU factorization (equations (8, 9)) to ensure positive pivots. The CPU time does not include the time required for construction of the stiffness matrix. CPU times are reported for a SUN 670/MP.

Most tests were done initially on the *Cube* problem because of the ease with which both the material properties of the *Cube* and the mesh size being used could be changed. The test results for the *Cube* will thus be presented first followed by the experimental results for problems with more complex geometries.

5.1 Cube Results

Some representative results for the *Cube* problem are given in Tables (2, 3, 4, 5). In both Tables (2, 3), the line *J & M Add* refers to the procedure used in [21, 22, 23] applied to the preconditioning method listed in the line above. Table 2 clearly shows that for **P0** preconditioning (equation (11)), the number of iterations increases drastically as the *Cube*'s aspect ratio decreases. Similar behaviour has been noted by others [10, 16].

Tables 3 and 4 indicate that **P2** preconditioning (equation(14)) requires about one half (on average) the iterations as a similar **P1** preconditioning. However, since each **P2** preconditioning iteration requires more than twice the work of a **P1** preconditioning iteration, the **P2** method always requires more total CPU time than an equivalent **P1** technique. Consequently, there does not appear to be any advantage to using a **P2** preconditioning, even for problems with poor aspect ratios ($l/lz \gg 1$).

Table 3 shows that for problems with aspect ratios near unity, a diagonal scaling preconditioning $ILU_m = diag(A_{mm})$ is quite effective. However, the performance of this method

Table 2: P0 Preconditioning for Cube 4x4x4

	CPU seconds (iterations)		
Preconditioning	$l/lz = 1.0$	$l/lz = 10$	$l/lz = 100$
$ILU(1, 0.0)$	8.48 (18)	24.9 (88)	124.26 (649)
J & M add	16.23 (52)	44.64 (217)	*
$ILU(2, 0.0)$	13.17 (14)	19.8 (40)	112.2 (469)
J & M Add	18.34 (27)	33.9 (101)	115.7 (485)
$ILU(\infty, 10^{-5})$	16.51 (3)	15.98 (6)	63.75 (271)
J & M Add	20.59 (8)	29.1 (35)	76.5 (312)

$*$*Did not converge in 1000 iterations*

degrades severely as l/lz increases. Note that the very accurate ILU factorizations of A_{mm}, $ILU_m(\infty, 0.0), ILU_m(\infty, 10^{-5})$ are more robust (fewer iterations) and require less CPU time than the less accurate ILU factorizations of A_{mm} as l/lz increases.

Tables (2, 3) also indicate that the *J & M Add* technique [21] becomes slower as the aspect ratio deteriorates (becomes smaller) for relatively inaccurate factorizations. For inaccurate ILU factorizations, the discarded fill terms are fairly large, so that the amount added to the diagonal is also quite large. This tends to cause an iterative method which uses *J & M Add* to slow down. However, as the ILU becomes more accurate (higher level or smaller drop tolerance), the amount added to the diagonal is smaller, and hence this technique becomes more competitive.

Representative computations for the $10 \times 10 \times 10$ *Cube* problem are given in Table 5. The level based ILU **P0** methods are competitive with **P1** preconditionings for l/lz small, but are poor for $l/lz \gg 1$. As for the smaller version of the *Cube* problem, the preconditioners $ILU_m = diag(A_{mm}), ILU_m(0, 0.0)$ are poor as the aspect ratio becomes more extreme ($\ll 1$). More accurate factorizations of A_{mm} are required as l/lz increases. Table 5 also shows an interesting phenomenon. The very small drop tolerance ILU factorizations of A_{mm} require more CPU time for cubes having aspect ratio near unity compared to problems with poor aspect ratios ($l/lz \gg 1$). This is because for a given (small) drop tolerance, more fill is generated for $l/lz = 1$ than for $l/lz \gg 1$. For anisotropic M-matrix problems, this effect has been noted previously [20], however, it was noted in [20] that this effect is strongly dependent on then ordering of the unknowns. It would appear that for problems with small ($\ll 1$) aspect ratios, a small drop tolerance ILU is more effective than for problems with aspect ratios ~ 1.

Table 6 repeats many of the tests given in Table 5, except that the *J & M Add* method was used to prevent negative pivots (instead of adding an *a priori* number to the diagonal (equations (8, 9)). Generally, the *J & M Add* method was slower than the method of equations (8, 9), even if the ILU was attempted several times before no negative pivots were

Table 3: P1 Preconditioning for Cube 4x4x4

	CPU seconds (iterations)		
Preconditioning	$l/lz = 1.0$	$l/lz = 10$	$l/lz = 100$
$ILU_v(\infty, 0.0)$ $ILU_m = diag(A_{mm})$	5.78 (58)	38.62 (431)	*
$ILU_v(\infty, 0.0)$ $ILU_m(1, 0.0)$ *J & M Add*	6.9 (38) 7.6 (40)	9.69 (54) 15.2 (99)	47.24 (346) 114.4 (870)
$ILU_v(\infty, 0.0)$ $ILU_m(2, 0.0)$ *J & M Add*	8.6 (36) 9.63 (38)	9.71 (43) 13.2 (62)	52.8 (329) 77.2 (493)
$ILU_v(\infty, 0.0)$ $ILU_m(\infty, 0.0)$	17.22 (34)	16.25 (37)	29.19 (105)
$ILU_v(\infty, 0.0)$ $ILU_m(\infty, 10^{-4})$ *J & M Add*	12.59 (35) 15.6 (37)	11.199 (40) 14.7 (52)	40.72 (285) 58.0 (421)
$ILU_v(\infty, 0.0)$ $ILU_m(\infty, 10^{-5})$ *J & M Add*	17.22 (34) 20.7 (36)	14.96 (38) 18.77 (43)	27.44 (152) 43.9 (262)

$*$*Failure to converge after 1000 iterations.*

Table 4: P2 Preconditioning for Cube 4x4x4

	CPU seconds (iterations)		
Preconditioning	$l/lz = 1.0$	$l/lz = 10$	l/lz = 100
$ILU_v(\infty, 0.0)$ $ILU_m(1, 0.0)$	7.9 (21)	15.02 (46)	71 (251)
$ILU_v(\infty, 0.0)$ $ILU_m(2, 0.0)$	9.7 (20)	10.95 (24)	72.8 (211)
$ILU_v(\infty, 0.0)$ $ILU_m(\infty, 0.0)$	16.6 (17)	17.46 (19)	32.55 (54)
$ILU_v(\infty, 0.0)$ $ILU_m(\infty, 10^{-4})$	14.3 (19)	12.38 (21)	56.1 (190)
$ILU_v(\infty, 0.0)$ $ILU_m(\infty, 10^{-5})$	19.27 (18)	16.5 (20)	49.8 (135)

Table 5: P0, P1 Preconditioning for Cube 10x10x10

	CPU seconds (iterations)		
Preconditioning	$l/lz = 1.0$	$l/lz = 10$	l/lz = 100
P0			
$ILU(0, 0.0)$	557 (153)	**	**
$ILU(1, 0.0)$	513 (67)	2033 (394)	***
$ILU(2, 0.0)$	964 (43)	1954 (187)	*
$ILU(\infty, 10^{-1})$	540 (154)	**	**
$ILU(\infty, 10^{-2})$	391 (84)	**	**
$ILU(\infty, 10^{-3})$	521 (44)	1869 (344)	*
$ILU(\infty, 10^{-4})$	539 (154)	1856 (127)	*
$ILU(\infty, 10^{-5})$	539 (154)	3756 (112)	*
P1			
$ILU_v(\infty, 0.0)$ $ILU_m(\infty, 0.0)$	***	***	***
$ILU_v(\infty, 0.0)$ $ILU_m = diag(A_{mm})$	421 (58)	1172 (323)	***
$ILU_v(\infty, 0.0)$ $ILU_m(0, 0.0)$	662 (42)	825 (36)	**
$ILU_v(\infty, 0.0)$ $ILU_m(1, 0.0)$	457 (39)	512 (49)	1316 (248)
$ILU_v(\infty, 0.0)$ $ILU_m(2, 0.0)$	584 (39)	563 (36)	1632 (244)
$ILU_v(\infty, 0.0)$ $ILU_m(\infty, 10^{-1})$	298 (41)	**	**
$ILU_v(\infty, 0.0)$ $ILU_m(\infty, 10^{-2})$	312 (39)	**	**
$ILU_v(\infty, 0.0)$ $ILU_m(\infty, 10^{-3})$	427 (39)	437 (48)	1404 (315)
$ILU_v(\infty, 0.0)$ $ILU_m(\infty, 10^{-4})$	847 (38)	633 (35)	1378 (240)
$ILU_v(\infty, 0.0)$ $ILU_m(\infty, 10^{-5})$	2182 (38)	1243 (35)	1676 (240)
$ILU_v(\infty, 0.0)$ $ILU_m(\infty, 10^{-6})$	***	2650 (34)	1311 (92)

$*$*Failure to converge after 1000 iterations.*

$**$*Failed due to negative pivot in ILU (five attempts).*

$***$*CPU time limit (4000 seconds) exceeded.*

Table 6: P1 Preconditioning for Cube 10x10x10 with J & M Add

	CPU seconds (iterations)		
Preconditioning	$l/lz = 1.0$	$l/lz = 10$	l/lz = 100
$ILU_v(\infty, 0.0)$ $ILU_m(0, 0.0)$	406 (43)	867 (179)	*
$ILU_v(\infty, 0.0)$ $ILU_m(1, 0.0)$	475 (40)	617 (79)	*
$ILU_v(\infty, 0.0)$ $ILU_m(2, 0.0)$	635 (39)	697 (52)	3476 (626)
$ILU_v(\infty, 0.0)$ $ILU_m(\infty, 10^{-1})$	316 (43)	696 (146)	*
$ILU_v(\infty, 0.0)$ $ILU_m(\infty, 10^{-2})$	333 (42)	567 (112)	*
$ILU_v(\infty, 0.0)$ $ILU_m(\infty, 10^{-3})$	440 (39)	524 (68)	3549 (924)
$ILU_v(\infty, 0.0)$ $ILU_m(\infty, 10^{-4})$	1071 (39)	774 (40)	2168 (441)
$ILU_v(\infty, 0.0)$ $ILU_m(\infty, 10^{-5})$	3331 (39)	1763 (35)	1618 (214)
$ILU_v(\infty, 0.0)$ $ILU_m(\infty, 10^{-6})$	***	3734 (35)	1796 (118)

**Failure to converge after 1000 iterations.*
****CPU time limit (4000 seconds) exceeded.*

encountered.

In summary, it seems that a robust method for various aspect ratios uses a **P1** preconditioning with

$$ILU_m(1, 0.0)\ ,\ ILU_m(2, 0.0)$$
$$ILU_m(\infty, 10^{-3})\ ,\ ILU_m(\infty, 10^{-4})$$

preconditionings for A_{mm}, and an exact factorization of A_{vv}. If it is known that the aspect ratio is near unity, then **P1** with $ILU_m = diag(A_{mm})$ is effective, as are **P0** methods.

Further tests were carried out to determine the effect of having a nonconstant Young's modulus (E) within the *Cube*. A problem with $l/lz = 10$, and **P1** preconditioning with $ILU_m(2, 0.0)$, $ILU_v(\infty, 0.0)$ was used. Even if E had large jump discontinuities of several orders of magnitude, there was very little effect on convergence.

A similar test was carried out to determine the effect of varying Poisson's ratio. ν was kept constant in the *Cube*, but different values of ν were used in different tests. The performance

Table 7: P0 Preconditioning for Complex Geometry Problems

	CPU seconds (iterations)				
Preconditioning	Balljoint	Block & Prism	Bushing	Multi-Material Bracket	Corroded Pipe
$ILU(\infty, 10^{-1})$	435 (109)	**	4135 (511)	**	**
J & M Add	830 (219)	896 (281)	7845 (994)	6686 (845)	*
$ILU(\infty, 10^{-2})$	300 (57)	294 (64)	2151 (222)	1979 (190)	**
J & M Add	790 (192)	747 (210)	7406 (853)	5575 (659)	*
$ILU(\infty, 10^{-3})$	448 (33)	395 (32)	2173 (114)	1478 (69)	11684 (789)
J & M Add	892 (120)	894 (146)	6782 (530)	4868 (375)	*
$ILU(\infty, 10^{-4})$	1283 (19)	1018 (16)	4162 (59)	2929 (32)	7178 (319)
J & M Add	1703 (64)	1510 (76)	8064 (281)	6225 (202)	*
$ILU(\infty, 10^{-5})$	4951 (9)	2535 (6)	19516 (48)	13328 (29)	10790 (311)
J & M Add	5299 (33)	3401 (33)	15941 (131)	11995 (92)	13567 (504)
$ILU(0, 0.0)$	**	**	3707 (451)	**	**
J & M Add	805 (220)	819 (253)	*	6818 (851)	*
$ILU(2, 0.0)$	489 (47)	411 (49)	2730 (193)	2077 (125)	**
J & M Add	856 (103)	822 (124)	6106 (482)	5479 (399)	*
$ILU(2, 0.0)$	1041 (28)	801 (27)	4175 (118)	3257 (68)	**
J & M Add	1583 (56)	1245 (61)	7062 (248)	6596 (215)	*

$*$*Failure to converge after 1000 iterations.*

$**$*Failed due to negative pivot in ILU (five attempts).*

of the iterative methods was highly dependent on ν as expected [10]. For example, the problem with $\nu = 0.49999$ required about 15 times the number of iterations as for the same problem with $\nu = 0.40$.

5.2 Results from more Geometrically Complex Problems

The test results from the problems having more complex geometries basically confirm the conclusions developed in the previous section. Tables (7, 8, 9) contain some representative results.

A comparison of Table 7 (**P0** preconditioning) and Table 8 (**P1** preconditioning with direct solve of A_{vv}) clearly shows that using a hierarchical basis provides generally better computation times. For one choice of preconditioning parameters the **P0** preconditioning method does give slightly better performance for the two smaller problems, *Balljoint* and *Block & Prism*. However, this performance gain was not repeated for any of the larger test problems. In particular, **P0** was an extremely poor technique for application to the *Corroded*

Pipe problem (poorer aspect ratio).

The best preconditioners seem to be based on using an exact factorization for A_{vv} and $ILU_m(\infty, 10^{-1})$, $ILU_m(\infty, 10^{-2})$, or $ILU_m(\infty, 10^{-3})$ for A_{mm}. Smaller drop tolerances were required for worse (smaller) aspect ratios (see Table 1), as would be expected [10]. As the complex geometry problems do not have as extreme aspect ratios as were used in the *Cube* problems, the recommended drop tolerance is larger. In general, the most robust method in these tests was found to be preconditioning by $ILU_m(\infty, 10^{-3})$ (and exact factorization of A_{vv}).

Unlike the results from the previous section, it is not clear that a level based drop tolerance would provide good results. The tests indicate that a level based preconditioning technique is almost always out-performed by the suggested drop tolerance technique (**P1**, $ILU_m(\infty, 10^{-3})$, $ILU_v(\infty, 0.0)$) and that the level based preconditioning provides less consistent results across the tested range of aspect ratios.

Given more information about the equations to be solved, it is possible to improve on these techniques. As discussed earlier, experimentation demonstrates two dominant factors that affect the rate of convergence of the iterative techniques. They are the average aspect ratio of the elements in the mesh and the value of ν. However, the value of ν seems to have a uniform affect on all iterative techniques leaving only the element aspect ratios to be considered.

For the test case meshes generated by the I-DEAS VI: Mesh Generation Package [40], it is noted that the average aspect ratio was always approximately 0.77. With these aspect ratios, the maximum rate of convergence using a preconditioner with an exact factorization of A_{vv} was attained by using a coarse drop tolerance of $ILU_m(\infty, 10^{-1})$ or, for fast results and minimum space requirements, using a $ILU_m = diag(A_{mm})$. However, even these rapid convergence rates can be improved upon.

Table 9 shows results from incomplete factorization of both A_{vv} and A_{mm}. Examination of these results shows that for average aspect ratios of 0.59 or more, very fast convergence can be attained by a **P1** preconditioning with $ILU_v(\infty, 10^{-1})$ or $ILU_v(\infty, 10^{-2})$, and $ILU_m(\infty, 10^{-3})$ or $ILU_m(\infty, 10^{-4})$. This technique was not effective for worse (smaller) aspect ratios.

Extensive testing of the *J & M Add* [21] method was also done. Table 8 shows the results for a **P1** preconditioning with direct solve of A_{vv} with and without using the *J & M Add* technique. This technique demonstrates several positive features. It usually only resulted in a small increase in convergence times for the complex geometry problems. For the three cases where attempts to avoid a negative pivot by adding an *a priori* number to the diagonal failed, the *J & M Add* technique did succeed. In several cases, faster convergence times were observed for specific tests of the *Corroded Pipe* and *Multi-Material Bracket* problems. However, the *J & M Add* method was never faster than the recommended **P1** technique (direct solve of A_{vv}, $ILU_v(\infty, 10^{-3})$) alone (Table 8).

In general, there is little reason to use the *J & M Add* method for problems with average aspect ratios of 0.59 and better or 0.02 (*Cube* with $l/lz = 100$) and worse. For problems

Table 8: P1 Preconditioning with Direct Solve of A_{vv} for Complex Geometry Problems

	CPU seconds (iterations)				
Preconditioning	Balljoint	Block & Prism	Bushing	Multi-Material Bracket	Corroded Pipe
$ILU_v(\infty, 0.0)$ $ILU_m(\infty, 10^{-1})$	371 (68)	354 (96)	1436 (57)	1355 (87)	**
J & M Add	390 (69)	356 (89)	1706 (70)	1237 (74)	4159 (321)
$ILU_v(\infty, 0.0)$ $ILU_m(\infty, 10^{-2})$	381 (66)	349 (86)	1469 (55)	1236 (71)	7063 (554)
J & M Add	402 (68)	380 (90)	1615 (57)	1255 (72)	2550 (168)
$ILU_v(\infty, 0.0)$ $ILU_m(\infty, 10^{-3})$	496 (66)	465 (85)	1726 (52)	1521 (71)	1889 (105)
J & M Add	542 (67)	497 (88)	1888 (53)	1565 (71)	2068 (111)
$ILU_v(\infty, 0.0)$ $ILU_m(\infty, 10^{-4})$	818 (65)	746 (79)	2656 (52)	2373 (70)	2103 (102)
J & M Add	1017 (66)	936 (86)	3400 (53)	2591 (70)	2268 (106)
$ILU_v(\infty, 0.0)$ $ILU_m(\infty, 10^{-5})$	1653 (65)	1383 (79)	5024 (52)	4018 (70)	2508 (100)
J & M Add	2320 (65)	2013 (86)	6762 (52)	5119 (71)	2714 (104)
$ILU_v(\infty, 0.0)$ $ILU_m(0, 0.0)$	466 (69)	384 (96)	2099 (58)	**	**
J & M Add	462 (69)	389 (89)	2468 (79)	1593 (75)	5257 (409)
$ILU_v(\infty, 0.0)$ $ILU_m(1, 0.0)$	542 (66)	444 (88)	2211 (54)	1733 (71)	6393 (438)
J & M Add	554 (67)	483 (87)	2271 (53)	1788 (71)	2261 (111)
$ILU_v(\infty, 0.0)$ $ILU_m(2, 0.0)$	740 (65)	597 (81)	2620 (53)	2170 (71)	2574 (112)
J & M Add	799 (66)	705 (86)	2833 (53)	2312 (70)	2678 (109)
$ILU_v(\infty, 0.0)$ $ILU_m = diag(A_{mm})$	403 (98)	422 (153)	1883 (121)	1374 (116)	6900 (731)

***Failed due to negative pivot in ILU (five attempts).*

Table 9: P1 Preconditioning with ILU Factoring of Both Domains for Complex Geometry Problems

	CPU seconds (iterations)			
Preconditioning	Block & Prism	Bushing	Multi-Material Bracket	Corroded Pipe
$ILU_v(\infty, 10^{-1}), ILU_m(\infty, 10^{-1})$	294 (114)	1423 (204)	**	**
$ILU_v(\infty, 10^{-2}), ILU_m(\infty, 10^{-1})$	263 (99)	1686 (222)	**	**
$ILU_v(\infty, 10^{-3}), ILU_m(\infty, 10^{-1})$	277 (96)	876 (107)	723 (87)	**
$ILU_v(\infty, 10^{-4}), ILU_m(\infty, 10^{-1})$	310 (96)	1039 (81)	775 (87)	**
$ILU_v(\infty, 10^{-1}), ILU_m(\infty, 10^{-2})$	315 (112)	1510 (202)	**	**
$ILU_v(\infty, 10^{-2}), ILU_m(\infty, 10^{-2})$	270 (92)	1845 (223)	**	**
$ILU_v(\infty, 10^{-3}), ILU_m(\infty, 10^{-2})$	282 (89)	929 (105)	698 (73)	**
$ILU_v(\infty, 10^{-4}), ILU_m(\infty, 10^{-2})$	305 (86)	1089 (78)	705 (71)	5669 (543)
$ILU_v(\infty, 10^{-1}), ILU_m(\infty, 10^{-3})$	445 (110)	2044 (193)	**	**
$ILU_v(\infty, 10^{-2}), ILU_m(\infty, 10^{-3})$	385 (90)	2695 (219)	**	**
$ILU_v(\infty, 10^{-3}), ILU_m(\infty, 10^{-3})$	398 (88)	1238 (101)	1224 (72)	**
$ILU_v(\infty, 10^{-4}), ILU_m(\infty, 10^{-3})$	414 (83)	1541 (75)	1021 (70)	3226 (247)
$ILU_v(\infty, 10^{-1}), ILU_m(\infty, 10^{-4})$	787 (109)	3692 (191)	**	**
$ILU_v(\infty, 10^{-2}), ILU_m(\infty, 10^{-4})$	703 (90)	5976 (219)	**	**
$ILU_v(\infty, 10^{-3}), ILU_m(\infty, 10^{-4})$	709 (87)	2398 (101)	2410 (72)	**
$ILU_v(\infty, 10^{-4}), ILU_m(\infty, 10^{-4})$	732 (85)	3282 (75)	1776 (71)	3675 (246)
$ILU_v(0, 0.0), ILU_m(0, 0.0)$	**	1404 (211)	**	**
$ILU_v(1, 0.0), ILU_m(0, 0.0)$	273 (101)	1094 (157)	**	**
$ILU_v(2, 0.0), ILU_m(0, 0.0)$	283 (98)	980 (126)	**	**
$ILU_v(0, 0.0), ILU_m(1, 0.0)$	**	1674 (201)	**	**
$ILU_v(1, 0.0), ILU_m(1, 0.0)$	335 (94)	1294 (147)	1112 (112)	**
$ILU_v(2, 0.0), ILU_m(1, 0.0)$	339 (90)	1128 (117)	1007 (86)	5560 (494)
$ILU_v(0, 0.0), ILU_m(2, 0.0)$	**	2380 (200)	**	**
$ILU_v(1, 0.0), ILU_m(2, 0.0)$	495 (92)	1864 (145)	1682 (111)	**
$ILU_v(2, 0.0), ILU_m(2, 0.0)$	505 (90)	1644 (116)	1592 (84)	4750 (330)

***Failed due to negative pivot in ILU (five attempts).*

inbetween these extremes (*Multi-Material Bracket & Corroded Pipe*) the results seem to indicate that the method may provide reasonable computation times with more stability than the recommended **P1** technique alone (Table 8). No firm conclusion could be drawn from just these test results.

When the *J & M Add* method was combined with a **P1** approach with ILU preconditioning for both A_{vv} and A_{mm}, convergence would generally take twice as long or more, depending on the average element aspect ratio. This was also found to be true when the *J & M Add* method was used with a **P0** preconditioning technique. There is some evidence the *J & M Add* method improved the robustness of the both the **P0** (Table 7) and **P1** technique, but only for non-recommended ILU's (i.e. $ILU(0, 0.0)$, $ILU(\infty, 10^{-1})$).

Some experiments were also carried out using a iterative method to carry out the operation in equation (17), for the midpoint block (A_{mm}), and an ILU preconditioner. This performed very poorly compared with simply using an ILU of A_{mm}, and hence no results will be given.

In summary, a **P1** preconditioning with an exact factorization of A_{vv} and a drop tolerance of $ILU_m(\infty, 10^{-2})$ or $ILU_m(\infty, 10^{-3})$ appears to be a generally robust method for a range of average element aspect ratios. A smaller drop tolerance should be favoured as the aspect ratio becomes worse. When the element aspect ratio is known to be above 0.58, even better results can be had by using incomplete factorization for both A_{vv} and A_{mm}. In this case having $ILU_v(\infty, 10^{-1})$ or $ILU_v(\infty, 10^{-2})$, and $ILU_m(\infty, 10^{-3})$ or $ILU_m(\infty, 10^{-4})$ is recommended.

Note that the above remarks are concerned solely with average element aspect ratio. The minimum aspect ratios for the *Balljoint* and *Block & Prism* problems were much smaller than the minimum (which was also the average in this case) aspect ratio for the *Cube* problem with $l/lz = 100$, but clearly the *Cube* problem was much more difficult to solve than the former two problems.

5.3 Direct Solve Results

A direct solve was attempted for the $10 \times 10 \times 10$ *Cube* ($l/lz = 10$) and for each of the five complex geometry problems using the Yale Sparse Matrix Package [38]. Minimum degree ordering was used for each of the direct solves. Due to memory constraints (maximum of 128 Meg available), only three of the five complex geometry test problems could be completed (Table 10).

For the *Balljoint*, *Multi-Material Bracket*, and *Cube* problems, the direct solve cost from six to seventeen times the CPU time to complete compared to the recommended iterative technique (direct solve of A_{vv}, $ILU_v(\infty, 10^{-3})$). The direct solve of the *Block & Prism* required only three times the CPU time compared to the iterative technique because of a very small amount of fill which occurred. The remaining two problems (*Bushing* and *Pipe*) required too much memory for a direct solve to be performed.

The solutions for each problem from both the direct and the iterative approaches were

Table 10: Direct Solve vs. Suggested P1 Technique

	CPU seconds (iterations)					
Preconditioning	$10 \times 10 \times 10$ Cube	Balljoint	Block & Prism	Bushing	Multi-Material Bracket	Corroded Pipe
$ILU_v(\infty, 0.0)$ $ILU_m(\infty, 10^{-3})$	437 (48)	496 (66)	465 (85)	1726 (52)	1521 (71)	1889 (105)
Direct [38] Solve	7771	3091	1361	*	14170	*

∗*Insufficient memory available to complete a direct solve.*

compared. This comparison was done by calculating:

$$\frac{\|X_{direct} - X_{iterative}\|_\infty}{\|X_{max}\|_\infty}, \frac{\|Y_{direct} - Y_{iterative}\|_\infty}{\|Y_{max}\|_\infty}, \frac{\|Z_{direct} - Z_{iterative}\|_\infty}{\|Z_{max}\|_\infty}$$

where:

- X, Y, and Z are vectors of the calculated x, y, and z displacements for each node
- subscripts *direct* and *iterative* refer to method used to calculate displacement and
- subscript *max* means the maximum displacement from both the iterative and direct solution.

For all problems, the maximum disagreement between solutions was less than 0.001 (or 0.1%) and was often as small as 0.01%. Thus the iterative solution can be expected to have roughly three or more significant digits of the direct solution, with the applied tolerance (19).

It can be concluded that the recommended iterative solver will provide a large CPU cost savings, while requiring less memory.

6 Conclusions

The performance of PCG iterative methods for linear elasticity is highly dependent on the average element aspect ratio. For aspect ratio near unity, many preconditioning methods will be successful. However, as the aspect ratio becomes small, the iterative methods encounter difficulty unless powerful preconditioning methods are used. The test problems used in this study had average tetrahedral aspect ratios varying between 0.78 and 0.02.

For unstructured grids with quadratic tetrahedral elements, a robust iterative method for a wide range of aspect ratios uses a hierarchical basis block preconditioner (**P1**), and

a direct solve of the vertex block ($ILU_v(\infty, 0.0)$), coupled with a drop tolerance incomplete factorization ($ILU_m(\infty, 10^{-3})$) of the midpoint block. This method can be fine-tuned by using a coarser or finer drop tolerance for situations where the average aspect ratio is known/expected to be be better or worse than 0.50 respectively. Level based ILU methods were not generally competitive with the drop tolerance ILU techniques.

With this type of preconditioning, the iterative methods outperformed direct solution methods by a factor of between three and seventeen times in terms of CPU cost. For the larger problems, direct solutions could not be obtained due to the excessive memory requirements ($>$ 128 Meg) for the direct solver. In those cases where direct solution methods succeeded, the iterative and direct solutions for the displacements agreed to three figures.

The hierarchical basis **P1** block preconditioner with $ILU_v(\infty, 0.0)$,$ILU_m(\infty, 10^{-3})$ generally outperformed a level based or drop tolerance based incomplete factorization preconditioning (**P0**) of the nodal basis stiffness matrix. Also, there did not appear to be any advantage to using the more accurate block preconditioner (**P2**) with the hierarchical basis.

Faster convergence rates can be achieved for meshes with average aspect ratios above 0.58 by using an incomplete factorization for both A_{vv} and A_{mm} regions. $ILU_v(\infty, 10^{-1})$ or $ILU_v(\infty, 10^{-2})$, and $ILU_m(\infty, 10^{-3})$ or $ILU_m(\infty, 10^{-4})$ seem to be the best combinations.

There are two standard methods for ensuring that the incomplete factorization of a symmetric positive definite matrix produces positive diagonal pivots. The *J & M Add* method [23] adds the dropped terms in the incomplete factorization to the diagonal. This method is guaranteed to produce positive pivots, but overestimates the amount necessary to add to the diagonal, and hence may produce a poor preconditioner. The alternative, is to keep adding a small relative amount to the diagonal until the incomplete factorization succeeds (see equation (8)) [7]. The situation regarding these two methods is not altogether clear. For a hierarchical basis block preconditioner (**P1**), with a small drop tolerance for the ILU of the midpoint block (ILU_{mm}) the method of equation (8) was generally superior to the *J & Add* method. However, if a larger drop tolerance (or low level) is used for the ILU of the midpoint blocks, then there are some cases where the *J & M Add* method does succeed, while the technique of equation (8) fails due to an excessive number of attempts to incompletely factor the midpoint block.

For fully three-dimensional objects, available mesh generators appear to be capable of producing tetrahedral meshes with average element aspect ratios greater than 0.50, in which case we can expect good performance from a block hierarchical basis preconditioner compared to a direct solver. However, even for extreme element aspect ratios ($\simeq$ .02), the block hierarchical basis preconditioner will also outperform a direct solver if a small drop tolerance is used for the midpoint preconditioner. Such extreme aspect ratios may arise in full three-dimensional modelling of plates and shells constructed of composite materials.

References

[1] M.P. Robichaud and P.A. Tanguy. Finite element solution of three dimensional incompressible flow problems by a preconditioned conjugate residual method. *Int. J. Num. Meth. Eng.*, 24:447–457, 1987.

[2] P. Chin, E.F. D'Azevedo, P.A. Forsyth, and W.P. Tang. Preconditioned conjugate gradient methods for the incompressible Navier Stokes equations. *Int. J. Num. Meth. Fluids*, 15:273–295, 1992.

[3] O.P. Iliev, M.M. Makarov, and P.S. Vassilevski. Performance of certain iterative methods in solving implicit difference schemes for the 2-d Navier-Stokes equations. *Int. J. Num. Meth. Fluids*, 33:1465–1479, 1992.

[4] A. Behie and P.A. Forsyth. Incomplete factorization methods for fully implicit simulation of enhanced oil recovery. *SIAM J. Sci. Stat. Comp.*, 5:543–561, 1984.

[5] S.C. Eisenstat, H.C.Elman, and M.H. Schultz. Block preconditioned conjugate gradient like methods for numerical reservoir simulation. *SPE J. Res. Eng.*, 3:307–312, 1988.

[6] B. Brusson and V. Sonnad. A comparison of direct and iterative methods for sparse unsymmetric systems of linear equations. *Int. J. Num. Meth. Eng.*, 28:801–815, 1989.

[7] T. Manteuffel. An incomplete factorization technique for positive definite systems. *Math. Comp.*, 34:473–497, 1980.

[8] N. Wiberg and P. Moller. Formulation and solution of hierarchical finite element equations. *Int. J. Num. Meth. Eng.*, 26:1213–1233, 1988.

[9] C. Farhat and N. Sobh. A coarse/fine preconditioner for very ill conditioned finite elemnt problems. *Int. J. Num. Meth. Eng.*, 28:1715–1723, 1989.

[10] M. Jung, U. Langer, and U.Semmler. Two level hierarchically preconditioned conjugate gradient methods for solving linear elasticity finite element equations. *BIT*, 29:748–768, 1989.

[11] F. Angeleri, V. Sonnad, and K. Bathe. Studies of finite element procedures - an evaluation of preconditioned iterative solvers. *Comp. Struct.*, 32:671–677, 1989.

[12] J. Mandel. Two level domain decomposition preconditioning for the p-version finite element method in three dimensions. *Int. J. Num. Meth. Eng.*, 29:1095–1108, 1990.

[13] L.H. Tan and K. Bathe. Studies of finite element procedures - the conjugate gradient and gmres methods in ADINA and ADINA-F. *Comp. Struct.*, 40:441–449, 1991.

[14] S. Foresti, S. Hassanzadeh, H. Murakami, and V. Sonnad. A comparison of preconditioned iterative methods using rapid operator application against direct solution methods. *Int. J. Num. Meth. Eng.*, 32:1137–1144, 1991.

[15] E. Barragy and G.F. Carey. Preconditioners for high degree elements. *Comp. Meth. Appl. Mech. Eng.*, 93:97–110, 1991.

[16] E.L. Poole, N.F. Knight, and D. Dale. High performance equation solvers and their impact on finite element analysis. *Int. J. Num. Meth. Eng.*, 33:855–868, 1992.

[17] V.E. Bulgakov and M.Y. Belyi. On the multi-grid technique for solving three dimensional boundary value engineering problems. *Int. J. Num. Meth. Eng.*, 33:753–764, 1992.

[18] E.J. Barbero. A 3-d element for laminated composites with 2-d kinematic constraints. *Comp. Struct.*, 45:263–271, 1992.

[19] E.F. D'azevedo, P.A. Forsyth, and W.-P. Tang. Ordering methods for preconditioned conjugate gradient methods applied to unstructured grid problems. *SIAM J. Matrix Anal. Applic.*, 13:944–961, 1992.

[20] E.F. D'Azevedo, P.A. Forsyth, and W.-P. Tang. Towards a cost effective ILU preconditioner with high level fill. *BIT*, 32:442–463, 1992.

[21] A. Jennings and G.M. Malik. Partial elimination. *J. Inst. Maths. Appl.*, 20:307–316, 1977.

[22] N. Munksgaard. Solving sparse symmetric sets of linear equations by preconditioned conjugate gradients. *ACM Trans. Math. Soft.*, 6:206–219, 1980.

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

[24] E.F. D'Azevedo, P.A. Forsyth, and W.-P. Tang. Drop tolerance preconditioning for incompressible viscous flow. *Int. J. Comp. Math.*, 44:301–312, 1992.

[25] O. Axelsson and I Gustafsson. Preconditioning and two level multigrid methods of arbitrary degree of approximation. *Math. Comp.*, 40:219–242, 1983.

[26] R.E. Bank, T.F. Dupont, and H. Yserentant. The hierarchical basis multigrid method. 1987. ZIB Technical report, Berlin.

[27] O. Axelsson and P.S. Vassilevski. Algebraic multilevel preconditioning methods I. *Num. Math.*, 56:157–177, 1989.

[28] S. Foresti, G. Brussino, S. Hassanzadeh, and V. Sonnad. Multilevel solution method for the p-version of finite elements. *Comp. Phys. Comm.*, 53:349–355, 1989.

[29] O. Axellson and P.S. Vassilevski. A survey of multilevel preconditioned iterative methods. *BIT*, 29:769–793, 1989.

[30] O. Axellson and P.S. Vassilevski. Algebraic multilevel preconditioning methods II. *SIAM J. Num. Anal.*, 27:1569–1590, 1990.

[31] J.H. Bramble, J.E. Pasciak, and J. Xu. Parallel multilevel preconditioners. *Math. Comp.*, 53:1–22, 1991.

[32] P.S. Vassilevski. Hybrid V-cycle algebraic multilevel preconditioners. *Math. Comp.*, 58:489–512, 1992.

[33] O.C. Zienkiewicz, J.P. Gago, and D.W. Kelly. The hierarchical basis concept in finite element analysis. *Comp. Struct.*, 16:53–65, 1983.

[34] G.F. Carey and E Barragy. Basis function selection and preconditioning high degree finite element and spectral methods. *BIT*, 29:794–804, 1989.

[35] A. Behie and P.A. Forsyth. Comparison of fast iterative methods for symmetric problems. *IMA J. Num. Anal.*, pages 41–63, 1983.

[36] D. Kershaw. The incomplete choleski conjugate gradient method for the solution of systems of linear equations. *J. Comput. Phys.*, 26:43–65, 1978.

[37] I.S. Duff, A.M. Erisman, and J.K. Reid. *Direct methods for sparse matrices*. Oxford, 1986.

[38] S.C. Eisenstat, M.C. Gursky, M.H. Schultz, and A.H. Sherman. Yale sparse matrix package I: the symmetric codes. *Int. J. Num. Meth. Eng.*, 18:1145–1151, 1982.

[39] Barry Joe. Tetrahedral mesh generation. University of Waterloo Scientific Computing Seminar, 1992.

[40] Structural Dynamics Research Corporation. Ideas VI Solid Modeling & Ideas VI Finite Element Modeling Packages, (1992).

[41] B. Chouchaoui. Evaluating the remaining strength of corroded pipes. Ph.D. Thesis, Mechanical Engineering, University of Waterloo. September 1992.

7 List of Figures

Figure 1: Surface Mesh of the Balljoint

Figure 2: Surface Mesh of the Outer Block with Hole for Prism

Figure 3: Surface Mesh of the Prism

Figure 4: Surface Mesh of the Bushing

Figure 5: Surface Mesh of the Bracket Face and Base Plates

Figure 6: Surface Mesh of the Bracket Supports

Figure 7: Surface Mesh of the Corroded Pipe